The Use of Systems Engineering Principles for the Integration of Existing Models and Simulations

By

Robert Luff

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of PhD

Electronic, Electrical, and Systems Engineering,

Loughborough University

March 2017

Loughborough University

ABSTRACT

With the rise in computational power, the prospect of simulating a complex engineering system with a high degree of accuracy and in a meaningful way is becoming a real possibility. Modelling and simulation have become ubiquitous throughout the engineering life cycle, as a consequence there are many thousands of existing models and simulations that are potential candidates for integration. This work is concerned with ascertaining if systems engineering principles are of use in the support of virtual testing, from desire to test, designing experiments, specifying simulations, selecting models and simulations, integrating component parts, verifying that the work is as specified, and validating that any outcomes are meaningful. A novel representation of systems engineering framework is proposed and forms the bases for the methods that were developed. It takes the core systems engineering principles and expresses them in a way that can be implemented in a variety of ways. An end to end process for virtual testing with the potential to use existing models and simulations is proposed, it provides structure and order to the testing task. A key part of the proposed process is the recognition that models and simulations requirements are different from those of the system being designed, and hence a modelling and simulation specific writing guide is produced. The automation of any engineering task has the potential to reduce the time to market of the final product, for this reason the potential of natural language processing technology to hasten the proposed processes was investigated. Two case studies were selected to test and demonstrate the potential of the novel approach, the first being an investigation into material selection for a squash ball, and the second being automotive in nature concerned with combining steering and braking systems. The processes and methods indicated their potential value, especially in the automotive case study where inconsistences were identified that could have otherwise affected the successful integration. This capability, combined with the verification stages, improves the confidence of any model and simulation integration. The NLP proof of concept software also demonstrated that such technology has value in the automation of integration. With further testing and development there is the possibility to create a software package to guide engineers through the difficult task of virtual testing. Such a tool would have the potential to drastically reduce the time to market of complex products.

ACKNOWLEDGMENTS

I have been fortunate to have had the support of many fine individuals throughout my time at Loughborough University, however there are some that deserve special thanks.

To Ron Summers, for his guidance and endless patience with my personal challenge of writing in an academic style, as well as his continued guidance throughout my PhD studies.

To the other PhD Students involved in the PSi project, for their lively debate and thought provoking questions.

To family and friends for their support and understanding throughout the many vacant conversations, as well as their persistence in trying to understand what I have been working on for the past three years.

Finally, to Jaguar Land Rover for presenting the opportunity to be part of the PSi project, and the support that made this research possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

TERM	DEFINITION
Abstraction	This is a concept by which the behaviour of a system can be conceived and judgements made without the need to fully understand all of the exact workings of the system.
Actor	An individual or role that interacts with the system of interest.
ANN	*Artificial Neural Network:* a mathematical model consisting of interconnected process units and weights.
API	*Application Programming Interface:* A defined set of functions and procedures to facilitate the creation of applications which access the features, or data of a specific existing application.
Black box understanding	An understanding of the inputs and outputs of a system combined with a functional understanding of how the system should behave, without a detailed understanding of how it operates.
Boundaries	The demarcation between two distinct areas of interest
Brownfield	When referring to an engineering situation it indicates that there are existing solutions, components, factors, or resources that need to be taken into consideration.
C	A procedural computer language
C#	An object oriented computer language
CFD	*Computational Fluid Dynamics:* a mathematical method for representing fluids which are moving in, on, or around a system that is being investigated.
Complex/ity	A descriptive term used to capture the behaviour of the interactions between system components as a result of the variation of the: components, control mechanisms, scales, communications, and number of elements.
Co-simulation	The use of two or more simulations that share or exchange values during execution.
COTS	*Commercial Off The Shelf:* sub system/s are outsourced to external companies to design and produce.
CPU	*Central Processing Unit:* a general purpose processor that can be used for many different computations.
CSV	*Comma Separated Values*: a commonly used file type for the saving and communication of data.
Data set	A collection of data that has been captured in a form that can be interrogated e.g. table, list, csv file.
Data transfer	The communication of data from one location to another.
DCPS	*Data Centric Publish Subscribe:* an approach to sharing data across a network.
DDS	*Data Distribution Service:* an OMG standard concerned with the transfer of data between multiple points involving different hardware and software.

Dependence	A resource that a simulation or model requires to operate that is external to it.
Development environment	Computer software used to create models and simulation software.
DLRL	*Data Local Reconstruction Layer*: a layer of the OMG DDS specification focused on the application level of the approach.
DMSO	*Defence Modelling and Simulation Office:* has been renamed the Modelling and Simulation Coordination Office. This falls under the jurisdiction of the USA Department of Defense.
Domain expert	An individual that is well versed, understands, and practices within a specific area.
Downstream	A term to indicate that decisions or actions taken at a point will directly affect work later on in the product lifecycle.
ECU	*Electronic Control Unit* an electronic device that is used to control elements of a system.
Emulation	The process of making one thing seem or act like another.
Environment	A boundary within which a system operates.
EPSRC	*Engineering and Physical Sciences Research Council:* an agency for funding research in engineering and the physical sciences in the UK.
Equation	A mathematical representation of a phenomenon.
exe	A common file extension denoting an executable file, suitable for Microsoft Windows or Dos operating systems.
FAMOS	*Framework for Adaptive Modelling and Ontology-driven Simulation:* a custom made ontology for the capture of integrated models and simulations.
FEA	*Finite Element Analysis:* A mathematical method involving breaking down the system under analysis into a number of discrete elements that are analysed individually.
Feature creep	The gradual increase in required functionality of the system compared to the initial requirements.
Federated system	In HLA terminology, a set of simulations that are interoperating is a federation.
Federates	Component simulations which are a part of a federated system.
Fidelity	A concept that compares the behaviour of a system to that of a phenomenon that it is representing.
FMI	*Functional Mock-up Interface*: a tool-independent standard to support model exchange and co-simulation.
FMU	*Functional Mock-up Unit:* a functional model or simulation that complies with the FMI standard.
FPGA	*Field Programmable Gate Array:* a type of integrated circuit comprising logic gates, the connections of which are user defined.

GB	*Gigabyte:* a unit of measurement for digital memory that equals 1073741824 bytes.
General systems thinking	The foundational concepts of systems thinking established in the main by Ludwig von Bertalanffy.
GPGPU	*General Purpose computing using Graphics Processing Unit*: using the functionality of GPUs for many repetitive calculations rather than using CPU.
GPU	*Graphics Processing Unit:* this is dedicated hardware which is used for the processing of computer graphics.
Grey literature	Refers to documents that are not found in the public domain.
HIL	*Hardware In the Loop*: the use of physical hardware with sensors as part of a simulation.
HITL	*Human In The Loop:* using a human machine interface as part of a simulation.
HLA	*High Level Architecture:* an architecture that is intended to provide interoperability and reusability across all types of simulations.
HPC	*High Performance Computing:* This term is often given to computer systems running applications that operate in excess of a teraflop.
HTTP	*Hypertext Transfer Protocol*: it is the foundation of communications for the world wide web.
IEEE	A professional engineering association dedicated to advancing technological innovation.
In house	To conduct the task within the confines of the organisation in question.
in silico	Models developed within a virtual environment.
Integration	The act of bringing together component subsystems into one system.
Interface	The accessible data inputs and outputs available from a system.
Interoperability	The ability for the system to work without issue with other systems either currently or in the future. This requires a complete transparent understanding of the interfaces and how the system operates.
ISO	*International Organisation for Standardisation:* a publisher of International Standards.
JLR	*Jaguar Land Rover:* An automotive manufacturer
LAN	*Local Area Network:* A network that connects a limited number of computers.
Macroscale	The largest physical scale considered part of a system or area of interest.
Mathematical function	A mathematical equation that contains a time variable.
MB	*Megabyte:* a unit of digital memory measurement. One megabyte is one million bytes of information.
MBSE	*Model Based Systems Engineering:* an approach to systems engineering and its application to project tasks.
Mesoscale	The physical scale that sits between the largest and smallest scale within the system or interest.

Meta	An overarching higher concept.
Microscale	The smallest physical scale within the system or interest.
Model	An application of mathematical functions which produces a set output.
Model/ simulation Platform	The support needed to implement a model or simulation. This could be in the form of software, hardware, or even pen and paper.
Multi-scale	The simultaneous consideration of phenomena at different scales to elicit greater understanding.
NATO	*North Atlantic Treaty Organization:* an alliance between European and North American governments.
Natural language	A language that has been developed by its users without being formalised. It is often the transport medium used to communicate ideas. It can be audible (speech) or visual (written word) in nature.
NLP	*Natural Language Processing:* is the use of a computer to analyse natural language, can take the form or textual or audio analysis.
NP complete	*Non deterministic Polynomial time complete:* a term used in computational theory denoting a problem that can be verified quickly, however there is no known efficient way to find a solution.
OEM	*Original Equipment Manufacturer:* a company that produces products that are used as parts by other companies.
OMG	*Object Modelling Group:* an international, open membership, not-for-profit computer industry consortium that has task forces which develop enterprise integration specifications for software technologies.
Operational environment	The environment within a system operates.
OS	*Operating System:* low level software that handles the basic functions of a computer.
Parameter	A value that is set at the start of a simulation and remains static.
Platform	A series of products designed using a common set of components.
POC	*Proof Of Concept:* A demonstration of technology to discern if it its application is feasibility for a defined situation.
Point to point integration	The bespoke, one-off integration of two or more models or simulations by considering their inputs and outputs.
Problem space	The conceptually bound area that the problems which are to be solved reside within.
Python	A high-level general purpose computer programming language.
QoS	*Quality of Service:* the performance of a network. Metrics include latency, error rate, and ability to achieve maximum possible bandwidth use.
RAG	*Red Amber Green:* a method of rating statements based of the concepts of traffic lights.

RAM	*Random Access Memory:* the volatile memory that is used by computers to temporarily store data.
Real time	This term is used to describe a system that will operate with the perception of executing at the same rate as the real phenomenon being modelled.
Real world	The physical world that is experienced.
Requirement	A statement detailing what a potential solution is to be capable of achieving, or constraining the solution space which it is to operate within.
Runtime engine	The computational solver used to execute a code written using a specific development environment. It can be run separately from the development environment.
SEIS	*Systems Engineering In Integrated Simulations:* A defined process detailed in this work in chapter 3.
Semantics	Relating to models and simulation - The underlying meaning of a model or simulation as a result of the developer's viewpoint. The viewpoint includes but is not limited to, scale, timing, subject, implementation platform etc. Relating to linguistics – A branch of linguistics concerned with what is meant by the components of a language.
SESEMS	*Systems Engineering of Selection of Existing Models and Simulations:* A defined process detailed in this work in chapter 3.
SI unit	The International System of units. Based on the Meter for length, Kilogram for mass, Second for time, Ampere for electrical current, Kelvin for temperature, Candela for luminous intensity and Mole for the amount of a substance. All other SI units can be derived from these, defined units.
Simulation	A simulation is an experiment which uses a dynamic implementation of a mathematical model.
SOA	*Service Oriented Architecture:* a style of computer architecture that uses applications communicating with sections of code (services) which are available over a network.
SoI	*System of Interest:* the System that is the focus of work to be conducted.
Specialist	A domain expert that has focused on a specific aspect of their domain.
Stakeholder	An individual who interacts with the system of interest.
Standard	Engineering standards documents specify the characteristics, technical details, and other aspects that are to be met by the systems and process that are specified.
Sub-system	A section or component part of the larger system.
Syntax	A set of agreed up on rules, relating to the structure of a sentence covering, order of words, phrases and punctuation.

SysML	*Systems Modelling Language:* a general purpose graphical modelling language for specifying, analysing, designing, and verifying complex systems.
System	A combination of interacting elements organised to achieve one or more stated purposes.
System being emulated	The system that the model or simulation represents the behaviour of.
Systems methods	The application of systems thinking to produce a tool procedure or process.
Systems theory	Relates to the philosophy of general systems thinking.
TCP/IP	*Transmission Control Protocol / Internet Protocol:* The transmission protocol which is used for communication over the internet.
Trade off	A decision which has to be made with two or more desirable objectives where it is impossible to completely satisfy both objectives.
UML	*Unified Modelling Language:* a specification language for object modelling.
US DOD	*United States Department of Defense:* the military branch of the United States government.
User	The individual or group that interact with the system of interest
Validation	Provides objective evidence that the system of interest, when in use, fulfils its stakeholder requirements, achieving its intended use in its intended operational environment.
Variable	A value that is used within a simulation that has the potential to change over the duration of the simulation.
Verification	Provides objective evidence that a system or system element fulfils its specified system requirements, architecture, design and behaviour.
Viewpoint	The perception of a stakeholder or actor based upon their world view and life experiences.
White box understanding	A full and detailed understanding of exactly how the system operates.
XML	*Extensible Markup Language:* A machine readable and human readable descriptive language that forms part of the interoperability standard.

1 INTRODUCTION

1.1 CONTEXT

This study investigates the application of systems engineering to modelling and simulation with special reference to the challenges involved in the integration of existing resources. The work will cover the information needed and technologies that exist to support the integration task, and in so doing will identify gaps in current knowledge and understanding of the topic. An example of a 'gap' is the future opportunity to automate the process of integration using the methods exposed in this thesis section 5.

As will be seen in later chapters, the systems engineering discipline is being applied to real-world phenomena of ever increasing complexity. It therefore follows that the models and simulations used to represent these phenomena are also becoming more complex. The increasing numbers of elements and their inter-relationships makes the integration task all the more difficult. At the same time methods and tools used to perform the integration task are becoming more specialised to meet the demand for a holistic interpretation with increased accuracy. Equally there is a business driver to reduce time to market of innovative engineering products. This study responds to these challenges by investigating methods that take the complexity indicated above and produce ways of integrating models and simulations that respond to the business driver in both a timely and purposeful way. This study is sponsored by Jaguar Land Rover, as a consequence some of the examples used stem from automotive applications including one of the case studies adopted to test out novel methods for model and simulation integration.

Current engineering processes often use virtual simulation as a way to test new ideas and designs before physical prototype testing. Virtual prototyping is used as it is far cheaper and quicker to test out new ideas than producing resource heavy physical prototypes. By reducing the numbers of physical prototypes, companies reduce the overall developmental costs of new products or services. It is clear from even the most casual glance at engineering literature which has been published in the past 20 years, that the use of simulation for testing has gathered momentum and become increasingly commonplace. Many large Original Equipment Manufacturers (OEMs) are now in the position where they have thousands of models that mimic the behaviour of many different parts of current and past products. Models and simulations are still costly to produce as they require skilled modellers to construct them, specific software, and computers capable of executing them.

Now that organisations have reached the point where the majority of the piece parts of their products are simulated, there is the potential to test all of the component parts in a monolithic full product test. The prospect of such a test is enticing as it may be possible to identify and isolate interactions between subsystems that previously had not been perceived as they cross between

departments or development teams. Producing a full system simulation opens up the possibility to conduct holistic optimization of constraints.

Simulation does not just have to be used for testing the design of a system. Hardware In the Loop (HIL) has made it possible to push simulation further into the development process. It is possible to take real physical prototype components and replace sections with simulations. This style of simulation not only brings in real world data that the system under test is likely to experience, but also gives the engineers a reference point to validate any virtual models that have been developed up to this point. In the same way that HIL replaces physical components with simulations, HITL (Human In The Loop) replaces any model of the human operator with an actual human user/s. This allows for aspects of the user experience to be assessed before a full physical prototype is constructed.

In the recent past the complexity of engineering projects has increased at a rapid rate [1]. This growing complexity has been met with an increased growth of the complexity of the simulations of the products being designed. To enable engineers to produce these detailed models and simulations of specific domains, specialist simulation packages have been developed. This is a growing market and there are many new software tools being released regularly. These tools add variety and further enhance the complexity of any potential integrated solution.

In many situations, the way in which organisations attempt to cope with the increased complexity of the designs and solutions is to take a reductionist approach. The result of such an approach increases the number of piece parts. This increase then often results in an increase in the number of people working on the project. The increase in the number of people working on a single project creates further issues relating to shared understanding, organisation, and product management and can exacerbate complexity even further. By breaking the problem down it also allows the organisation to use COTS (Commercial Off The Shelf) technologies. Using COTS allows for sections of the product to be designed and produced by third parties. COTS also allows for the organisation to benefit from expertise that they do not have in-house. By adding in third parties to the design process the support structure of the project also increases in variety. From this consideration of current business practices it is clear that any potential solution must take into consideration the wider issues of integration such as the business practices of modern engineering organisations.

Traditionally one of the most significant barriers to producing a high fidelity, full product simulation has been the computational power required. There are many reasons as to why the computational burden is high for such a task and these are discussed in section 6.2.10. It is only now with the advances in high performance computation that integrated simulations and co-simulations have the real potential for more than just the most vital test cases. This possibility stems from the significant computational power now available due to reliable distributed computing, high performance clusters, and cloud computing.

1.2 SYSTEMS ENGINEERING AND THE PROBLEM SPACE

The systems engineering aspect of this work uses a multi-disciplinary approach which aids in not only understanding the problem space but also supports the exploration of potential solutions and respective implementation technologies. Throughout this work various established systems engineering methods have been used. Additional systems methods have been created and developed as part of this work.

1.2.1 UNDERSTANDING OF THE PROBLEM SPACE

A critical part of the systems engineering approach is to understand the problem situation. The understanding of the problem space is the foundation for all further work. Any misconceptions regarding the problem situation can result in sub optimal solutions being formulated. It is indeed possible for such misconceptions to be the cause of potential solutions to be formulated that do not solve the problems that they were intended to.

For this research, multiple methods have been used to gain the best possible understanding of the problem situation and its environment. As this research is focused around the implications of using systems engineering, any potential solution cannot be domain specific. Any proposed methods are to be applicable in many situations across multiple domains. For this reason the information used originates from multiple domains so as to allow for a wide net of possible solutions to be considered. Within this problem space there are issues relating to the effective gathering of accurate information. Despite this, a detailed literature review will be conducted that incorporates both white and grey literature as well as informal communications with engineers from multiple disciplines and domains.

1.2.2 SOFT SYSTEMS METHODOLOGY

Soft systems methods focus on human activity systems and have proved themselves to be a worthy approach to understand the human perspectives in a system. Such successful soft systems method can be found in the literature [2].With soft systems methods having been used in similar ‘messy’ situations, some of their aspects have been applied as a means to establish a solid foundation of understanding of the problem space. This understanding facilitates the formulation of meaningful and purposeful aims and objectives.

1.2.3 RICH PICTURES

The rich picture was proposed by Peter Checkland as a part of his seven stage soft systems methodology [2].The purpose of a rich picture is to present a holistic view of the problem situation that can be easily understood by stakeholders. The rich picture is a visual representation of an observer’s viewpoint of the problem space where they deal with both the physical (hard) issues and the soft issues that include the opinions of stakeholders. The rich pictures that were created for this problem space can be seen in Figure 1.1. These images were verified by

showing them to engineers currently involved in model and simulation integration tasks. Alterations were made to capture the additional information as it became apparent during this verification process.

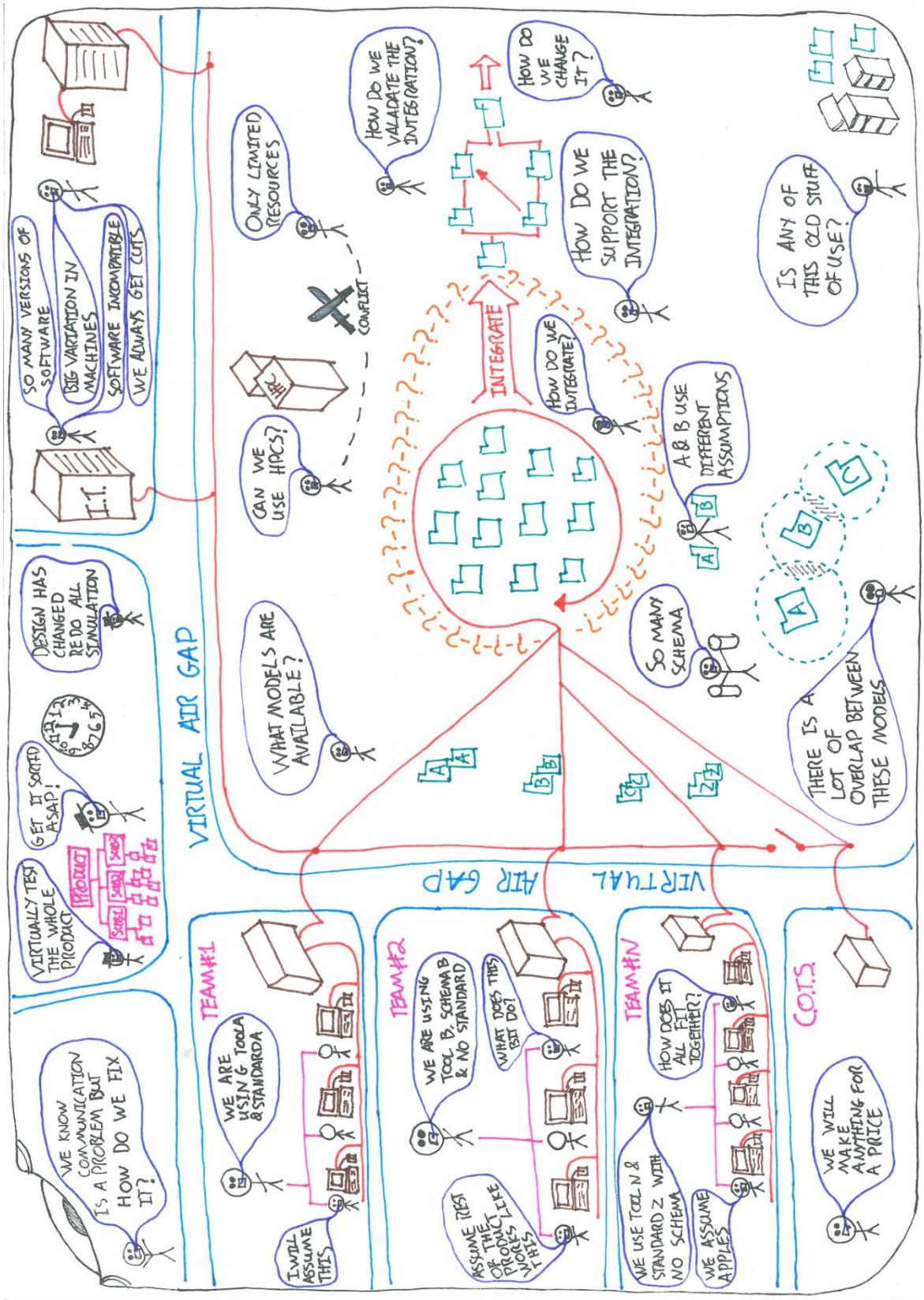

Figure 1.1Rich picture of the problem space.

1.2.4 PROBLEM THEMES

Once verification of the rich pictures had been established the next stage of Checkland's methodology could be implemented, that is the identification of problem themes. The purpose of identifying problem themes is to encapsulate aspects of the problem space in a clear, concise manner. The problem themes are textual representations of issues that have been taken from the rich pictures. Some of the issues in the problem space are not new as in the 1980s, Beer posed a research question that remains an open problem,

> *"How is it possible for multiple control elements, human or mechanical, each one possessing only limited powers of perception, computation and action, to achieve the enormous tasks of regulation needed to achieve complex purposes, or even any kind of identifiable continuity - that is to say, stability - in turbulent, noisy, and sometimes aggressively competitive environments?"* [3].

The problem identified is an element that appears in the problem space of this study.

The problem themes considered here are:

- Push from management [for high fidelity, full system, testing]
- Complexity of models and simulations
- Software environments
- Model ownership
- Communication between modellers
- Will to re-use existing models and simulations

The identified problem themes were verified with the same engineers that verified the accuracy and completeness of the information in the rich pictures. The engineers agreed that the problem themes were real and indicative of the current modelling and simulation domain. These definitions are to be used to formulate any potential solution throughout the research.

1.3 THE BOUNDS OF THE PROBLEM SPACE

To define factors that are within and outside the bounds of the research consideration is given first to the boundaries of the solution space. A graphical representation of the areas under consideration can be seen in Figure 1.2 below.

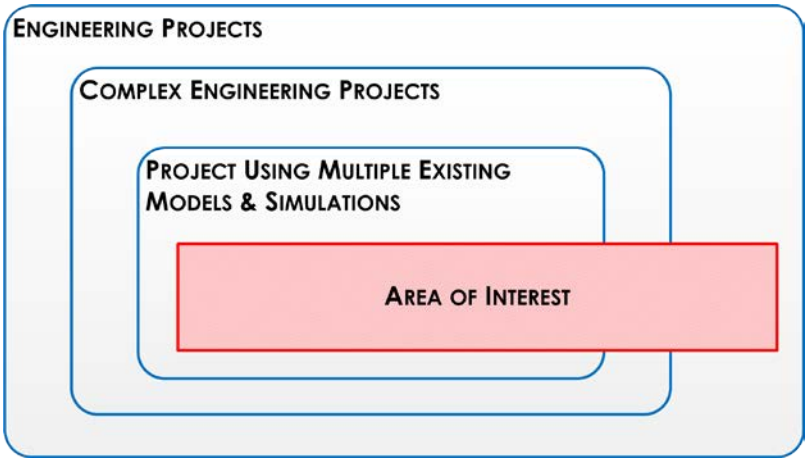

Figure 1.2: The area of interest for this research cuts across the identified areas of engineering projects.

As shown in Figure 1.2, the area of interest is encompassed within the overall umbrella domain of engineering projects, it also crosses through complex engineering projects, however predominately largely projects using multiple existing models and simulations. Studying the intersection between the identified boundaries provide interesting information. With the area of interest identified, consideration can be given as to what is within and outside of the bounds of this research.

1.4 SCOPE OF THE THESIS

This thesis follows the process of locating and developing technologies to assist engineers with the task of integrating existing models and simulations to provide a reliable mechanism by which full unit virtual tests can be produced repeatedly. A very high level representation of this can be seen in Figure 1.3 below, where a bucket of models exists, a selection is made, and the models integrated for the purpose of test.

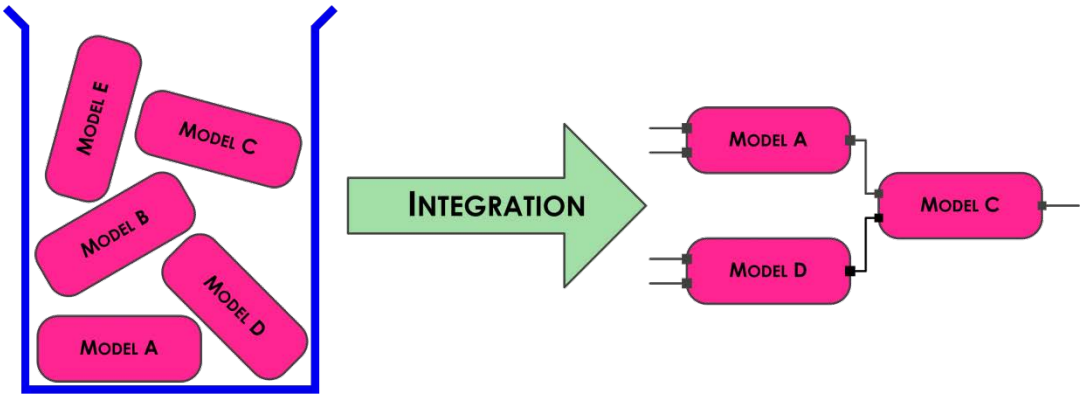

Figure 1.3: A high level representation of the scope of the research supporting the task of taking a bucket of existing models through to an integrated simulation.

The thesis contains methods that aid the integration engineer, from the desire to test a phenomenon through selection, integration, and throughout verification and validation. Any technology that is produced is not intended to be a fully useable product but rather a proof of concept for application in this domain. However any technologies that are produced will be tested with real engineering examples.

1.5 AIMS AND OBJECTIVES

Approaching the identified problem space from a systems engineering perspective results in a particular viewpoint being formulated. In this instance by using the first stages of Checkland's methodology, a large proportion of the viewpoint can be communicated between individuals. From the Rich Pictures and identified problem themes, the aim and objectives for this study were constructed.

1.5.1 AIM

Provide a complete end to end process for successful, meaningful, and purposeful integration of existing models and simulations from a repository.

1.5.2 OBJECTIVES

A. Determine the potential of automatic identification of model and simulation dependencies, and the respective assumptions encapsulated.
B. Discern the means by which levels of abstraction can be identified and the impact on the task of integration established.
C. Develop a means by which the generation and creation of middleware can be automated.
D. Validate the designed process with a case study using real world models.

1.6 Structure of the Thesis

This research is considered to be a project in its own right, and as such the way in which systems engineering principles are mapped to the chapters of this thesis can be seen in Figure 1.4 below.

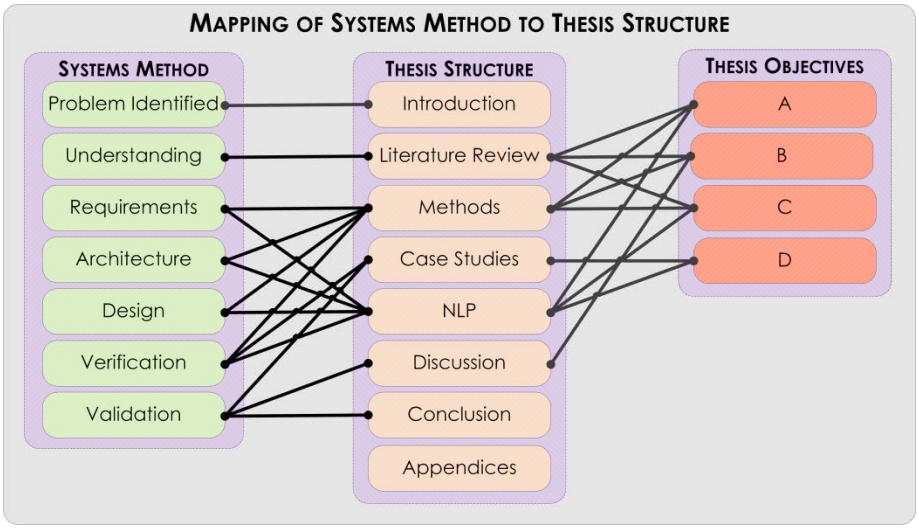

Figure 1.4: The mapping between the systems method, the structure of this thesis, and the mapping to objectives.

The mapping between the systems method and the thesis structure is not as simple as a one-to-one mapping. Both the methods and NLP chapters have elements of requirements, architecture, design, and verification. This is due to both of these chapters containing distinct sections of work that build from the initial understanding. Due to the approach taken, verification occurs in chapters where there is a reference point of work that has been created, hence why it is mapped to Methods and NLP. Validation is only conducted on the complete system in its operational environment, hence why validation is mapped directly to Case Studies. The results of the validation also feed into the discussion and conclusion chapters. The thesis objectives are also mapped to the thesis structure. Each of the objectives maps to two or more sections of the thesis.

1.7 Overview of Contributions to knowledge

The contributions to knowledge in this work are both in the methodological and process domains. For an in depth discussion of where each of the contributions are addressed see section 7.2.

The key contributions to knowledge proposed in this work are:

- A representation of systems thinking usage within an organisation
- The information needed for meaningful model and simulation integration
- A method of describing the levels of abstraction across a model or simulation integration
- A means by which to verify and validate the integration of two or more models or simulations
- Identification and potential solution to the dislocation between virtual modelling simulation testing and classical engineering
- The identification of requirements for a model and simulation being different from the requirements of the system being emulated
- A complete modelling and simulation requirements writing guide
- An end-to-end guide for engineers who are tasked with the virtual simulation and test of a potential design using existing models and simulations
- Natural Language Processing (NLP) technology has the real potential to aid in establishing if two or more models or simulations can be meaningfully integrated.

2 LITERATURE REVIEW OF MODELS, SIMULATIONS AND THEIR INTEGRATION

2.1 INTRODUCTION

This Chapter presents a literature review that investigates the challenges of model and simulation integration. The key areas that have been investigated are as follows.

- The reasons why models and simulations are used as part of the engineering project life cycle.
- The benefits of using integrated models and simulations as part of an engineering project.
- The methods used for model and simulation integration.
- The modelling techniques used in engineering and the effects on the integration task.
- How semantics and model assumptions are linked to whether an integration is meaningful or not.
- The challenge of crossing between spatial scales.
- The current state of communications within engineering projects, the forms that communication takes, and the tools that have been developed to assist.
- The current types of modelling and simulation tools that are in common use within engineering.

It is recognised that any literature review is essentially incomplete. However, it is hoped that the comprehensive nature of what is attempted here satisfies both the breadth and depth of background knowledge that is necessary to understand the problem space to a sufficient degree that is purposeful to satisfy the stated aims and objectives.

2.2 THE USE OF SIMULATION

Simulation is becoming ever more present in the engineering life cycle of complex products and services, however it is a time consuming and costly endeavour. Most simulations are primarily used to gain an understanding of phenomena that form a part of the problem that is being investigated. Hence the most common use of modelling and simulation is to investigate the behaviour of a potential solution, its environment, or a solution in its intended environment. By conducting such activities it is intended that a greater understanding can be gained in a more timely and cost effective way than using physical prototype testing [4]. In this case physical testing involves conducting tests in the intended operational environment, for example data collection of the environment or the testing of a potential design in the intended operational environment. Simulation has the benefit of allowing for large numbers of factors to be analysed and varied with little resource overhead which can be far cheaper than equivalent physical testing [5]. Regardless of the means of testing it is intended that the additional information will directly impact upon the design of the system being created.

In the automotive industry simulation and virtual testing is often used as part of the product design phase of the developmental lifecycle. This virtual testing has taken hold in this domain due to today's competitive market where time and cost is at a premium. Simulation allows multiple tests of many potential solutions to be conducted in a relatively short time frame, and more critically allows the understanding gained to be made while reducing the number of expensive prototypes that have to be developed. Bold statements are made in the literature such as,

> *"Execution speed is roughly 500 times faster than real-time…"* [6];
>
> *"…several weeks of engineering time can be performed in a matter of hours"* [7].

Such statements demonstrate not only the physical possibilities of the approach but also the verve and confidence that engineers currently place in simulation as a design philosophy. Due to these perceived benefits, it is common within the automotive industry for each sub-system of the product being developed to be simulated before any physical implementation is ever considered.

2.2.1 HIGH FIDELITY SIMULATION

The concept of fidelity can be defined as the extent to which the simulation output reflects that of the phenomenon that it has been created to represent. This definition means that it is a relative term rather than a static scale that all models can be held against. The reason to define fidelity in this way is to assure that as a simulation progresses, the concepts that are discussed are not lost

within the semantics of specific simulations and where they fit on a static scale of fidelity.

The idea behind using high fidelity simulation is to produce a simulation that uses a detailed understanding of the exact ways in which the phenomenon being simulated operates and to replicate this behaviour as closely as possible in an attempt to make the results of the simulation match the real world. This however is not necessarily as straightforward as it may first seem.

As time progresses and our understanding of the ways in which the universe works improves, so too can our simulations of it. Hence as time progresses what once were considered high fidelity simulations may not now be. In practice, this may appear as older work being considered as inaccurate or of a lower fidelity than what can now be produced. As organisations often do not openly share all of their knowledge relating to the modelling and simulation of phenomena, situations can occur where organisational understanding of specific situations are different. Hence what they consider to be a high fidelity simulation regarding specific phenomena is also different. This results in the term 'high fidelity' having a different meaning across domains but also a term that has a changing definition across organisations within the same domain.

From the literature various factors are alluded to when considering the fidelity of a model or simulation. The recurring trade-offs that have been identified for consideration include:

- Time to develop the simulation
- Tools available to conduct the simulation
- Current understanding of the phenomena
- Mathematical methods for mimicking the phenomena
- Run time of the final simulation
- Skills available within the organisation to produce such a simulation.

Due to the added investment in resources to successfully implement a high (rather than low) fidelity simulation, there is a question as to why an engineer would invest in the resources to create such a simulation. There is a trend for high fidelity simulations to be conducted on parts of the system being designed where current understanding does not allow for engineering judgement to bolster the results of low fidelity simulations. Alternatively, high fidelity models are used in situations where low fidelity models do not give the granularity of information that is needed for meaningful decisions to be made about the system being designed.

There are many teams of people working on means by which high fidelity simulations can be integrated and run as co-simulations. The reasons given as to why they wish to conduct high fidelity simulations vary from greater accuracy of simulating interconnected but mechanically different components [8], to

attempting to represent the behaviour of complex systems as a whole by modelling their component parts [9]. However, the common goal of all teams is to gain information.

The prospect of high fidelity simulations integration leads to the possibility for unknown interactions to be captured earlier in the design phase rather than waiting to find the issues later using resource heavy physical prototypes. This task looks to have the prospect of going further, making it possible to produce high fidelity simulations for all of the component parts of a vehicle [10]. This is a potential that requires many aspects to fall into line, including modelling methods, hardware, and software. On a more fundamental level, it will require a change in the way in which engineers approach modelling, simulation, and their testing.

From the grey literature there are many software tools that are being specifically designed for high fidelity modelling of specific situations. For example ANSYS is a general purpose high fidelity modelling and simulation tool [11] which has been developed specifically for simulating pumps [12]. Such tools are becoming more widespread, which can be used as evidence to infer their level of demand and use.

Topics regarding the integration of high fidelity modelling and simulation for co-simulation are currently either ignored or are only mentioned in passing. If any mentions or claims are made in the literature then evidence is not given to substantiate them. This void indicates that it is not currently possible, or that the competitive advantage of such a capability has been recognised and companies do not wish to publicise their capabilities. Either way this shows that this area is suitable for further research.

2.2.2 The Parts Being Integrated

There are many component parts of models and simulations which could potentially be integrated. Within the literature there are references to utilising the whole simulation, however such examples tend to be more apparent in the grey literature with tool vendors displaying case studies, such as Mentor Graphics when publicising their Flowmaster tool (the co-simulation of engine cooling systems) [13]. However the academic literature is more wary with consideration being given to the use of only some parts of existing models [14]. Bartholet [13] indicates that there are many component parts of a simulation that could be eligible for integration, from sections of code through to whole simulations.

2.3 CURRENT APPROACHES TO THE INTEGRATION OF MODELS AND SIMULATIONS

The concept of integrating models and simulations is not a new one, neither are the challenges of integrating existing models. Existing methods have been developed to help cope with the complexity of the integration task. The challenges are covered in detail in 0 and 2.6. However it is worth noting that there are only limited current methods of coping with the task of selecting possible components from a bank of existing models. Even this task has been found to be nowhere near as simple to conduct on a computer as it may at first seem. Work conducted in 2005 by Bartholet and colleagues used formal logic to prove that

> *"Even with the existence of an oracle that can in one step determine which objectives have been met by any component or set of components, component selection alone has been shown to be NP-Complete"*[15].

This is an important piece of information as if the problem is NP-Complete we do not have sufficient methods or computational power to solve it in a timely manner. This is potentially why the majority of the current methods ignore the selection process and give no assurance of semantic integration, but rather focus around the interconnection and data transfer between component parts. Such existing integration methods rely on engineering or scientific judgement to overcome these semantic issues rather than using automated computational processes.

There are issues relating to searching for tools and technologies that either conduct or facilitate the integration of models and simulations due to the lack of common terms between the technologies. From each engineering or scientific domain there is a different vocabulary that is used to describe the modelling concepts that they use. These tools and techniques can be called anything from 'simulation re-use' to 'high level architectures'. Searching for potential solutions means investigating different domains, from looking at modelling forest fires [16] to the addition of NATO viewpoints to US DOD architectures [17].

There are COTS integration software packages available to engineers, however due to the domain in which the author works any current or potential future capabilities of any COTS integration tool will not be explicitly discussed in this work. Generalisations, however, regarding the market offerings are identified. The available tools often fall into two broad categories; the first act as an orchestration engine such as Dassault Systemes application Isight [8]; and the second style is one built from the ground up for a specific type of analysis, such as Engin softs modeFRONTIER [18], which was designed primarily for multi objective optimisation. Regardless of if the tool was intended for general purpose integration and co-simulation, or for a specific purpose, many of these tools have

specific limitations regarding what they are capable of handling. Such limitations include but are not limited to: the number of potential component parts; the development environment or tool that was used to create the component part; the specific version of the package that was used to create the component part; and the communication protocols that can be used. The most significant issue with current COTS integration tools is that they give little to no assistance to aid in discerning if it is meaningful to connect two or more models or simulations.

By casting the search as wide as possible, an attempt was made to capture as many methods as possible, finding common themes, and asserting current state of art for model and simulation integration. From this investigation it has been found that there is a significant disconnect between academic studies and engineering practitioners. On the most part the academic literature makes bold claims about the capabilities of proposed methods and tools, however in practice these methods are either largely ignored (for a variety of reasons) or they only work in an impractically constrained problem space. However all is not lost for engineers working in this domain; there are instances where engineering tools have proved sufficient flexibility to allow for limited point-to-point integration. This does not mean that all of the academic literature has been disregarded in this study rather that each technology is commented upon for its suitability for the specified problem space based on the results of their use in real world situations.

With the ever decreasing cost per calculation and increasing overall computational power available to engineers, how to best use this capability in the product life cycle is a question yet to be answered. There is very little literature other than that produced by tool vendors that covers practical advice as to how to go about integrating models and simulations. Even less evidence is available to guide an engineer from conception through to testing as a complete end-to-end guide.

2.3.1 Current Integration Methods

At present there is no automated means by which models created in disparate environments can be integrated. There are some limited parts of the integration process that have been automated with success. The difficulties in integration of existing models and simulations in practice are covered later, see section 2.4. However at this stage it is worth noting that the ways in which some of the modelling and simulation packages are constructed make it impossible to integrate them with any other environment [14], or in some cases even other instances within the same environment. This is due to the way in which the software is constructed, there is no way of accessing the values that they use or produce via programming. Though such modelling and simulation environments are becoming increasingly rare, they still warrant a mention here.

Due to the difficulties faced when integrating models and simulations, instances where automation is demonstrated either only focus on the data handling

between component parts or they use model and simulation components that have been purpose built to be flexible with their IO and use a strict standard to reduce semantic issues. To reduce the difficulty of such automated integration they only use a constricted problem space. Many of the automated examples recognise that the solution may be sub-optimal for reasons discussed in section 2.6. The factors are not representative of the problem space that is defined in section 1.2.1. Such examples are not of concern to this work.

The implementation section has to be separated from the approach section to ensure that there is a clear distinction between the overarching way in which the model and simulation integration is being implemented and the details of each individual problem that have to be overcome when integrating two or more models or simulations, for such information section 2.4.

There are two common methods that are employed to aid in the manual integration of components: manually defined integration of models; and, simulations where an engineer specifies which component parts are to be connected, in what order, how they communicate; the orchestration of execution is often implemented in one of two ways that are discussed below.

The first commonly used method is the use of bespoke middleware to act as a means of communication between two or more models. Middleware code acts as a means of communication and can also be a way to orchestrate execution. The code for the middleware may be written in the development environment of one of the models being integrated [17] or in a completely unrelated environment using a completely different language and/or interface.

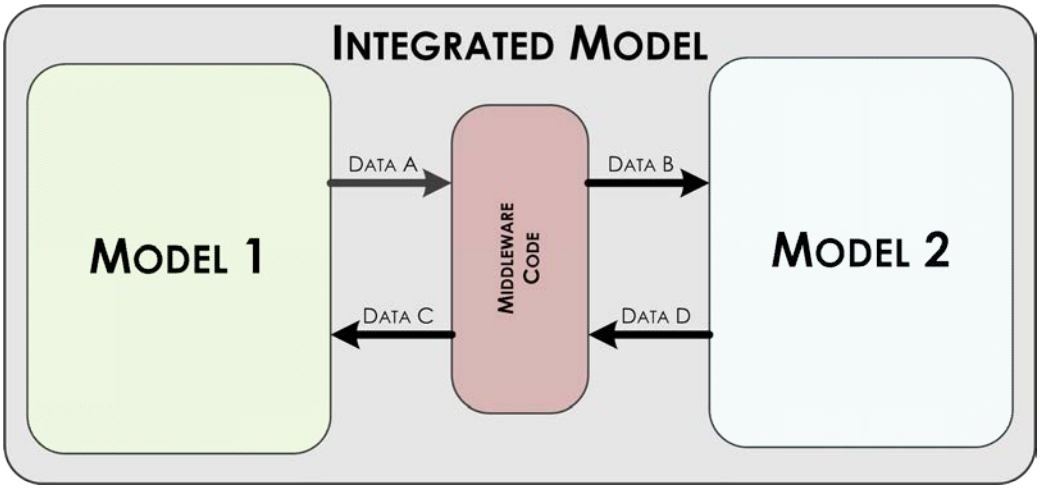

Figure 2.1 A representation of middleware between two models

In essence the middleware process involves finding a way to get the models and simulations to output data to an accessible port as well as receive data from external sources and daisy chain them together. A simple example of this can be seen in Figure 2.1. The middleware code sits between the two models and acts as an interface. Data A may contain the same information as Data B but in a

different format be that via a data transform or otherwise; the same goes for Data D and Data C. This can be easier said than done as not all tools have a means by which a programmer can access the variables and parameters in a desirable form, if at all. If using existing models or simulations with this method it may involve a considerable amount of re-work of the model and simulation to get them to a point where the necessary information is accessible (see section 2.4.2 for challenges of variable accesses during run time).

Middleware is used and demonstrated extensively throughout the literature; the technology is at a maturity where it has been widely commercialised. Some COTS middleware has been specifically designed to operate with two clearly defined software packages [13]. Others attempt to be more general in their approach by having multiple packages that they can aid in the communication between the component parts [8]. There are strengths and weaknesses in having integration middleware tools that can support multiple simulation environments. One of the most significant criticisms of such flexible methods is that they enable users to connect variables without knowing if such a connection is meaningful. However this argument can be used about any model or simulation integration. A limited integration tool has the advantage by only allowing the user to connect particular parts together hence reducing the likelihood of erroneous connection as well as reducing the number of potential instances where the middleware tool would be of use. One of the most significant weaknesses of middleware is that when purchasing a tool there is no absolute guarantee that it will continue to support all future and past versions of the modelling and simulation tools with which it communicates. Many tool vendors release new versions of their tools on a regular basis and only continue to support a few previous versions. Such factors have to be taken into consideration when selecting tools to use throughout the lifecycle of the project, as well as in the long term from an organisational perspective, considering that projects can run for many years.

Middleware can take both an active and passive role during the execution of the integrated solution. A passive role is considered as purely handling the communication between the constituent parts of the integrated solution relying on the internal capabilities of the component parts to handle any orchestration and analysis. The conversion of data types and the potential to change the format for communication is still considered in this classification to be a passive role, whereas active middleware can go as far as having its own mathematical solvers external to any of the simulation component parts. Some middleware tools also offer analytic capabilities to monitor performance of the component parts and the communication between them during run time execution.

In some cases, a limited loose form of integration is possible, even if there is no means of integrating the environments which the component parts are constructed within. Such loose methods are possible by using intermediate files that are written to by one component and read from by another. This allows for

an indirect communication between environments that otherwise share no common means of communication. This method works for processes in which the models or simulations either run sequentially or have a relationship that allows for a relay in passing between environments between time steps. However producing middleware for orchestrating such communications is far from simple. Instances where such approaches are used produce an output that is brittle, meaning any changes to the component parts can have dire consequences to the analysis as a whole.

The second commonly used method for integrating models and simulations is to import one model or simulation into another. This can be relatively straightforward (from a communication stand point) in the instance where all of the components were created using models or simulations from the same development environment. However this approach is far from straightforward if the models or simulations do not share the same development environment or even the same versions of the same environment. Making one model or simulation run in a different run time or development environment that it was created in is considered to be a model interchange. Some tools have the ability to import sections of code from other development environments. It is common that the native functions of one tool will only support a specific subset of another tool for a very specific version and even down to a specific set up. When issues arise with such interchange and exchange issues it can require both tool vendors making changes to their code allowing the two environments to exchange code. FMI is a standard that is gaining popularity amongst simulation packages for specifying model exchange [19], FMI is covered in more detail in section 2.3.3. A simple representation of the principle of running one model or simulation in another execution environment can be seen in Figure 2.2 below. The purpose of the execution environment wrapper is to transform the data that pass through and ensure that these data are in a usable form for its destination. This means that in Figure 2.2, Data A will have the same information as Data B but it may be in a different form, the same goes for Data D and Data C. It is poignant to note that despite Model 2 being within the execution environment, it may be running in its own environment and the Model 1 environment is calling the Model 2 environment as and when it is needed.

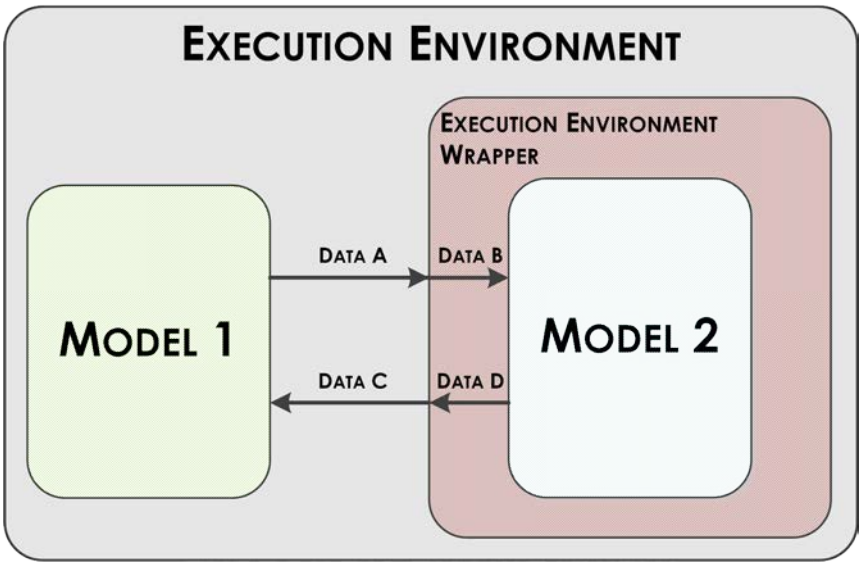

Figure 2.2 One model running in another models execution environment

Having one environment running code from another does not necessarily negate the issues of needing the programs installed on the machine that is running the simulation. Many development environments have a 'runtime engine' that generates the necessary information to run the simulations during execution. They handle configurations, schedules, mathematical solvers, data management and other operations [20]. The exact functionality of runtime engines of each tool is not widely publicised as they often form the majority of the innovation and hence competitive advantage that a simulation tool possesses. Simulation code may call a specific function that only the environment it was created in has and the relevant runtime engine can handle. Hence to run the model or simulation it will require the run time engine installed on the machine to run without exceptions. This can be the case for compiled code or dll files. Hence it may be necessary for multiple run time engines to be installed on a single machine, or for the runtime engines to be available over a network for an integrated solution to execute. If the simulation is to be distributed across a network, then the task of assuring that the machine that is running the code has access to the appropriate run time engines has to be taken into consideration. The considerations of run time environments and how to get the modelling and simulation tools to communicate adds another layer of complexity when considering the integration of models developed at different times using different software.

The issues of model exchange are such that there are active projects such as those at NIST (US National Institute of Standards and Technology) where the issue of model exchange is investigated and the claims of tool manufacturers tested. An example of this is the work being conducted with the leading UML tools currently available. A formal model checker with specific inputs has been developed [21]. The checker requires the tools from various manufacturers to produce standard models. These models are checked against a standard source file model which is defined by OMG standards. Issues remain with all of the major

tools tested. With the secrecy of how the tools work it can be difficult for tool manufacturers to get their tool to behave exactly the same way as another with the same inputs. From talks with engineering practitioners a repeated theme became evident; when an organisation is looking to integrate their models from different environments using the approach of importing code from one environment into another it can be a somewhat of a 'suck it and see' approach to see if the integration will be successful or not. Interestingly this is a topic that is not widely reported in literature.

Using either dedicated middleware or importing code from one environment to another are both currently labour intensive and have a considerable margin for both human and machine based error. This also highlights the issues of finding where errors arise during validation. The question becomes one of 'are the errors originating in the component parts, the way they are integrated, or the communication between them?' Without experienced users of the individual components it can be difficult to tell if there are any errors in the first place. If an error is detected it can be difficult to locate the source and often relies on engineering judgement, intuition, luck, or a mixture of all three. The two integration methods presented are currently used in both industry and academia, which indicates that they are at present the most preferable means of conducting model and simulation integration. Both approaches have been made to work and produce integrated simulations and have produced useable meaningful outputs.

The custom written middleware approach has the flexibility to be manipulated to integrate nearly any possible simulation or model, if the integration is meaningful or not is another matter (see section 2.6 for more detail regarding meaningful integration). This almost infinite flexibility means that this approach initially seems to be an attractive solution to the integration problem. Wrappers can be used to make multiple models or simulations all appear to each other as having everything in the correct form for communication which is a simplified version of how the FMI standard proposes a form of integration [22]. With FMI being adopted by many existing tools it allows for engineers to continue to use the tools that they are already trained and experienced in using. This reduces the disruption to an organisation which increases the likelihood of it being adopted. This does not eliminate the need for an integration engineer, however it does reduce the need for a specialist tool coder. Having all of the code in one environment allows for an *ad hoc* standard to be enforced due to the nature of putting the entire model in one environment. When integrating two or more models or simulations it may be necessary to introduce controls as to what component parts execute and in what order. Some tools such as Isight [8] have this additional functionality. However if custom bespoke middleware is used then, significant time and effort has to be invested to create the functions and verify that they behave as intended by those who selected them for use.

Irrespective of which of these methods are used, an architecture for the desired integrated simulation has to be in place before work is conducted. If an architectural pattern is not formulated and rigidly implemented, then the combined model can quickly become unmanageable due to the sheer number of interactions, not to mention the complexity of specific integrations. Having such a disorganised implementation can cause issues with determinism of the simulation, resulting in the confidence of the output of the simulation being negatively affected.

The current approaches take considerable time and effort for each individual integration task. One of the key areas that integration tool vendors focus upon in their marketing is the manual specification of the communication between elements, and the specification of the data used is a time consuming task [8], [13]. Even if this task is taken care of by a tool, the user still has to manually specify the IO connections between the component parts. Due to the current nature of the integration approaches the work is unlikely to be re-used for another project due to the resultant fragility of the point-to-point custom integrations.

Due to the way in which the development and execution environments are produced means that there can be integration problems not only with the different development environments in which each part is constructed but also with the different versions of the same development environments. The issue of versions of the same tool not working in harmony with each other results in further issues about using existing code. If existing code from past projects is considered for use it can require significant time and effort to bring the old code up to the version currently being used by the project. In some cases this is not possible without a complete re-write. The issue of versions of software is such that it is widely recommended within software-centric projects that specific versions of the programs and languages used are held constant throughout the entire course of the project. With engineering projects becoming ever more software-centric, specifying software versions is becoming as applicable to engineering projects as it is to software development. Variations in software versions have wider implications as it means that if a company wishes to integrate code from multiple departments then all developers have to have the same development environment with exactly the same version. This is not a trivial business or IT support task. On the other hand by using one development environment for full unit testing produces a single point of failure. If only one environment is used it can become difficult to mitigate any circumstance where the environment becomes unusable for any reason, for example if the software company who support it goes out of business.

Using a single environment for all of the code often means that there has to be a trade off with the functionality of the model or simulation as currently no one tool has all of the functionality that is required for an optimal, single simulation of modern complex systems. To increase the usage of their tools, vendors have

attempted to make sections of their tools accessible to other tools through Application Programing Interfaces (APIs). However getting two such tools working in harmony is not easy and has weaknesses such as ensuring the use of specific versions (as above) or when only specific libraries or a subset thereof are used.

There is a current drive in industry to find a means of running simulations over a network to take advantage of high performance parallel computing. This task is more suited to the use of middleware with a higher level architecture (not necessarily HLA) sitting above it. At present efforts are being made to implement such an idea using federated systems, see section 2.4.4 for more details.

With the methods presented so far there is the problem of how to handle a situation where there is a change. Often in a complex design there are changes in requirements throughout the project life cycle. A change in system design will have an impact on the requirements for the simulation and hence to the requirements of the component simulation parts. Such change means that the supporting integration code may also have to change. The result of a requirement change could be something trivial or it could mean that whole sections of the simulation have to be re-written. The flexibility of a simulation to change is directly affected by the means by which it has been implemented. The methods discussed here suffer from the potential to produce a very brittle integration, meaning any changes require considerable work to implement. Some of these issues can be limited by implementing a strict architecture (see section 2.4.5) or a strict standard (see section 2.3.3) to specify the interfaces which mean changes do not require the interface to be re-specified each time. Even with such methods, implementing both approaches has significant challenges such as to keep track of changes in the requirements and which model relates to which integration implementation.

One of the reoccurring features of the tools that engineers currently use for integrating models and simulations, such as MATLAB [23], Isight [8], and TestStand [24], is that they have a means of communicating internally. They are designed to aid communication with other environments though the user has to configure any external calls manually. Such manual configuration requires significant understanding of the component models or simulations, what each port represents, as well as how the data are composed.

There is a significant gap in the literature regarding these methods that concerns the way of ensuring the semantic similarity between the component models and simulations that are being integrated. Conveniently the issues of ensuring that the integration is meaningful is not mentioned in the supporting literature or it is dismissed as being outside of their research concern. For more information relating to the semantic issues see section 2.6.

2.3.2 Evaluation of The Integration Approaches

These approaches to integration are not without their shortcomings. Without careful planning such approaches can have significant issues throughout the rest of the lifecycle. The high degree of flexibility that comes from the bespoke nature of the integration using custom middleware initially looked promising, however the further a project develops, both in size and maturity, the more of a challenge management of the integration becomes. Putting all of the code into one development environment can solve a large proportion of the issues. However this approach can become unmaintainable as projects grow due to limitations of the possible component capabilities, and long term support of ever changing software. These approaches clearly have strengths and weaknesses however there is considerable room for improvement. The areas which have been identified from the literature for potential improvements are: organisation of middleware, long term software usage planning, guidance to those who are undertaking the task, any automated assistance to the integration task, and semantic assistance for connections.

2.3.3 The Role of Standards in Integration

One proposed means by which the complexity of the integration task can be reduced is the application of a standard. In essence a standard attempts to make aspects of a system adhere to a pre-determined format. The ISO organisation defines a standard as

> *"A document that provides requirements, specifications, guidelines or characteristics that can be used consistently to ensure that materials, products, processes and services are fit for their purpose."*[25]

When such an idea is applied to the model and simulation integration task they often focus on requiring that the interface of the model or simulation is specified and the internal workings conform to a set of specified constraints. One of the key areas where standards are used is in an attempt to enable models and simulations developed in isolation to integrate with each other.

Within the defined problem space (see section 1.3) there is no single common standard that covers it all. In isolated areas of the problem space there are standards that are used, for example: FMI, DDS, ISO 15288, ISO 15926-2, and ISO 42030. The work that proposes the use and benefit of such standards also often proposes the use of higher level architectures such as HLA, DDS, SOA, MODAF, or DODAF. As such, higher level architectures require uniformity for them to successfully operate. From the identified standards only FMI was specifically designed for model and simulation integration, more can be found on this in section.

It is not uncommon within large projects for a sub-system team to use a standard for their section of work while other standards are used elsewhere in the system lifecycle. Producing a single standard that could capture all of the work that

needs to be conducted through a product lifecycle is far from a trivial task. Such a single standard would not only have to cross multiple scales and domains while still remaining useable, but also be relevant to many possible solutions. There is also the challenge of balancing the level of specification, if too much is specified it restricts the potential solution space, while too little specification the standard loses its effectiveness.

Due to the established nature of many industries and the economic pressures that they face, enforcing a standard retrospectively is not only unreasonable but is also in some cases impossible. This is due to the vast number of components, the sparse or undetailed nature of any available existing documentation. This investment is further increased if the structure of the model has to be changed to make it complicit with a standard. Such re-work alterations can be resource heavy and hence should be avoided where possible.

Functional Mock-up Interface

A standard that is gaining traction within the model and simulation community is FMI. This standard is a tool independent standard that supports co-simulation and also model exchange [22]. FMI is a relatively new standard and currently there are two versions: the first was published in 2010; and the second version in 2014. There are currently 80 tools that have some capabilities of FMI version one and 42 tools that have some capabilities of FMI version two [26]. This indicates that tool vendors are taking this standard seriously and are investing in making their tools compliant.

FMI is structured with two purposes in mind, model exchange and co-simulation. In this context model exchange is considered to be the generation of code in one tool and executing it in another. Whereas co-simulation is when two or more simulations are run with their own independent mathematical solvers, but share data across communication ports during execution, it is worth noting that FMI does not specify any algorithms to orchestrate the co-simulation execution, this is for the user to specify. It is also intended that all post processing is conducted in the individual component tools. The way that FMI suggests managing the variation between specific tools is to use the standardised 'C' language functions to form a wrapper. The C language was chosen as they consider it to be the most portable across all platforms. An XML file accompanies the C file and describes the interfaces and accessible variables. Having such a machine readable file allows for different tools to handle the data structures as they wish, enabling the use of multiple tools.

The FMI standard uses the concept of FMUs (Functional Mock-up Unit). These are functioning parts of models and simulations that comply with the FMI standard that may or may not need an external mathematical solver to operate. The purpose of FMUs is to have component parts that will interface without implementation issues. The C wrapper includes not just the interfaces to the model but also any required run-time libraries used in the model as well as any

binaries for Windows or Linux machines. It is intended that the exact tool and even operating system that is used to develop the FMU should not affect the interface. This is an attempt to take a significant element of variation out of the integration. This communication is also intended to extend to ensure that the processor that is used to compute the simulation. FMUs can be generated manually or automatically using specific modelling environments. Not all tools which advertise that they have FMI capabilities are fully complicit which can be seen from the FMI organisations own testing [26]. Anecdotally some practitioners say that tool compliance is currently the greatest challenge of using FMI in practice.

A key area of the FMI standard that is of interest to this research is the information that is captured about the FMUs in regard to the model description. It is a detailed description that covers not only the interfaces but also other aspects that are required for successful integration.

The means by which the units of the data which are to be communicated are defined is of specific interest. The units are defined based on a unique name, how they relate to SI units. There is also functionality for the potential for use of conversion factors and offsets. FMI however only allows a set of standard data types available in a defined set of C to be used to represent this information. This is an instance of a standard being both strict and still having flexibility in a specific area. Along with each value available not only is the unit documented but also the computation type that is used to capture it in the simulation within FMI. This in essence allows for all values of any simulation to be identified and quantified. The defined subset of C is the only language available due to the C wrappers that are used for the FMUs. Aspects of the model structure are captured including all outputs of the model, a list of all exposed state derivatives, and all initial unknowns at the start of the simulation. The standard identifies when integrating FMUs can be useful to have a default experiment set up for all of the values available at the interface, as well as: start time, stop time, tolerances, and preferred communication step size. If any of these properties are outside the boundaries which the designer specifies, the validity of the FMU's output could be brought into question. There is also a place for the capture of the tool that was used to create the FMU as well as vendor annotations. It is worth nothing that not all of the FMI model descriptions are mandatory and many of the values are indeed optional.

Model exchange using FMI is enabled by specifying models that use differential algebraic discrete time equations. It is intended that FMI for model exchange can be used to port models between different modelling environments as well as to embedded control systems. Using FMI for model exchange is intended to capture models that consist of ordinary differential equations, or discrete time equations. This is only a small subset of all of the possible ways in which phenomena can be represented, hence this is only suitable for a constrained set of problems as part of the FMI standard documentation. The issues of algebraic

loops are discussed, this is where there can be dependences between different component parts of simulations that rely on values from another simulation component. It can become a self-referential loop where there is no clear starting point. Components being dependent on values from other components can be a powerful way of calculating complex behaviours, however there are additional complications such as algebraic loops.

For co-simulation, FMI is intended to aid in the coupling of two or more simulations. FMI utilises the concept of master and slave for orchestrating execution of the simulation. FMI defines the interface routines for the communication between the component parts. By specifying the communication combined with the way in which FMI for co-simulation is structured enables the user to conduct co-simulation within a single execution environment, between two or more instances and even distributed across a network. FMI for co-simulation relies heavily on complicit development and execution environments. By taking this approach it is possible to take advantage of the powerful simulation environments that are available for domain specific situations. This allows engineers to capitalise on the properties of co-simulation without compromising on the mathematical solvers that are tailored to the specific phenomenon that they are investigating.

As a technology for the integration of models and simulations FMI shows real promise and has demonstrated there is a real commercial demand for tools to seamlessly integrate. However due to the nature of the way in which it has to be implemented its success is in the hands of the tool vendors. If the tool vendors implement the interfaces and auto code generation then it has the potential to become a viable tool for large complex projects. However at present it only works with a small subset of the overall potential methods for the modelling and simulation of phenomena and even then there are issues that need to be overcome. The models that are to be integrated using FMI also have to abide to the standard and dependent on the individual model as with making any model comply with a standard re-work may be required.

BS ISO/ICE/IEEE 29148:2011 Systems and software engineering – Life Cycle Processes – Requirements Engineering

This standard specifies processes to guide the engineering of requirements for systems through the system lifecycle. It brings together the standards ISO /IE 12207:2008 and ISO/IEC 15255-2008 and gives guidance on how requirements are to be implemented. The standard is intended to be a general purpose guide used for physical and virtual systems as well as services. It is also intended to scale with the project and be capable of being equally of use for large complex systems as it is for smaller ones.

The standard takes the user from the recognition of there being a need for a system through, elicitation, capture, what requirements should contain, and on to how to use them for the verification and validation task.

The standard specifies the forms in which a textual requirement is to take. If adhered to it reduces the variation and has the potential to simplify the requirements statement structure and thus reduce ambiguity. The syntax of textual requirements is specified in Figure 2.3 below.

[Condition][Subject][Action][Object][Constraint]

Example: when signal x is received **[Condition]**, the system **[Subject]** shall set **[Action]** the signal x received bit **[Object]** within 2 seconds **[Constraint]**.

Or

[Condition][Action or Constraint][Value]

Example: At sea state 1 **[Condition]**, the Radar System shall detect targets at ranges out to **[Action or Constraint]** 100 nautical miles **[Value]**.

Or

[Subject][Action][Value]

Example: The invoice System **[Subject]**, shall display pending customer invoices **[Action]** in ascending order **[Value]** in which invoices are to be paid.

Figure 2.3 The ISO/IEC/IEEE example of required syntax of textual requirements[27]

When considering the automation of requirements manipulation and analysis having such a strict standard significantly reduces the complexity of the task.

Of particular interest to this research is the information that it specifies as necessary for functions or programs that are intended to be a component part in a larger system. The following topics are identified: communication interface, memory constraints, operations, site adaption requirements, product functions, limitations, assumptions and dependences, external interfaces, functions, performance requirements, and software system attributes. This covers the basis of what is needed for a software component part of a larger system, however it is not intended specifically for modelling and simulation systems and so does not have the specifics that need to be considered for use in that domain. A relevant and significant part of the standard is that these requirements are intended for new systems that are being designed and not necessarily for attempting to use existing component parts. There is the train of thought that there is no difference between the use of existing and new components to satisfy requirements, however this would depend on how the requirements are to be used in the project. If the requirements are to be used for selection purposes then there may need to be further specification than this standard details.

The application of this standard in the problem space, as defined in section 1.3, could be of some use as it would ensure homogeneity across all requirements that are used. However this standard is only concerned with the structure and content of requirements and does not give a usable structure for validation or verification testing.

THE APPLICATION OF STANDARDS IN PRACTICE

There is an issue with the use of standards that is not frequently discussed in the academic literature but does appear more frequently in less formal publications: it is whether the use of standards is purposeful in practice. It is clear there are

standards for almost all engineering processes. However how many are actually adhered to in practice is a question that is difficult to answer. There are very few articles that have anything negative to say about standards and their use in engineering. However there is a view that proposes that standards alone are not enough to guarantee end-to-end interoperability [28]. Here they step through many reasons why standards in their current form are insufficient, such as vendors adding extensions to maintain competitive advantage, standards not fitting with organisational structure, inappropriate application of standards, conflicting standards (if using more than one), over flexible and unwieldy, inflexible to meet situation. This indicates that the application of a standard is not always straightforward as it first seems and the motives of all concerned with the standards used may not be allied with the project that it is being applied to.

On a non-peer reviewed blog, one author refers to coding standards as unicorns [29] as they are mythical beings that people have heard of but never seen in reality. This article has been cited in many times as being a realistic representation of writing computer code on a large project. Such comments would not make it into a high quality publication, however this does not lessen the validity of the observation, and due to its un-sanitised representation may reflect the situation that many practitioners face but do not feel that they can report or publish. Such publications are indications that standards may not be as easy to implement in practice as the main stream literature reports. This is an example of where main stream literature may be disconnected from real-world practice.

2.4 Implementing Integration of Model and Simulations

The desire to get component parts of a software-based system to be integrated is not novel, and in the literature there are numerous ways in which this has been achieved. However many of the methods were not specifically designed with the application of model and simulation integration in mind. This has resulted in a situation where research has been conducted to see what is the most effective means of implementing model and simulation integration. There is a distinction between the approach that is taken to integration as discussed in section 2.3 and this section which is concerned with how the method is implemented.

A consideration that has to be taken into account when assessing integration implementation technologies is their suitability to operate over a distributed network. Having the ability to distribute the computation of the component simulation parts allows for simulations that would otherwise require the use of a high performance computer; lower cost; or dummy components (see [30] for more information on running dummy components during simulations),HIL or HITL experiments. For this problem space five technologies have been identified from the literature and show varying amounts of promise for this problem space. All have been documented as being used for the purpose of model and simulation integration. A specific HLA method DDS has been developed further than the basic concept of HLA and is discussed separately from the core HLA concept. Hence five technologies and the six methods have been evaluated:

- Data sharing
- Variable sharing
- Middleware
- Federated simulations
- High-level architecture
- Data distributed Service

2.4.1 Data Sharing

The simplest form of integration implementation is the sharing of data between two or more models or simulations. This often takes the form of the output of one model being the input to another. This can be done as a continuous or post-process mechanism. A representation of this form of integration can be seen in Figure 2.4. This means that integration can become complex and difficult to follow especially if the two models are sharing data in a bi-directional form.

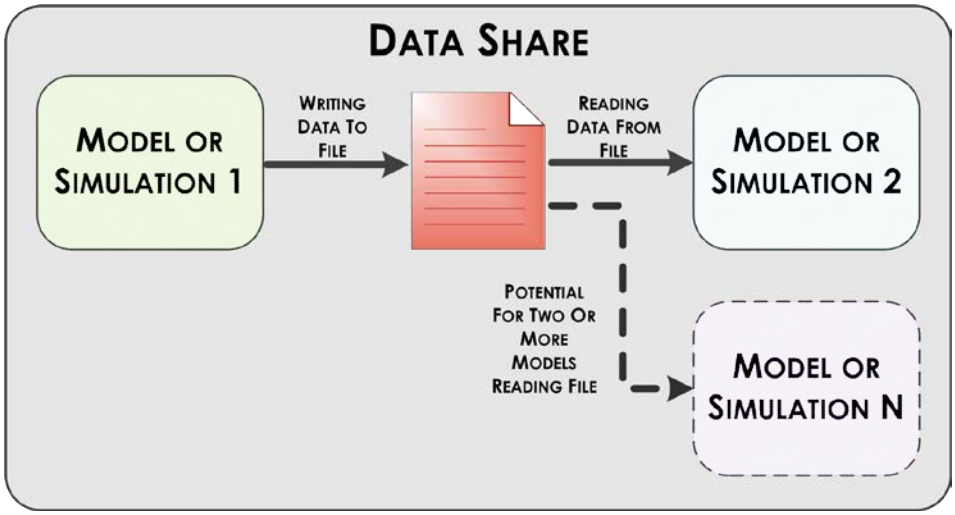

Figure 2.4 Data share integration between two or more models and simulations. The dotted line denotes the potential for further models to be integrated.

Data sharing requires both models to use the same data type, and for semantic information to be the same in both models. This requires a detailed architecture and requirements that specify each information exchange. A race condition is a situation where code can produce different results if it is run multiple times. Race conditions are caused by non-deterministic code executing in different orders. If race conditions in simulations are noticed it results in a lack of confidence of any results generated by it. Often a strict standard is decided upon and implemented before the models are created to ensure that all of the component parts produce and have access to the data in the requirements. This integration method is at a very low level of abstraction and hence requires a high level of understanding of exactly how the component parts interact. As this solution does not have any oversight to orchestrate all of the individual component parts, race conditions are a real concern. The capabilities that this integration gives a simulation are limited and require a linear progression of calculation. Control loops and other feedback mechanisms between or across component parts are difficult to successfully implement using this method.

This method is timely to design and implement when the number of models to be integrated is low. The time overhead grows with each extra model that is implemented. As the interface and the data transferred are different in each case the complexity also grows with the variance of components. The specific means by which the data are transported from one model to the next can be fulfilled by using a variety of existing technologies, be they the simple reading and writing the data into a directory, to a complex networking solution. The specific network implementations are outside the consideration of this research.

The resulting integrated solutions which use this method tend to be very brittle and any change to any of the component parts results in considerable re-work. There is also the potential for a loss of confidence in the simulation as a whole due to the potential of unknown interactions.

2.4.2 VARIABLE SHARING

Models and simulations often have internal values that change during runtime. Often these internal values are referred to as variables. There are occasions where these values may represent semantically identical variables across multiple models or simulations. In such cases it can be hugely powerful if the models and simulations share their values during runtime. The information exchange can be seen in Figure 2.5 below.

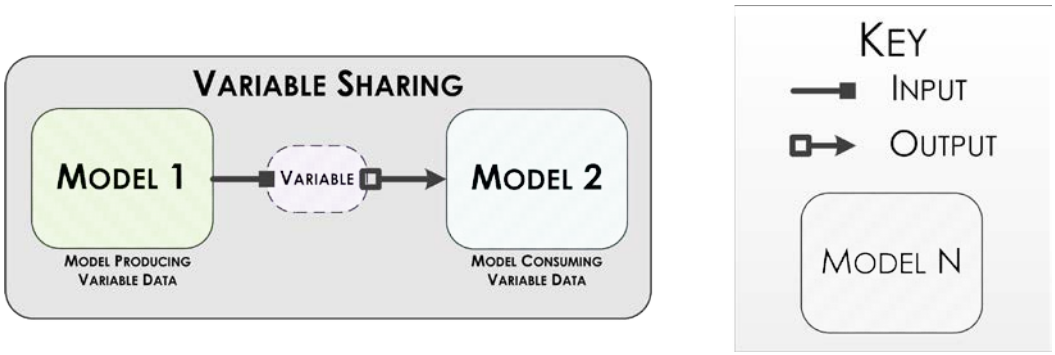

Figure 2.5 Integration using the sharing of variables. Model 1 is producing a variable and Model 2 is consuming the data.

For this type of integration to be possible the run time environment that is being used has to allow access to variables during execution. At present not all COTS modelling and simulation products have this capability. Setting up an environment where values can be manipulated is not a trivial requirement as it requires the model to continue to execute despite one or more of its values changing while the program is using it, as well as handling when a model is to update its own values and the related complications. These issues can be further problematic if the model is highly recursive in its architecture. Breaking into nested loops mid-iteration can cause inconsistences or errors within calculations.

To avoid conflicts between models there are recommendations such as those found in FMI [22] whereby a master-slave configuration is utilised. A master-slave configuration is where one model is the master it and it dictates to the slave what specific values are. This concept can go further to timing orchestration, initiation, and termination of the simulation. When sharing values there are also issues of where data to be communicated is held in memory. Often such memory issues are overcome by having a table of data that can be read or written to by the component models and simulations. In such cases the table is often external from the inner workings of any of the components. The table is written to (updated) and read from as and when required by the components. This method of having an external data table has been shown to work not only on a single machine but also across a distributed network. However to implement this method a means of defining which model is to be read and which is to write at any given circumstance is required, as otherwise conflicts can arise causing issues with simulation validation.

This method of sharing variables can be used for either of the approaches discussed in section 2.3.1 It is often the case that the master-slave paradigm fits very well with one environment calling the other, the caller being the master and the called being the slave.

If there is a desire to use a distributed solution where parts of the simulation are being run on different computers, a network layer is needed. The network layer handles which data are going where and when, it also has to handle what is needed to facilitate when each component is to start its next cycle of evaluation. All of this communication needs to be moderated by a communication layer to ensure the quality of service. The communication network layer has its challenges. However network communication is a well-established research field with many mature technologies that have capabilities that can facilitate this data transfer.

To implement an integrated situation using variable sharing requires the architecture of the overall simulation to be specified before any integration can occur. This is due to the level of knowledge that is needed about what information needs to be accessible and where it is going, in advance of the decisions as to which simulation is in control of writing which variables at which point.

This form of integration has shown to be useful for instances when there are only a limited number of component parts. There are commercial tools such as Flowmaster [13] where this type of integration has proved to be highly successful. This method operates on a single machine and does not offer much of an enterprise challenge to an organisation as long as the engineers involved have intimate knowledge of the component parts. However once this method is expanded to be used for many models and simulations distributed over a network the overheads rapidly increase. For large distributed simulations a well-defined architecture can aid with what needs to be communicated where and when. A strict standard can ensure that the interfaces are in a state which keep the variation to a minimum. If any changes are made to the component parts they can only be internal and minimal to ensure that the variables that are being communicated are no compromised.

2.4.3 MIDDLEWARE

The concept of middleware is used profusely within the modelling and simulation integration literature [31]–[44]. What is referred to as middleware can vary from a pre-defined transformation of a number from an eight bit integer to a 24 point floating point number, to a system that automatically identifies the available data from a set of available models and its type with the required data needed by other models, assemble the transformations and handle the network communications. A general representation of middleware can be seen in Figure 2.1. Hence middleware is considered to be anything that sits between two or more component models or simulations. Due to the variation as to what middleware actually refers to it will not be discussed any further.

2.4.4 FEDERATED SIMULATIONS

To enable a fully distributed simulation to be formulated and executed there are those who have taken the federated systems technology and applied it to modelling and simulation. This technology was developed within computer science to enable sections of code to be distributed over a network that all work together for a common goal. The overall code is broken down into smaller component parts or federates. Each federate is available over a network and discoverable via use of appropriate program code. The intent is for a mechanism to select which federates are needed, and then handle the communication between them. In the case of simulation if it can be broken down into constituent parts where a simple script may be sufficient to make the selection [20]. Figure 2.6 below shows a simplistic representation of a federated system.

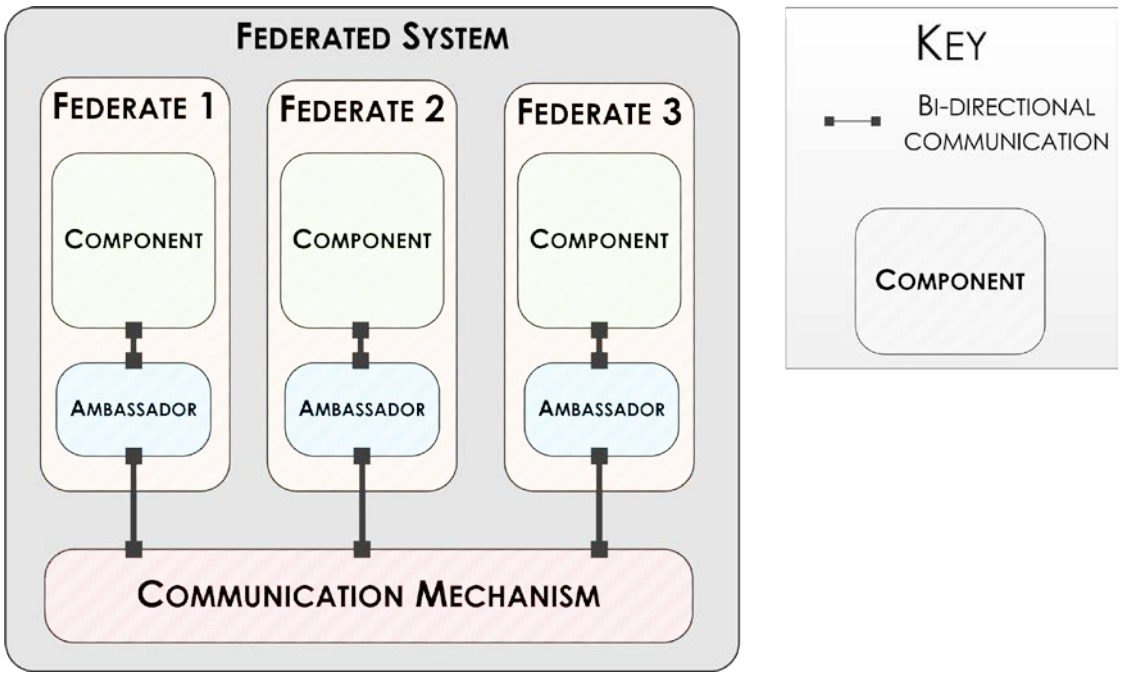

Figure 2.6 Simple representation of federated system

Often federated systems use web based technology and use a service registry, a broker, a service requester, and a service provide [39]. The service registry stores the location and content of all of the available services on the network. A broker handles which components can be connected and the means by which the communications can be made. A service requester requires a particular set of data to operate, whereas the provider produces a set of data. Federate technology can be of use to the integration challenge with each modelling component being its own federate. The communication layer between them acts as the integration implementation. This approach is represented in Figure 2.7 below.

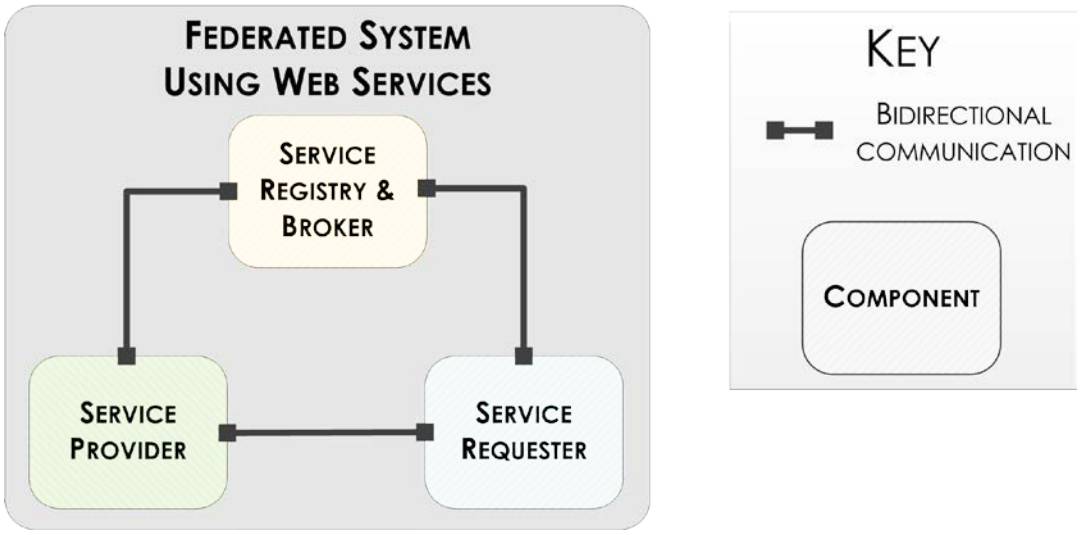

Figure 2.7 Representation of web services often used as part of federated systems.

At present there is an ever growing body of research as to the best way to implement the concept of federated systems. This research however is not limited to the implementation of modelling and simulation. Currently the majority of the research focuses around the network layer, broker (decisions as to which service to use), middleware, and schemas to capture the required information. The majority of federated systems use some variation on the 'publish and subscribe' paradigm. This is where publishing components make information available over a network and subscribing components read the published data from the network. The concept of the publish-subscribe paradigm is well established and hence has been shown to be useful e.g. [45], [46]. Current federated systems often use standards such as DDS and HLA to ensure that the federates can effectively communicate with each other [47]. There are few sources that demonstrate working instances where more than tens of publishers and tens subscriber federates are communicating over a network at any one time. This may be a limitation of the technology (this is known regarding DDS) or it may not be reported. Whatever the reason for the small scale demonstrating for a full system integrated test to be possible, many hundreds of models over many hundreds of machines all work in unison without fault would be needed. Such issues highlight the differences between looking at the theoretical science of the technology and the practical engineering application of such a technology. However this technology demonstrates potential to make full unit testing possible in the future.

Using a federated approach to produce an integrated system allows for teams to work in isolation and at a given time integrate their work together. It has been identified in the literature that just because the federates can communicate may not give meaningful outputs when connected [47]. This is a prime location for systems engineering to aid in the support of such a task, as the principles of architecture design and V&V could give confidence to the validity of the overall integrated simulation.

2.4.5 HIGH LEVEL ARCHITECTURE

The concept of High Level Architecture (HLA) with regards to simulation is one of general purpose architecture for the integration of distributed simulations. There have been various architectures that have been proposed as a meta solution to federated simulation. However one dominates the domain landscape: HLA was developed by the U.S. Defense Modelling and Simulation Office (DMSO) to provide simulation interoperability and re-usability across all types of simulations and has been adopted as an IEEE standard.[48]. This method means models and simulations can be integrated and has been heralded with many statements such as

> "A promising approach to building and evolving large scale distributed simulation" [37].

The means by which the HLA is able to bring models together is by specifying the boundaries and formalising the means of communication.

A visual representation of an instantiation of the HLA architecture can be seen in Figure 2.8. This shows how a federated system is created using individual simulations that are formed into federates that operates as one of many communicating over a runtime infrastructure. The two ambassadors in this example handle the calls to and from the run time infrastructure (RTI). To enable the models to have a standard interface allowing for communication with the RTI, wrappers are used. To allow for external reference to a library to be used, references are stored in a linked list. In the case of the federation that is shown in Figure 2.8 below, C has been used to wrap the external references. C is often used as it is considered by many to be the most portable language between operating systems and environments.

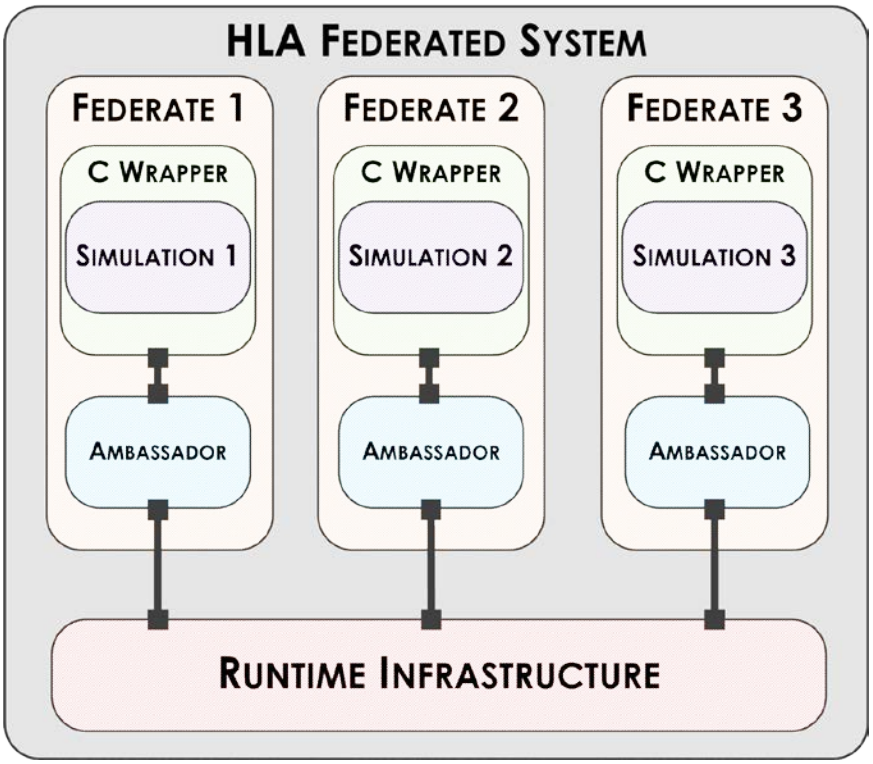

Figure 2.8 The organisation of a HLA agent note the simulations being only a part of the federate

The Operation of HLA

The underlying principle of HLA is to specify the format the simulations have to adhere to. This specification takes the form of rules and templates. HLA specifies rules which are intended to ensure a particular means of interactions between the component simulations of the federation, as well as describing the federates responsibilities. Part of HLA is the object model template which is a formal means of specifying the specific simulation data used. The specification is in the form of a hierarchy of object class, attributes, and interactions [37]. HLA in essence identifies how federates are to interact with the federation as a whole as well as each other.

The Benefits of Using HLA

The key benefit in using HLA is the potential for simulation re-use and the ability to connect the simulations in different combinations [48]. This is desirable when engineers are confronted with changing requirements and if the solution they are working on is using a platform-centred approach. A possible benefit of using HLA is using the descriptions to allow for the properties of the federate to be analysed and the information used to optimise which host to run which federate. These benefits have led to bold statements being made in the literature such as

> *"HLA initiative has become the de-facto standard technical architecture for military simulations."* [34].

Interestingly within the literature there are fewer non-military examples of the use of HLA. Not all are so convinced and as with all technologies constraints have been identified.

Constraints of the HLA Architecture

The critiques of HLA state problems of scalability, as well as the applicability of using this technology in a brown field environment. The problems with scalability issues are formed from one of the very founding blocks of HLA. The way in which communication between the system nodes are orchestrated is based on each federate broadcasting each update message to all other federates in the simulation [45]. This is not an issue if the network has capacity to take the quantity of messages that are pushed across it. However as the number of federates increases or if the size of the messages grow, the network can become saturated resulting in the quality of service between the federates to become far from guaranteed. This raise serious questions about its appropriateness to be used in this problem space as for a full system simulation it may require many hundreds of models and simulations communicating with each other.

These concerns are confounded when considering HLAs suitability for application where the models are created without the intention of complying with the HLA format. Those who question HLA in the literature make comments such as

> *"HLA forces developers to provide a particular functionality or to conform to specific standards in order to participate in the integration process; the rigid assumptions and limitations on participants makes it difficult to integrate pre-existing simulations without significant modification (especially in non-military domains)"*[34].

From the problem identified in section 1.3 the models and simulations are existing and not necessarily from the defence domain. Make existing models comply with HLA and be made into a federate could depending on the model or simulation in question take a considerable amount of re-work.

2.4.6 DATA DISTRIBUTED SERVICE

DDS (Data Distributed Service) is a specific publish-subscribe architecture that aims to improve on HLA and allow for one specification to cover many different model intercommunications. This has been further elaborated in the literature as *"DDS distributes data where you want it, when you want it."*. The publish-subscribe paradigm handles the *where* as it allows for communication across a distributed network while the data centric enables the *when*.[49]. This is an attempt to make a deterministic publish-subscribe communication possible.

Definition of DDS

DDS is an OMG standard concerned with the transfer of data between multiple points involving different hardware and software [33], [37], [49]–[51]. The means by which DDS is capable of doing this is by using a specification that identifies the type of data being published and the requested subscriptions. The specification has a platform independent model that can be mapped to various project specific models using a variety of programming languages.[51]. By platform independent what is meant is a concept rather than a specific instantiation of a

concept. The model can be implemented in many languages even to satisfy the requirements for the same system. The OMG DDS specification has two levels of integration, the data level or Data-Centric Publish-Subscribe (DCPS) [52], and a higher level aimed at the application level the *"Data-Local Reconstruction Layer(DLRL) level"*[52]. For the purpose of this review, this body of work will focus on the DCPS level. For reference, the structure of the overall DCPS model and the significance of the data communicated as well as the importance of the data type that is used, refer to the OMG standard [52].

To achieve real time capabilities, the DDS specification has a specific set of profiles that target real time communications [37]. They focus on the data, its form, and the deterministic nature of the data delivery which is stated using a policy of QoS [49]. The defining of topics allow for the harmonisation of what is being written and what is being read. The topics are defined from the type of data which needs to be transferred, a routing label, which is combined with the quality of service that enables not only effective communication but also the potential for implementation optimisations [52]. This feature is considered useful when dealing with hardware and software that differ from one another. When a system is operational the topics also allow for searches to be carried out for what data are available on the network and in what form. This capability allows for reduced time of implementing a variation of a simulation or of a brand new simulation made of existing compliant components.

DDS in Use

Some consider that DDS can be more flexible than a strict HLA implementation. This is due to DDS having a rich set of QoS policies while leaving out some aspects in HLA such as time management and federation management[38], [52]. This is due to DDS being much more focused on the distribution of data at the application level rather than the architecture of distributed simulations at many levels of integration. This gives the system designer a greater flexibility in the solutions that they propose, however this comes with a caveat as it means that system architecture will need to be designed for each new system rather than relying on a pre-determined existing architecture, meaning future re-use may become an issue.

It is possible using DDS to set up communications over networks using existing standard communication methods in both hardware and software for example Ethernet, Shared memory, TCP/IP, HTTP, and others [49]. Meaning that a DDS solution can be implemented alongside or as part of an existing hardware and or software solution. A feature of DDS that is compelling to many is that

> *"multiple independent communications networks (Domains) each using DDS can be used to over the same network transport protocol."* [37].

Such a feature further endears it to use by teams of engineers working on a distributed project working on the same network. The implication of this for an

organisation is that many simulations could be run over the same network potentially sharing models or simulations during execution.

There are many instances where DDS has been shown to work in many real time applications including: safety critical systems, various defence projects, as well as conferencing applications [37]. This demonstrated success indicates that this technology can work at multiple scales and for very different system level application requirements.

Challenges With Using DDS

Despite DDS being a relatively new architecture it is not without criticism. Problems associated with its scalability, time management and parallel execution on the same host, have been reported as indicated below.

Issues of scalability focus around the way in which one writer can supply to many readers. This is due to DDS using IP multicast, which has known issues with scalability [10]. However the multicast issue is not the only way DDS can be implemented as

> *"Using custom meta information DDS can be bound to any transport protocol."*[46].

Hence current research into DDS has shown promising signs using technologies other than IP Multicast.

As previously mentioned the DDS specification is not specifically designed to encapsulate all simulation semantics. This allows for greater flexibility of implementation however it will require greater effort from the simulation designer as they will have to ensure all communication is meaningful. The simulation designers will inevitably have to rely on the QoS policies (for communication) and the specification of the federates (for semantics). Hence the role of simulation architecture is of importance.

There is an issue of using DDS for access to data through a middleware thread, as it may be more complex to pass the data between multiple threads on the same host [52]. This could result in greater process overheads and potential issues with parallel processing. This problem would affect simulations on case by case bases as it not only dependent on the environment they are written in but also the way in which they have been written inside said environments.

2.4.7 Evaluation of the Implementation of Integrated Models and Simulations

The current methods which have been identified all facilitate the communication between models and or simulations, however they all have their drawbacks and criticisms. At present none of the methods provide an adequate solution to all of the issues that are present in the identified problem space.

Data and variable sharing produces integrations that are brittle and vulnerable to any engineering changes that may occur, be these requirements or implementation based. These two methods are well established and there are tools on the market which capitalise on them. The methods themselves are adequate for model and simulation integration however they require a high level of understanding of the component parts to make them work.

Middleware with its almost infinite flexibility has the potential to enable almost all models and simulations to communicate, but this flexibility comes with the potential of disorganisation, and minimal possible re-use of any parts of a solution. If an organisation wishes to use middleware and simulation re-use, standards have been shown to be of use. However this approach is best suited for *de novo* projects. Hence if an organisation is wishing to get value from existing models, this approach may require considerable re-work.

High level architectures which are designed with the explicit intention to aid in the integration of models and simulations have been shown to be of great value to defence projects. There is also potential for such architectures to be implemented in non-military projects. These architectures have been shown to work well with projects that are creating the component parts from scratch rather than re-use. Such architectures aid in what needs to be communicated and guides the user however they still require another method to pass the data between the component parts.

The publish-subscribe technology has shown that it is possible to create dynamic distributed simulations that will execute over a network. The best current architecture to use is very much dependent on the specifics of the situation where it is to be implemented. From the literature, case studies of HLA and DDS (despite the concerns that have been raised) has increased interest in the idea of distributed simulation systems. These distributed systems could potentially be used for large sale distributed models and simulation execution.

Distributed simulations running across a network is a challenging, target-rich environment. Research is being invested by both academic and industrial organisations, in this topic.

For the purpose of integrating models in this problem space none of the methods do more than allow for models and simulations to pass data between each other [15]. This means that none have any means to ascertain if the integration is

meaningful. This puts a high reliance on the individual using the system to understand what the models are doing (white box understanding). In this problem space it is unlikely that any individual will be able to have such a complete understanding. The network layer is possible in principle; however there is still room to improve.

The integration methods which support the integration approaches are at a point where the transmission of data between two or more component parts is possible. There are ways that have been successfully demonstrated in the literature that use different numbers of component parts ranging from less than ten to over fifty. The various methods are suited to the various numbers of constituent parts. At present there is no one singular method that is suitable for all model types or simulation component parts. This is not a significant issue as appropriate methods can be selected dependent on the project in question. However producing a single product that is suitable for both small and large numbers of models would require multiple options for implementation.

2.5 THE EFFECTS ON INTEGRATION FROM TYPES OF MODELLING AND SIMULATION

Within science and engineering there are many different types of models and simulations. They affect the choice of integration techniques which can be used, or in some cases make integration impossible. Each of the major modelling and simulation types will be investigated below and include:

- Linear methods
- Reduced order models (ROMs)
- Logic-based simulations
- Feedback loops
- Statistical methods
- Artificial Neural networks (ANN)
- Computational Fluid Dynamics (CFD)
- Finite element analysis (FEA)

2.5.1 LINEAR METHODS

There have been many mathematical methods developed for the purpose of tackling the problem of analysing non-linear data. Many of these methods take the data and through various manipulations produce linear equations that represent sections of the non-linear data. There are many linear methods developed in mathematics and now with the increased computational power some of the previously disregarded methods (due to computational overhead) are being used in engineering [53]. It is now possible for the simplest local analysis of varying one factor of a simulation at a time to find a useable result can be used [53] however this can be a time consuming task. There are too many linear methods to discuss each one and the specific implications to integration. There are many books and papers available on the application of linear methods. A review of which would consist of a significant task in itself and falls outside the bounds of this research. However there are some generalities that can be identified from the literature. Linear methods are only valid over a specific range. If values are passed to them which are outside the effective range then the validity of the results cannot be assured. These methods are often fitted to existing data sets, so a detailed understanding of any candidate data set is required.

2.5.2 REDUCED ORDER MODELS (ROMS)

Within science but more so engineering there is a current trend to characterise complex systems not with a series of high fidelity models but rather with a higher level of abstraction that expresses a series of complex interactions with lower order polynomial equations. This process takes the behaviours of many aspects of the system and either by means of statistical analysis, acquired measurements, combined outputs from simulations, or other means, the behaviour of the system is characterised. Reduced ordered models are by their very nature

computationally simpler to implement than a high fidelity model concerning the same system. The execution resources necessary can be significantly less. ROMs can be formulated using one tool, the equations captured, and the equations implemented in another tool [54]. The power of ROMs has been demonstrated in many instances. However there are criticisms in using ROMs as a method, specifically when attempts are made to integrate them as a component part in a larger simulation.

The suitability of ROMs for whole unit or system wide test may not necessarily be the most appropriate type of modelling. The suitability of the use of ROMs depends on what the scientists or engineers are looking to find from the test. Often ROMS are created by taking a series of complex behaviours and fitting a low order equation to the resulting trend. This reduced order model is then only going to give expected behaviours based on established knowledge. Questions have been raised as to if this approach will aid in the identification and understanding of unknown interactions be that either their cause or effect. For whole unit tests a ROM can give the broad behaviour of the system and this should not be undermined as it is a useful tool in an engineer's tool set. There are occasions, such as early in the design processes of an incremental change in a product, when such a simulation can be valuable.

However the effective ranges that the parameters of the ROM can take are often limited. Any testing that is conducted with any of the variables outside of their intended range (which would have been built into the model), leads to the accuracy and even the validity becoming suspect. Dependent on the structure of the ROM it may also be difficult to ascertain which factors relate to which parameters. This is because often multiple factors are represented in a single variable. In the case where a modelling change is required the ROM may have to be completely re-designed or a simple value change of one of the parameters. Whichever is necessary the process of changing the ROM requires intimate knowledge of the ROM in question. The architecture of most ROMS is such that any change in the requirements of the ROM will likely requires a considerable amount of re-work to make it compliant.

Integrating ROMs with other ROMs or with other forms of models of differing levels of fidelity produces a challenge in itself. As the variables within a ROM may not relate to a single physical characteristic but rather a combined behaviour the integration cannot just be at the boundary of the model but will have to go deeper. The full models semantics have to be taken into consideration. To understand the full semantics of the model will require a deep and full understanding of the ROM, which usually comes from developing it in the first place, and can be difficult to capture all nuances in documentation. Hence questions as to the effectiveness and value of such an integration exercise are inevitable.

If a series of ROMS were produced with the intention that they were to be integrated and there was a usable separation of variables, then integration of variables would be possible. However this goes against the way in which ROMs are currently being advocated [55]. This is also not the state of the ROMS within the problem space as defined in section 1.2.3.

Integrating ROMS to work in a federated system in a traditional fashion (exposure of inputs, variables and outputs) may not be the way forward with this technology because of the above mentioned reasons of separation of variables. To be usable the testing procedure would need to be fully defined before the start of the procedure.

2.5.3 Logic Based Simulations

There are instances where all of the possible outcomes are needed to be accounted for. In such cases engineers often turn to formal logic-based systems. Logic-based simulations are less common than logic-based control systems, and these must not be confused. This work is concerned with the integration of models and simulations, not just control systems. To integrate a logical simulation the foundations and assumptions must be identical. If there any differences in the premises that the logic statements are based upon, the validity of any of the result will be undermined. Logic-based simulation was heralded in the mid-1980s as the future of simulation and there are sources that paint a hopeful future for formal-based logic often with Artificial Intelligence (AI) [56]. However there are far fewer instances and case studies in recent times. The instances of it occurring as a proposed method purely for a means of simulation have become far less common in the current literature.

2.5.4 Feedback Loops

It may be useful for simulation to have an element of memory from a previous iteration. These often feature in control systems. Having an element of memory can make integrating existing models or simulations a challenge with respect to handling initial conditions. There are also issues of parallelism if two or more loops are running in an integrated simulation. Within control systems there are established methods for integrating control loops which all involve a high level of understanding of the component models and simulations. Caution is to be taken when investigating the control literature as the integration of models in this domain can refer to the amalgamation of control strategies which is a far deeper integration that passing behavioural information between component simulation parts. An example of this is found in studies work regarding the integration of two controllers [57]. Each controller is not a simulation as it is the realisation of a control algorithm rather than a representation of a system or phenomena. In this example both controllers had feedback loops. To 'Integrate' these two controllers a rule-based method was developed. This works for handing as to when the various controllers are best suited to operate, however this form of integration is not helpful for models and simulations as it is unlikely that

the engineers are only looking for the output of one of the models or simulations during run time. This demonstrates that just because there are examples of where loops have been integrated does not necessary mean the methods are applicable in the modelling and simulation domain.

An interesting simulation integration implementation issue is that feedback loops often do not have a means of extracting these data which can mean that the only way of obtaining data is to wait until the loop has reached its termination condition.

2.5.5 Statistical Methods

Statistical methods are used for situations where the behaviour is not exactly known, such as elements of fluid flow, quality control, product life time, and vision systems. Often these simulations are concerned with the likeliness of an event occurring or not. In some cases, such as satellite development, there is a lack of empirical data and as such statistical models are used [58].

When statistical methods are present in one or more of the component parts of an integrated model or simulation, they can exhibit behaviour that is not found in components that use other methods. Statistical predictive methods may give different outputs if the same inputs are run multiple times. This gives rise to issues of verification and anticipating what is and what is not an expected output value. Some simulations have ranges for which the possible values could possibly take whereas others do not have this level of expectation.

2.5.6 Artificial Neural Networks

Perceptron – based Artificial Neural Networks (ANNs) were seen as a potential solution to many engineering problems. However their limitations have become apparent and now within engineering they are commonly used for optimisation and surrogate modelling. ANNs are mathematical models consisting of interconnected processing units known as neurons. Weights and biases are performed at each neuron via the use of a stated transfer function. The values of the weights and biases are defined by using training data sets [59]. The training data requires solutions to exist which the ANN is to mimic. For this reason ANNs have been used for optimisation tasks. The computational overhead for ANNs is predominantly used during the training stage. The issues of integrating ANNs either with other ANNs or other models in general is ensuring that the inputs to the trained ANN are within the bounds of the training data set. Keeping within the bounds of the training data is vital to maintain confidence in the output [59]. There are many instances within the literature surrounding ANNs where they have been used as substitutes or surrogates to more complex models. The ANNs are trained to mimic the outputs of the complex simulations with similar inputs. This allows for integrations to be run where if the original complex simulations were used the time expense would be significantly higher.

2.5.7 COMPUTATIONAL FLUID DYNAMICS (CFD)

There are many applications where there are fluids which are moving in, on, or around a system that is being designed. To simulate the effects of fluid flow, Computational Fluid Dynamics (CFD) methods are often used [60].

CFD uses numerical methods and algorithms to give a representation of the behaviour of a fluid flow of interest. Due to the very nature of fluid flows to achieve greater accuracy the computational burden is high. It is not uncommon for engineers to use high performance computing to run CFD simulations and they still end up taking many days to execute [61].

2.5.8 FINITE ELEMENT ANALYSIS (FEA)

This method involves breaking down the system under analysis into a number of discrete elements that are analysed individually. This method is often used for the investigation of stress and strain of physical elements under load. Tool vendors, such as Enginsoft, provide FEA case studies across many engineering disciplines [18]. This well established method has demonstrated results that have been validated against real world test data such as tread block forces for tyres in contact with road surfaces [62]. The resolution of the final result is dependent on the number of elements that the system is split into. Something that is repeatedly mentioned in the literature is the trade-off between accuracy of the solution, the number of elements, and the computational burden. With each element there is a finite amount of computation that will need to occur this takes time and uses resources. The computational burden of FEAs is the most significant challenge to their integration. The computational burden often results in these simulations taking a significant amount of time per experiment. It is not uncommon for an FEA simulation to take days to run.

2.6 SEMANTICS AND THEIR EFFECTS ON MODELLING AND SIMULATION

There are factors that make models or simulations identifiably different from one another. This is due to the nature of the phenomena being mimicked as well as the means by which the model or simulation was implemented. When it comes to integrating these existing models or simulations there is a striking difference between the academic and professional literature. Within the academic literature, individual parts of variation or difference are identified which are often the focus of the solution that is being proposed. In contrast the commercial literature often ignores any variation other than the implementation platforms used. Within this work both viewpoints are taken into consideration. However both sides do recognise that there are semantic issues that can and do affect the successfulness of model and simulation integration.

The semantics of a model are how the assumptions are implemented and the model or simulation is realised. This includes any and all decisions that are made such as but not limited to, mathematical representation, platform implementation, data types, and hardware that it is implemented. If there are differences between the semantics then they are often the causes of insurmountable integration challenges. Semantics are different from assumptions though they are often taken together and side lined as issues to be solved by those implementing the proposed methods. There are key areas of the semantic differences that are common throughout the literature, these include; application of standards, data types and forms, dependences, and variations in scales. Each of the identified areas is explored below.

Standards

Standards can go a long way to reduce the variation present within the selection of viable integration candidates. However due to flexibility of most modern modelling standards, the practical implementation results in situations where the semantic similarity between components is not guaranteed. Neema [63] recognises that even with the application of a standard such as FMI, the semantic issue is far from solved and still remains a significant challenge. For more information regarding standards see section 2.3.3. It is conceived that a standard could negate the issues of semantics, however such a standard would need to cover all aspects of the potential models and simulations including how they are to be implemented.

Data Types and Forms

Data and information can be represented in many different types and in various forms of each of those types. When a modeller is constructing a model, choices are made as to the form in which the information is represented, communicated, and stored. There are also a choices as to what form the information is to going to be produced and where appropriate consumed by the model or simulation. When a modeller is confronted with the challenge of representing the

information that requires manipulation there is a multitude of data representations each with their specific merits and weaknesses. With the various qualities of the data types means they hold very different specific information.

There are established methods for converting between data types. However issues arise regarding the information that is captured by specific data types. Different data types often capture different nuances of information. There may be information lost when converting between data types as the type that is converted may not be capable of capturing the exact same information in every situation. If a conversion is made there will be a loss of information. Hence data conversions between models are not a trivial matter and should not be undertaken without consideration as to what information will be lost once the conversion is made.

This issue with data types is not often covered in peer-reviewed journal articles as it is seen as an environmental issue rather than one worthy of being documented. However it is covered in many online help files such as Microsoft developer network with titles similar to Type Conversions and Type Safety (Modern C++) [64]. This is an example where the academic literature differs from the material aimed at practitioners. This indicates that some of the issues that an individual would come up against in replicating published work is not covered in the articles.

Dependencies

Once a model is implemented in an environment (*in silico* or otherwise) often it will require access to other hardware and software components which are considered to be model or simulation dependencies. The dependencies may not only be for the particular implementation environment but also for any external sources that it calls. The external calls can take the form of look up tables, DLLs (Dynamic Link Library), text documents or any other means by which information can be stored and recalled. These dependences will also have to be integrated or subsumed into any solution as without them the models or simulations may not function as intended, or even at all. Considering dependencies when integrating may not be as straightforward as it first seems. The dependencies may only work in specific environments, and the original information contained has to be conserved for repetition while there may be the need to manipulate the data, not to mention that the addresses of the dependences may have to be changed manually in the models when moving between computers. How to manipulate the dependences has many of the challenges that are present in the integration of models themselves. It was demonstrated in 2005 that an optimal selection of models including their dependencies was NP complete [15] and hence requires an inductive step by an engineer. This has interesting ramifications for future and the possibility for automated model and simulation re-use.

When models from different environments are brought together they will not only bring with them the dependencies they were written calling but also often a

dependency on the very environment they were written within. Some environments require that the whole run time engine of the development environment be installed on the machine that the model is to be run on, whereas others only require a subset of the environment present. Such dependencies on the runtime environment are discussed in the FMI standard [26]. The solution to this issue in FMI is to capture the dependencies and make them a part of the FMU. For more information regarding FMUs see section 2.3.3. Capturing the dependencies and keeping in mind the changes of versions of COTS software also adds to the difficulty of potential integration of existing models and simulations.

Assumptions are necessary for us to understand and represent the world around us. However these assumptions can cause issues when integrating two or more models or simulations. The variation in assumptions and the issues that it can cause is not a new problem [65]. Researchers in artificial intelligence recognised that selecting the most suitable modelling assumptions is an engineering challenge in its own right. They also highlight that with large complex projects that there is the need to decompose the solutions and by doing this there is likelihood of a variation of assumptions across the component models.

When consulting the literature regarding integration tools, very few actually tackle or even recognise the issue of verifying assumptions. One such example is the field of Service Orientated Architecture (SOA). Within the SOA research community there is the recognition that just handling the communications between the component parts does not ensure that the communication and hence the integration is meaningful [66] [39]. In both cases the differences in assumptions are heralded as a reason to meaningless integration. However that is as far as the recognition goes, and in such publications it is often stated that it is outside of the interests of SOA research to discuss methods of ensuring the aligning of assumptions.

In other fields such as model re-use [65] assumptions may differ to the extent of being contradictory which can result in any integration being of questionable validity. Hence assumptions must be explicitly stated as otherwise it can be impossible for anyone other than the modeller to know all of the assumptions that were encapsulated in the model or simulation. Some standards such as ISO 15288 System life cycle processes rev 2013 CD [67] recognise that assumptions can cause issues when analysing requirements. If there are issues with differences in assumptions at the requirements level it has the potential to cascade through the implementations resulting in outputs that are spurious at best. This indicates that the harmonisation of assumptions may need to go back as far as the requirements that are written for the creation of the component models or simulations to be integrated.

When considering how to refer to differing assumptions there have been those who try to categorise assumptions. Two categories of assumptions have been

proposed: simplifying and operating [65]. Simplifying assumptions contain the underlying approximations of the models and the modellers' perspective. Operating assumptions are made as to how the system being modelled is expected to function. This classification highlights the breadth of the effects of assumptions on the integration challenge. Assumptions go far further than simply what range the initial parameters take.

Often assumptions refer to models and simulations in their entirety, however this is not the only way of looking at the issue of model and simulation integration. The communication of the data is also affected by assumptions. ISO 15926-2 [68] details the importance of understanding the assumptions of the data which are being transferred. This is an interesting idea as it suggests that the data encapsulates the effects of any assumptions that were made during the creation of said data. In any potential solution this understanding, capture and analysis of assumptions needs to be taken into account.

2.6.1 MULTISCALE MODELLING

When a phenomenon is modelled scales are set, be they formally stated or otherwise. In traditional models and simulations the modeller often chooses to stay at one scale (spatial or temporal) throughout the representation. However there are those who now believe that this is no longer enough for the level of understanding that is needed, and there are statements in the literature such as

> *"Most physical phenomena of interest to humankind involve a range of temporal and spatial scales."*[69].

Such bold statements indicate the belief that multi-scale modelling has the potential to open up a wealth of previously unknown knowledge. In many engineering projects it has been identified that they have the potential to cross various scales. In engineering it is often the case where each model or simulation has its own modelling technique, assumptions, and critically scales. How to go about crossing these scales is a research field in its own right.

To highlight the issues of scale in an automotive example, consideration is given to a high performance luxury vehicle. The effects of the atomic properties of materials are known to have effects on the overall performance of the vehicle. For high power internal combustion engines there are significant issues of producing vibration especially during the starting sequence. Vibration of any form is far from desirable for many customers and so it is aimed to be eliminated. Often such engines are mounted to the chassis upon solid rubber blocks. Rubber is a glass like material whose physical characteristics change over time. The physical characteristic changes over time that the rubber goes through, causes it to behave differently under cyclical loading over time, even when the same forces are applied to it. This transition of characteristics is greatest over the first few months, but after moulding, a plateau is reached, and then it tails off years later. It would be desirable to know how to tune the engine at production so that

it can be calibrated to reduce the vibration and tuned to be at peak performance when the customer receives the vehicle not as it rolls off the production line. There are models that represent the changes in the mechanical properties of the rubber blocks over time at the micro scale. The engine is modelled at a higher scale of the vibration that it produces. The chassis is modelled at a yet higher scale in the way in which vibration is passed through it. Integrating these models together to enable the design and test of the engine mounts. Once the mounts are designed and engine mapping defined, testing can be conducted. During the starting procedure of the vehicle it is desirable for only a small amount of vibration through the chassis. At present the tuning of the engine is achieved as a manual tuning process conducted by a tuning expert by ear making the process more of an art than a science. The reason why this is still a manual process is due to the difficulties that crossing these scales pose. At present there are more variations in modelling techniques as there are means to cross scales between them.

The challenges of multiscale modelling have been discussed in a review [70] which details an automotive example when considering the mechanical properties of the car body as a whole. Consideration is given to the interactions at the atomic level of specific materials and the effects it they have on the whole. In this work six scales of modelling are identified and the challenges of crossing them discussed. The result of crossing the scales can be considered a type of model integration.

Challenges of Multiscale Modelling

When the models from different scales are brought together there are many issues that face the integrator. These include not only many of the issues already highlighted in this chapter but also specific challenges that come from crossing scales of modelling.

Within physics there are different equations that have been formulated to mimic the behaviour of different phenomena as viewed at a particular scale. This need for different equations at different levels all stems from the current understanding of the physics or the lack thereof. The need for different equations shows that there are differences in the fundamental behaviour of the equations and hence indicates the challenges of combining them. The issue of scale may not only be focused on the physical size (or scale) of the phenomena being modelled but also in abstraction, scale and others.

2.6.2 MULTISCALE INTEGRATION METHODS

There are many methods to cross modelling scales. However many of these methods are only suited to the problem for which they have been developed and are found to be less applicable in others. One of the reasons that is given as to why this is the case is making methods that are applicable in multiple situations is the fact that by the very nature of multiscale modelling it is interdisciplinary [70] [71]. Hence the methods that are created also need to be interdisciplinary. Some

methods attempt to cross the scales by combining different scaled algorithms in an attempt to cover overlapping domain which aides in the crossing of scale [70]. Such methods have proved to be a successful approach in limited demonstrated cases.

Often when engineers or scientists consider modelling, be that multiscale or otherwise these often start with the methods that they are used to using at a single scale. This is often differential equations as they are well established at representing rates of change. However the literature indicates that there may be more suitable methods to use. It is proposed that a broader view of mathematics is taken and methods such as Fourier analysis, matched asymtotics and multipole methods be investigated [72].

The subject of differing time steps has proved to be a common and continuing issue. It is often

> *"dictated by the dynamics at the smallest scale, which for atomic motion is on the order of* $\times 10^{-15}$ *seconds"*[69].

It is not feasible to integrate atomic simulations with those that use much larger time steps, such as 0.2 second *as* there are many orders of magnitude between them. There are methods to overcome such issues of timescale such as timescale extension [69].

Not all methods that have been proposed are based around the concept of linking equations. There are also methods that have been demonstrated that use statistical methods, such methods are referred to as equation-free multiscale methods. Fine, coarse, and special time scales are linked by using statistical means [69]. This indicates that there may be other domains of mathematics that could be of use that as of yet have not been fully investigated.

2.7 CURRENT METHODS TO OBTAIN A SHARED UNDERSTANDING OF MODELS AND SIMULATIONS

It may seem trivial to point out but in a modern engineering project there is the need for individuals to share their understanding regarding particular aspects of the overall project they are working on. This is often facilitated by meetings, phone calls, emails, or use of a proprietary project management tool intended for communication across a project. Many of the project management tools have been developed for the intended purpose of ensuring that information is not lost throughout the project in an attempt to ensure traceability of all communications and decisions that are made as part of the project. Regardless of the tool that is used be that an email client, or otherwise at present written text and the spoken word are the primary means by which engineers formerly communicate with each other, as well as with other stakeholders. Using English as the language for verbal and textual means of communication causes many issues which could be considered to be one of the root causes of many misunderstandings during the integration task. The study of linguistics is one that is well established and the English language has been the focus of much scrutiny. The issue of ambiguity within the English language is an issue that has been stated as strongly as

> *"The question of standard English is one of the great, unresolved problems that we are carrying into the twenty-first century"*[73].

This demonstrates that even in the specific field of linguistics there are challenges on the informal nature of language used and at present there is no uniformly agreed way to resolve it. One of the reasons for this informality is that English is a descriptive rather than prescriptive language [74]. Being descriptive means that the definition of words is fluid over time as well as its grammatical use and hence syntax. Thus, English is an ever changing language which impacts this work as it implies that each person will have their own individual idea of what the definition of a word is as well as the potential change in meaning which is only gathered from the grammatical syntax surrounding it. Therefore when one person (engineer or otherwise) attempts to share an understanding there will be an inevitable mismatch in definition which results in a situation which can become problematic. Now in everyday life when we talk to each other and attempt to convey a general idea, the inherent mismatch is not so much of a problem as the overall essence of what is being expressed is exchanged. However in a complex setting where shared understanding is vital, this dislocation in understanding can cause significant issues.

For the reasons of precision formal and semi-formal languages have been created, such as in order of increasing formality Unified Modelling Language (UML), Systems Modelling Language (SysML), and Alloy. UML and SysML are an attempt to fuse graphical representation with natural language to capture

requirements, architectures and designs, whereas Alloy is a completely formal mathematical language.

There are those who believe that the issue of ambiguity is such an issue that the only way in which we can communicate is formally with the use of mathematics. The result of this is a means in which mathematical descriptions and models are used to describe what is wanted from the system and the system itself. Such an approach has been shown to work with safety critical systems however it does have limitations as it requires that all concepts used in the project are capable of being described with mathematical concepts. For some concepts this may not be possible or time effective to do so. Despite the myriad of alternatives due to the flexibility and lack of training that is required natural language is still the prevailing method of communication and documentation that is used in industry. With this in mind it is clear why and how ambiguities can arise and cause issues in projects. This is a factor that will also affect the understanding of the textual descriptions of any models or simulations that are intended for re-use.

2.8 CURRENT TOOLS FOR CREATION OF MODELS AND SIMULATIONS

The review of software available for modelling, simulations and integration, contained in this work is a snap shot. The marketplace for such products is fiercely competitive with some software developers purposely obsoleting their own products to maintain their competitive position in this market [75]. Due to the transient nature of these products no single versions are identified however the general types and capabilities are discussed.

There are existing standard integration software packages for industry that attempt to provide vehicle manufacturers the means by which they can take their existing high fidelity models and integrate them together for whole vehicle tests. However, complex product manufacturers are having many challenges with such software packages. One significant business challenge is matching the organisations process with the process that is dictated by the tools. This can affect the order in which aspects of the project are developed and even the number of people who can work on a component part at any one time.

2.8.1 MATHEMATICS-BASED SOFTWARE

Mathematical simulations are used across all engineering disciplines due to the fact that mathematics is the most frequently used means to model physical phenomenon. Often this simulation software provides an environment for mathematical models to be constructed, executed, the result analysed, and the presentation of the findings be they graphical or otherwise. This suits the standard engineering design process of; understand requirements, design a system, create mathematical model of the system, test the mathematical representation of the system, analyse the results and iterate until favourable results are ascertained. There are many mathematical-based simulation tools available on the market. Some are more effective than others at specific mathematical operations. The way in which they are programed and set up is are often very similar however the specific operation is dependent on the package. This creates the situation where practitioners often gain a preference over time for a specific simulation tool or provider. The means by which the majority of the mathematical simulation software handles the challenge of integration is to either use the Application Programming Interface (API) or inbuilt code integration means. Some such software has the facility to input sections of code from different programming language and interact with it through a defined means.

2.8.2 General Purpose Co-Simulation Software

As the potential of co-simulation is being recognised by industry, the drive for software to support it has been produced to meet this demand. The new co-simulation tools are still in their infancy and the capabilities are often limited to only being able to execute a few models at any one time. The tools often require considerable understanding of all component models and the simulation package itself. The software solutions are gathering pace and increasing their functionality with each subsequent release. Due to the current limited functionality of the integration tools they are hence aimed at specific domains, specific modelling processes, or specific tools. Examples of such tools include; Flowmaster, TISC Suite, ChiasTek, VLAB Works and CosiMate. A subset of the co-simulation tools will allow limited integration of models from different environments. Such integration between environments tends to be limited to specific tools and only specific versions, and even only specific functions.

2.8.3 Off-The-Shelf Modelling Packages

There has been a recent trend for domain specific modelling and simulation packages to be developed. Many engineering domains now have these specific packages that allow for off-the-shelf, low fidelity simulations of whole systems. The automotive industry is not different in this regard. There are tools available that simulate the entire behaviour of vehicles as well as their major sub-systems. The intent of such software is to aid in the design of new vehicles rather than incremental improvements of an existing vehicle. These off-the-shelf tools have specified system architectures often for a multitude of specific vehicle types (e.g. hatch back, 4by4, and saloon). Often these tools are out of the box prepopulated with models. In some cases it is possible for the users to tune parameters of specific attributes of these models. This allows the automotive engineers to quickly ascertain if potential solutions are indeed plausible. Often these tools come with a means by which the final simulation can be seen in a pseudo real time virtual environment and basic full system characteristics such as 0 to 60mph can be captured.

At first inspection such tools appear to be the perfect solution to solve many of the problems associated with developing a new product, however using such a tool does have its weaknesses. When engineers wish to replace the existing models with more representative ones for their potential designs they are faced with restrictions. The integration between the component models often use a strict, bespoke standard and the way in which the tool is structured implies a specific architecture for possible solutions. When models from different environments are integrated into the tool they have to apply with the standards and architecture. The similarity between the original model and the one which it is to replace often has to have many of the same attributes such as time, inputs and outputs, and the same data types. Such fundamental modelling aspects such at timing and causality have to be identical; any deviation will mean that

the model integration tool will give erroneous and in some cases non-repeatable results.

Often the way in which such full vehicle modelling tools are currently being used is for the verification of whole virtual vehicle tests using models that have been produced by subject matter experts in isolation from one another. For the purpose of modelling individual sub-systems to the required fidelity for innovative research and development it is often necessary for the subject matter expert to use methods and tools that are only available in specific development environments. This indicates a growing problem as organisations push the envelope of understanding so to must the boundaries of such integration tools also have to be pushed. The tools also have to balance being flexible enough to adapt to fast changing modelling culture, while still being reliable for meaningful testing.

2.9 SUMMARY

Using the work that was conducted to understand the nature of the problem space, this literature review was formulated to ascertain the current state of model and simulation integration approaches and techniques. It is clear from the overwhelming weight of evidence that modelling and simulation has become an integral part of the current engineering process. It is also apparent that there are many who have been researching potential ways in which models and simulations can be integrated to gain more information. This has led to many approaches and methods to have been developed in many different scientific and engineering domains. Currently there are many competing approaches to the problem. Each of the identified approaches has been evaluated based on the demonstrations given in case studies and their suitability for use within this problem space. It has been found that there is no single approach that has been found to be applicable across all examples from every domain. A distinction between the approaches that are taken and the methods by which the approaches are implemented has been found. There is a plethora of methods that have been developed and each one has its strengths and weaknesses. Some have even attempted to gain the strengths of multiple methods by amalgamating one or more of them together. However each method is suited to a particular domain or even a subdomain.

As stated above, it is clear that there is currently no one approach with a single implementation method that is suitable for all situations. It is recommended that each situation is taken on its own merit with the most appropriate approach and method selected for each specific situation. When considering how to integrate models and simulations the modelling methods that have been used in their creation has been shown to have a direct effect on the integration task. This highlights the need to understand how the models are created as the internal workings have a direct effect on how and if it is even possible to integrate it. The semantics and assumptions that make one model and simulation like no other has a direct effect on whether the output of integrating two or more models or simulations will produce a meaningful output or not. From the literature it is evident that for integration to be meaningful moving towards white box understanding of the semantics and assumptions made during their development need to be understood by those undertaking the integration of models and simulations.

One of the most significant challenges faced with the possibility of full product simulation is the issue of modelling scales and the need to cross between them. There are existing methods to aid in this task however their effectiveness is limited. Hence this area has an active research community. This also highlights there are limitations as to what we can mathematically align for meaningful integration.

An issue that was identified early on was shared understanding between those who are involved with the design of the system. All through this initially appears to

be somewhat of a trivial problem there are still issues that have not been resolved. Many of the current methods that are in uses revolve around the use of written text and the spoken word.

Throughout the literature there are clear differences between academic journals and the less formal literature produced by practitioners. This may be indicative of a deeper problem within the reporting of findings. For this reason more trust has been placed in approaches, methods, and tools that have working realistic case studies to support them.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a vehicle which has a combined ABS and steering system.			✓
2) The simulation is to capture the behaviour of the vehicle with a sinusoidal steering input.			✓
3) The simulation needs to be run multiple times with the speed of the vehicle changing across operational speeds from 10KH-1 to 115KH-1			✓
4) The model is to contain; Driver input, ABS System, and steering system.			✓
5) The outputs of the component systems are to be recorded.			✓
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) If LabVIEW or Matlab is used, only a single license may be used.			✓
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓
G) All component parts are to be in the public domain.			✓

Table 4.1: Verification RAG assessment of preliminary architecture.

With the architecture being found to be complicit with the simulation requirements the next stage can be moved onto.

2.9.1 PRELIMINARY SIMULATION DESIGN

The behaviour of the parts defined by the architecture can now be broadly defined.

User Defined Input
A user interface is to be used whereby the user can input parameters and view the results of the simulation. This is to be a single input interface for all of the component parts.

Steering profile
The inputs steering profile is to be a sinusoidal waveform.

Steering Control
Model that represents a steering system, which takes user inputs and assists the user alter the angle of the wheels. The user inputs are too represented by inputs from the steering profile component.

ABS System
Model that represents the behaviour of an existing anti-lock brake system. The model is to capture the behaviour of the modulation of the brake pressure to stop the wheel from locking completely.

Control System
A controller that takes inputs and gives outputs to and from other system components as required.

Results
A component that takes inputs from other simulation components and saves them in a suitable format.

2.9.2 VERIFICATION OF PRELIMINARY SIMULATION DESIGN

The preliminary design can be verified against the simulation requirements. The RAG assessment method has been used to analyse whether the preliminary design metes the simulation requirements. This RAG assessment can be seen in Table 4.26.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a vehicle which has a combined ABS and steering system.			✓
2) The simulation is to capture the behaviour of the vehicle with a sinusoidal steering input.			✓
3) The simulation needs to be run multiple times with the speed of the vehicle changing across operational speeds from 10KH-1 to 115KH-1			✓
4) The model is to contain; Driver input, ABS System, and steering system.			✓
5) The outputs of the component systems are to be recorded.			✓
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) If LabVIEW or Matlab is used, only a single license may be used.			✓
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓
G) All component parts are to be in the public domain.			✓

Table 4.2: RAG assessment of the preliminary design.

The RAG assessment in Table 4.26 shows that the preliminary design meets the simulation requirements and so it is possible to proceed to the next stage.

2.9.3 ARE THERE ANY EXISTING SIMULATIONS AND MODELS

The elements that are composed as parts of this system are well established. It is recognised that in a genuine engineering setting there would be existing models and simulations that could be of use in this situation. To emulate this repository a literature review was conducted and it is well known that models exist. This will impact the SESEM process but that will be explained in due course.

2.9.4 SYSTEMS ENGINEERING OF SELECTION OF EXISTING MODELS SESEMS

The SESEM is a sub-process that is concerned with aiding in the selection of existing models. As the potential for existing models being of use then this process can be implemented.

1 Preliminary Simulation Design

The overall goal of the system being designed as well as the simulation being designed is well understood. There is also an intended architecture and design for the simulation being produced. The preliminary requirements for the SESEM process can considered satisfied.

2 Boundary of the Existing Model Selection Process

A secondary check has been conducted to assure that all of the prerequisites have been satisfied. From this point on it is considered that the SESEM process is the structure that is being followed.

3 Assess the Model and Simulation Landscape

An assessment has been made of the ways in which the interactions that are to be investigated have previously been modelled and simulated in the past. Information has been gathered as to the mathematical tools and representations which may be of use.

4 Are Potential Models Available?

From the assessment of the modelling and simulation landscape there are a number of potential models that could be of use.

6 Is this a New Product/ Platform?

The component sub-systems that forms the bases of the proposed system are well established and so too are the platforms. Hence the decision to consider that this is not a new product or platform has been made.

16 Locate Previous Product Models

An assessment of the various off the shelf models that could be of use have been collected. The models that have been located include:

- Modeling an Anti-Lock Braking System
- Modified Anti-Lock Braking (ABS) Model
- Vehicle Body
- Power-Assisted Steering Mechanism
- Simple 2D kinematic vehicle steering model
- Tyre simple
- Tyre (Magic Formula)

There is more than one tyre and steering models that could potentially be of use. However at this stage having more than one model is not a hindrance.

17 Available Model Documentation

For each of the identified model a simple assessment of whether there exists sufficient documentation available has been conducted. If the documentation is not available or has insufficient detail, is there someone in the team that can understand the model explicitly has been ascertained. The results of this assessment are captured in Table 4.27 below.

MODEL NAME	DOCUMENTATION AVAILABLE	TEAM UNDERSTANDING	SUITABLE
Modeling an Anti-Lock Braking System	✓	–	✓
Modified Anti-Lock Braking (ABS) Model	✘	–	✘
Vehicle Body	✓	–	✓
Power-Assisted Steering Mechanism	–	–	–
Simple 2D kinematic vehicle steering model	✓	–	✓
Tyre simple	✓	–	✓
Tyre (Magic Formula)	✓	–	✓

Table 4.3: Representation of the assessment of the documentation availability. Three symbols are used; ✓ full, – partial, and ✘ not at all.

Any models or simulations which do not have sufficient documentation or available understanding are deemed unsuitable. It is worth noting that Power Assisted Steering Mechanism only has limited documentation whether what is present is sufficient will be discerned later in the process. The team understanding is partial for all of the selected models.

19 Does the Model Match a Section of the Simulation Requirements?

For Each candidate model it is evaluated against the simulation requirements. A RAG assessment of each of the potential models has been captured in the Table 4.28 below.

REQUIREMENT	Modeling an Anti-Lock Braking System	Vehicle Body	Power-Assisted Steering Mechanism	Simple 2D kinematic vehicle steering model	Tyre simple	Tyre (Magic Formula)
1) The simulation is to capture the behaviour of a vehicle which has a combined ABS and steering system.	A	R	A	A	A	A
2) The simulation is to capture the behaviour of the vehicle with a sinusoidal steering input.	R	R	A	A	A	A
3) The simulation needs to be run multiple times with the speed of the vehicle changing across operational speeds from 10KH-1 to 115KH-1	G	G	G	R	G	G
4) The model is to contain; Driver input, ABS System, and steering system.	A	R	G	A	A	A
5) The outputs of the component systems are to be recorded.	A	A	A	A	A	A
A) The total run time of the simulation should take less than five minutes to fully execute.	G	G	G	G	A	G
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.	G	G	G	A	G	G
C) No specific computational hardware or peripherals are to be used.	G	G	G	G	G	G
D) The modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.	G	G	G	G	G	G
E) If LabVIEW or Matlab is used, only a single license may be used.	G	G	G	G	G	G
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.	A	A	A	A	A	A
G) All component parts are to be in the public domain.	G	G	G	R	G	G

Table 4.4: RAG assessment of the selected potential models. Red (R) does not comply, Amber (A) partly complicit, and Green (G) fully complicit.

20 Assess if Individual Models Can be Modified

Not all models can be modified for a number of reasons that are detailed in section 6.3. Each of the selected models are assessed as to if they can be modified. The extent to which the can be modified is not in conjecture at this point it is full access modification or nothing.

MODEL NAME	CAN MODEL BE MODIFIED
Modeling an Anti-Lock Braking System	✓
Vehicle Body	✘
Power-Assisted Steering Mechanism	✓
Simple 2D kinematic vehicle steering model	✘
Tyre simple	✘
Tyre (Magic Formula)	✘

Table 4.5: Assessment of whether the potential models can be modified. The assessment is a simple yes (✓) or No(✘).

14 Selected Models Unusable

For the issues with the documentation and or the ability to modify the existing components the following existing models are considered unusable.

MODEL NAME	MODEL USABILITY
Modeling an Anti-Lock Braking System	✘
Vehicle Body	✘
Power-Assisted Steering Mechanism	✘
Simple 2D kinematic vehicle steering model	✘
Tyre simple	✘
Tyre (Magic Formula)	✘

Table 4.6: Assessment of whether the potential selected models are usable. The assessment is a simple yes (✓) or No(✘).

None of the identified models are compatible for the intended simulation. If a model cannot be modified it has to meet all of the requirements exactly. For 'Vehicle Body', 'Simple 2D kinematic vehicle steering model', 'Tyre simple', and 'Tyre (Magic Formula)' require modification to make them complicit with the requirements.

With the understanding of two remaining models that is present in the documentation issues became apparent. The 'Modeling an Anti-Lock Braking System' is an unrealistic and only meant as a demonstration model. The power assisting model documentation does not cover the assumptions that were used during its creation. As well as many of the components are ideal in nature and

not representative of the systems that would be implemented in the design. The SESEM process dictates that in this instance it will then be necessary to locate more new models or follow the path of creating new models.

For this case study the models originally selected were the only freely available models of this system that were found during the sweep of available models within the constraints of this research. The process dictates that in such a position if existing models cannot be located then new models are to be made. This changes the task form integrating existing models and simulations to developing a new simulation which may not even need to be integrated. Developing a new simulation from a blank slate is not the focus of this research as that is a very different topic. For this reason this case study will not progress, any further.

2.10 EVALUATION OF METHODS

The two fully worked case studies provide a reference point to which meaningful evaluations of the proposed methods can be made. Throughout the work that was conducted for the case studies strengths and weaknesses of the proposed methods became apparent. These identified strengths and weaknesses are discussed below.

2.10.1 STRENGTHS OF THE PROPOSED METHODS

From conducting the case studies, clear strengths of the proposed methods became apparent when compared with the current identified methods as discussed in section 2. Many of the identified strengths of the proposed methods originate from the structure that they bring to the integration task. This structure breaks down the seemingly chaotic integration task into repeatable stages that otherwise rely on the engineering experience and understanding of those taking part in the simulation task.

The way in which the process has been created is such that it does not dictate how to conduct each task, but rather it is more of a requirement for the thought process of the user. This allows for the user to select the most appropriate tool for the job for their domain to conduct the required analysis at that stage. As demonstrated in the two case studies, different tools were used for the same process element dependent on the information that needed to be processed. In different situations some tools will inevitably be more suitable than others and having this flexibility increases the potential area where this process remains relevant.

Having stages in the integration task make it possible for more than one engineer to work on the task simultaneously. In many cases it is possible for multiple engineers to work on the same task element at the same time. For case study two, elements one to six were effectively conducted by more than one engineer as the premise of the test came from negotiations with industrial engineers and academics while the rest of the stages were completed solely in an individual academic environment. The decision stages of the processes (traditionally a problem for group work) do not necessarily rely on just an individual making the decision as they could be conducted in a meeting setting or even using another systems engineering tool. At any of the stages it is possible for the process to be handed to another engineer as long as they are also familiar with the process and all of the previous stages have complete sets of the recommended documentation. This also allows for breaks in the work to be possible. While conducting case study two there was in fact a three week gap. However it took very little time to pick up where it had been left off. Having such a process allows for greater flexibility of engineers time.

By specifying the purpose of the simulation at the start acts as a reference point throughout the rest of the process. This reference point is used to formulate the

basis of the verification to which the whole simulation is subjected at various points. This static reference point has been of benefit as it allows for the simulation to be formulated in layers of complexity and functionality, meaning that the sources of the component parts to potentially be widely different, while keeping focus on what the simulation is attempting to achieve. In case study two there was the real potential for the work to progress and produce a simulation regardless, however the defined purpose of the test ensured that the work remained focused and hence halted. Without the static reference point this may not have been the output.

The requirements writing guide allows for structure and similarity across the requirement sets. This was found to be beneficial across both case studies as it improves not only the quality of the requirements but also the effectiveness of more simple analytical tools. Having the simulation and constraints separate ensures that consideration is not only focused around functionality but also to the computational capabilities of the resources available. For the second case study the simulation constraints could have coursed a real problem if certain potential simulation components were used.

The integration tables have demonstrated within the case studies as being one of the most valuable outputs of the proposed methods. The tables capture of the semantic information needed for meaningful integration. They are only completed once an understanding has been gained regarding the model which will be the subject of the tables, and only if the likelihood of the model being of use is high, waiting to this point to complete the tables is due to the resources required to conduct the task is considerable. Having such information stored in readily readable tables means that any two models with completed tables can be assessed to ascertain if they are semantically similar enough for potentially meaningfully integration. Critically this makes it possible to assess whether two or more models have the potential to be meaningfully integrated without the original modellers being involved. This is all before actually investing the time into not only the integration task but also the time needed to ascertain if the results of the integrated simulation are meaningful. This result is a considerable reduction in the resource overhead that it takes to produce a meaningful (with high confidence), integrated simulation when compared to current methods if rework is required. The integration tables facilitate many models being compared with the information stored in them being uniformly structured removing the need for integration engineers to hold all of the relevant information in their own working memory. Even in the limited example of case study one, the amount of information that was required to have available was such that the tables became of considerable benefit.

If the model integration process does not go any further than the generation of the integration tables it does not mean that the work is of no use. It was found that in both case studies that the integration tables were far more thorough and concise when compared against the documentation that was supplied with the

models. Once the integration tables are generated they can be stored for others to potentially use at a later date. The semantic information held within them will still be relevant for later work.

The work that was conducted in case study two highlights the processes ability to locate the potential integration issues of existing models on a semantic level. This is of even more significance as it is capable of this even when the person following the process is not a domain expert of the potential simulation components. This validates that the engineer does not necessarily need to be from the domains of the potential models to be able to assess if integrating them could be meaningful or not. This capability was also found in case study two, interestingly at the exact same point of the process. This indicates that the understanding of the models and the comparison against the simulation requirements is the first real sorting stage of whether or not a potential component is suitable.

The work involved with the setting up of the experiment for the first twelve elements of the Systems Engineering in Integration of Simulation process took a considerable amount of time. However having a designed experiment is of significant use when formulating the verification of the simulation. The output of the first twelve stages of SEIS give a static reference point which can be used for the verification task later on. Having a reference point means that if there is a variation from said reference it can be identified and corrected. The reference point critically allows for verification to be more than just the validity of the data transfer between components being acceptable. Without such a well-established reference point the validity of the verification testing could be brought into serious doubt.

2.10.2 WEAKNESSES OF THE PROPOSED METHODS

While conducting the case studies it was not without issue and weaknesses of the proposed processes became apparent. Many of the issues stem from the increase in time that it takes to get to the point where programming starts to take place. This is due to the trade-off between the time that is invested in getting the integration correct the first time, rather than the time after the simulation is completed and the inevitable rework that has to be done.

When conducting the proposed methods it became apparent that the amount of effort and time that goes into a stage is not represented by the textual output. This could be a potential issue when managing engineers using these processes. Elements of the SEIS process where this is particularly prevalent are 1, 7, 11, 19 and 26. For the SESEM process it is elements 3, 8, 10, 11, 18, 23, 24, 25, and 28. From the number of tasks that require considerable thought and work with little visible output may trouble some managers.

The task of locating previous models in this process is but one single stage however it is a non-trivial one. It would benefit from further research and

guidance as to how the engineer could go about locating these existing models and simulations. This stage was one of the reasons why the second case study ran into the problems that would later halt the progress of the work.

When conducting the SESEM process during both the case studies a potential issue became apparent regarding stages 12, and 19. At this stage existing models are held against the requirements for the simulation being designed. There is hence the potential need for requirements for the potential design. As the overall requirements of the simulation may not be fully applicable in this situation as the potential components are unlikely to be applicable at that stage. The user has to make the decision as to whether the potential component is of worth to continue using and investing in potentially reworking of the component.

2.10.3 The Effectiveness of the Proposed Methods

The application of the proposed methods provides structure to the integration process and produces a situation where there is both a potential solution and a means of conducting meaningful verification. This however does come at a cost of time and resources. In case study one the methods proved that they could be used to produce a simulation form existing models, whereas in case study two the processes highlighted that there were real serious semantical difference and potential problems with using the identified models. This indicates the effectiveness of the proposed methods.

2.11 SUMMARY OF CASE STUDY TESTING

The two case studies demonstrate how the proposed methods in section 3 can be implemented. The first case study is a full worked example from product concept through to the point where a physical prototype can be made and the virtual testing validated. The second case study represents an automotive example where the selected component models and simulations were identified as having significant semantic differences. The extent to these sematic differences resulted in any potential integration being meaningless.

The critical findings from the two case studies are: the value of having a defined reference point that can be used through development for verification and validation, the ability to reliably identify semantic differences between the potential component models and simulations, and that the methods allow for non-domain experts to be able to make an assessment of the suitability of potential integrations before the work is put in to integrate and test the full simulation. However the greatest weakness of the proposed methods is the time it adds to virtual simulation and test. However there is the indication that automation technologies such as NLP could vastly reduce the time it takes for current repetitive deductive tasks, see section 5.

3 SYSTEMS ENGINEERING FRAMEWORK AND PROCESSES

3.1 INTRODUCTION

From the understanding gained of the problem space identified in section 1.2.2 combined with the information gathered in the literature review section 2, a novel representation of the systems engineering framework is proposed. This systems engineering framework forms the basis of a proposed end-to-end process that guides the user from customer wants through virtual design and testing, and to a point where a system design can enter manufacture. This process is expressed using flow charts combined with textual descriptions. Some of the elements in the flow charts represent processes in their own right and are decomposed further.

It is proposed that not all integration is meaningful. Just because it is possible to pass data between component parts does not mean that the resultant output is meaningful. The minimum amount of information required to make an assessment of two or more models or simulations that can be meaningfully integrated is proposed. A means of how to capture, store and integrate this information is detailed. The potential to automate the data capture process is discussed.

Levels of abstraction were recognised in the literature as being a key source of difficulty when assessing the meaningful integration of models or simulations. A definition is given to the term abstraction, from which a means of ranking and recording models and simulations is proposed.

3.2 NOVEL SYSTEMS ENGINEERING FRAMEWORK

With the development of new methods and thinking, innovative ways of representing this knowledge are often produced to support communication. To show the interrelation between the different stages of the systems engineering life cycle a diagram has been developed to demonstrate how sections build upon each other, while also showing the interrelation of the component parts. Within systems engineering while conducting one part of the project simultaneous consideration is given to another part. The reasoning behind this is to identify potential problems before they become too much of an issue. This concept can be seen in the Vee model [76] and Spiral Diagram [77]. The core systems engineering concepts of verification and validation that are expressed in Figure 3.1 through Figure 3.4 are explicitly stated in section 3.5.2 and 3.5.3 respectively.

3.2.1 LINEAR SYSTEMS ENGINEERING

When systems engineering principles are implemented in an organisation, the component systems methods are often linearised to fit with existing traditional engineering practices. This linearisation results in an implementation where each step builds upon the previous one. A representation of this process can be seen in Figure 3.1 below.

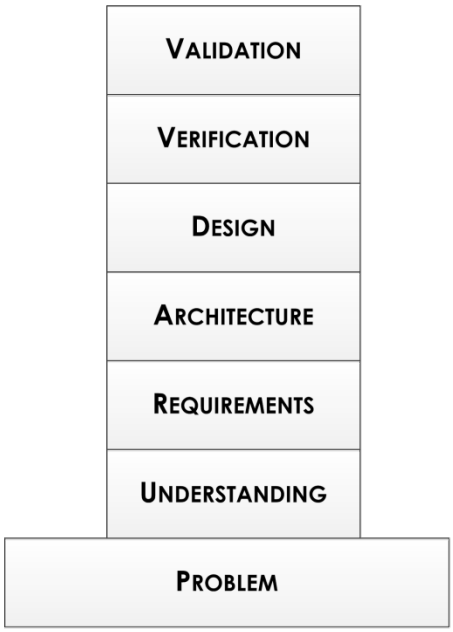

Figure 3.1 Linearised systems engineering method. This representation shows how the stages starting at the problem with each stage building on the previous stage. Note how the understanding occupies less area than the problem.

In Figure 3.1 each stage builds on the previous. The starting point is the problem being addressed. Without a problem being present it is unlikely that any engineering would take place. Often the problem is expressed in the form of a tightly bound space. The 'Understanding' section is the first active stage of the process. The understanding of the problem being addressed can only be a

perception of the problem due to all understanding only being a representation of the real world. The 'Requirements' that are formulated are built directly on the understanding of the problem space. However they may not utilise all of the understanding that was gained of the identified problem. The architecture is constrained by the requirements. The design is formulated using the defined architecture and the requirements. Once the design is implemented it is verified (in sections or as a complete system) using the requirements as a reference point. Validation is applied to all previous steps and requires all other stages to be completed. It is the last task to be conducted before the design is put into production.

3.2.2 STACKED SYSTEMS ENGINEERING WITH PARTIAL VERIFICATION

Taking the idea of linearised systems engineering and the representation of how some of the potential parallelism can represented using the principles of Figure 3.1, see Figure 3.2 below.

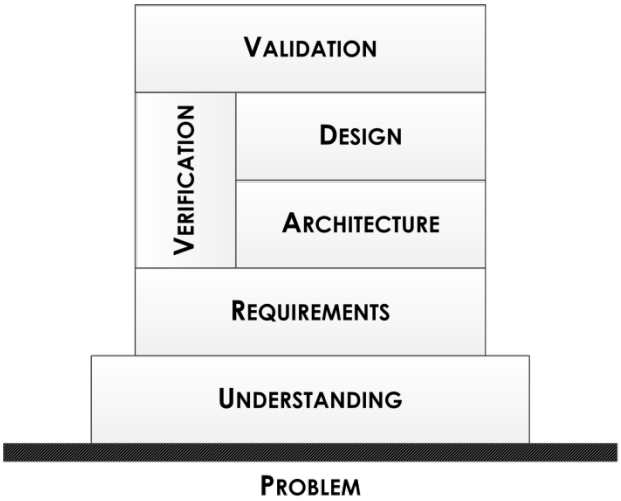

Figure 3.2 Basic stacked systems engineering. The problem is now an unbounded plane and verification happens concurrently with architecture and design.

Some of the assumptions made in the linear systems engineering are problematic and not reflective of what is found in the real world. The problem that is the source of the engineering solution is not a nicely bounded entity that can simply be plucked up out of the ether and used. It is rather an entity that needs to be investigated and explored. Even when the problem is investigated the viewpoint of the investigator consists of only what they are looking for rather than what is necessarily the real source of the problem. In Figure 3.2 the representation of the problem has been altered from a bounded box to a region of a plan that can be interrogated, resulting in information being gained. Verification is not something that is done at the end of a project but rather as intermediate checks throughout to ensure that the system is being built right. Hence verification happens throughout the architecture and the design stages of a project. Verification and architecture feed off the requirements, however the design feeds solely off the architecture. Validation is conducted off the design; it is also used to validate that the verification of the architecture and design were correct.

3.2.3 STACKED SYSTEMS ENGINEERING INCLUDING VERIFICATION OF REQUIREMENTS

There is a question as to whether requirements can be meaningfully verified. For this study it is considered that in some situations some requirements can be verified as long as there is a valid reference point. The standard practice for verification in the rest of a project is to use the requirements as the reference point. Therefore to verify requirements there needs to be some other reference point to use. This new reference point could be traceable to primary or secondary knowledge sources. In the case of the proposal for creating an integrated co-simulation (see section 3.5.5) the constraint requirements can be verified against functionality requirements. However the functionality requirements have no other reference points other than the stakeholders and the engineer's understanding of the problem space, both of which would have been used in the capture of the requirements in the first instance. Hence any verification test using the source of its creation as the verification reference point would result in the outputs being of questionable validity. To represent the question as to whether the requirements can be verified has been represented in Figure 3.3, where the intersection of the requirements box meets the verification box , this area has a question mark in it.

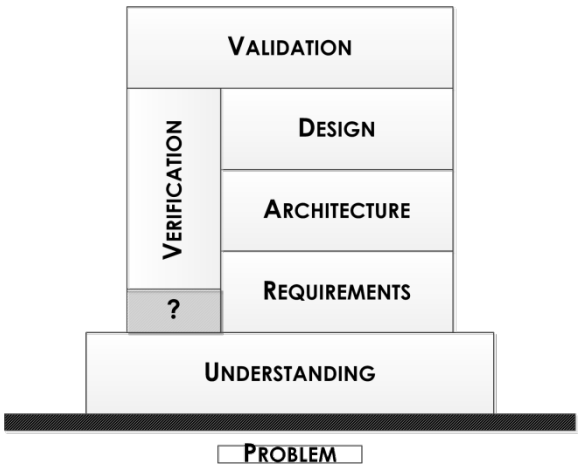

Figure 3.3 Stacked systems engineering verification of requirements questioned. Note the box with a question mark in denoting the area in question.

3.2.4 STACKED SYSTEMS ENGINEERING

In this representation of systems engineering the issue is how the validation has been addressed (see Figure 3.4 below). The understanding is built upon the problem. The requirements are built upon the understanding of the problem but do not use all of the understanding that has been gained. The requirements underpin and directly influence the verification procedure, the architecture, and also the validation tests. This representation of systems engineering indicates what parts are built upon which sections and how they interrelate.

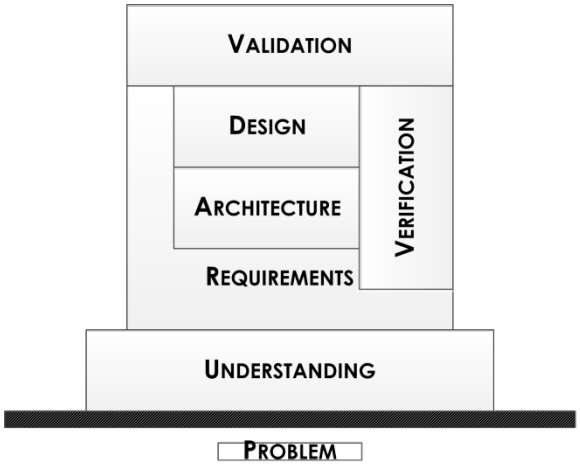

Figure 3.4 Stacked representation of Systems Engineering. The requirements are partially verified and are used to guide architecture, verification, and validation stages.

It is this representation that is the basis of systems processes for model and simulation integration in this work. The concepts of understanding the problem, requirements being a reflection of the understanding, verification from the requirements stage throughout the architecture and design phases, requirements being used as a source of validation test cases. These concepts are all used in the overarching systems engineering for model integration that is detailed in section 3.5.5.

3.3 DESIGN OF EXPERIMENTS

Before conducting experiments, strict boundaries of what the purpose of the test is and what measures are going to be taken need to be defined to ensure that only what is intended to be tested gets tested. This is an attempt to make the test as likely to succeed and the results be as valid as possible. When investigating the Design of Experiment (DOE) literature there appears to be some confusion as to what the DOE refers to. This appears to be caused by DOE referring to a loose collection of ideas, methods and tools. There are two broad areas of focus: statistical methods; and how to set up an experiment to prove or disprove a hypothesis. The two differing views of what DOE literature is concerned with is not new [78]. For the purpose of this review both views are considered. Within the literature concerning the statistical methods of DOEs Sir Ronald A Fisher is recognised as being the one to establish the core concepts in the 1920s with his work within the field of agricultural research [79].

The task of designing an experiment has been recognised as being as applicable to analysis conducted in a virtual environment as it is in the physical one [5]. Formally stating the concerns of the analysis to be conducted enables a shared understanding of the exact purpose of the analysis that is to be conducted.

The literature concerning DOEs shows that this process is being used across science and industry, including but not exclusively to; operational research [80], electrical engineering [81], chemical engineering [82], materials engineering [5], drug development [83], and general mathematical simulation [4]. Due to DOE being a loosely defined approach it is not used in all industries to the same extent. There are those who state that

> *"The design of experiments methodology is a technique that has been applied for many years in industry to improve quality"*[5]

whereas others are of the opinion that DOEs are not used as widely or effective as they should or could be which directly indicates that DOEs are not as ubiquitous as some may advocate [4].

The topic of DOEs is one that causes such confusion that articles are still published with full procedures that detail frameworks which scientists and engineers can implement for their simulations [4]. Such articles all follow similar themes with subtle differences. For the following sections regarding DOE the general themes are explored.

3.3.1 THE PURPOSE OF A DESIGN OF EXPERIMENTS

A DOE is intended to give assurance that the experiment analyses what is intended, and produces the most useful and useable results possible. The result of this is that DOEs can increase the transparency of simulation and the value of the reported results [4]. DOES can function as a reference point which can be referred to at a later date which details how and why a simulation was

developed increases the value of the existing simulation. Having a full DOE accompanying a simulation means a greater understanding can be gained by third parties. Having such an understanding has been identified in this work as being critical for successful and meaningful model and simulation integration (section 4.3)

The DOE documentation is beneficial for component selection. However there are those that see the value of DOE from a different perspective. The selection of components is further complicated if the objectives of the simulation do not remain constant. If there are changes then the engineers will have to throw-out the development of the simulations created to that point [15]. Optimal component selection is as a problem in itself and has been shown to be NP Complete [15]. Therefore to reduce a potential component of the complexity, holding the objectives of the simulation static is desirable. To enable the objectives to the simulation to be specified with enough assurance that they will not need to be changed half way through the selection or even build process, having a strictly defined DOE is important. If the DOE is not considered at all the likelihood of the objectives of the simulation changing half way through is high.

3.3.2 The use of Design of Experiments within modelling and simulation

Most of the literature concerning DOEs is focused around physical experimentation. However for the reasons discussed above there are potential benefits that are as applicable in the virtual world as they are in the physical. There are indeed highly critical remarks about how simulation is being used without considerations of scientific method such as,

> *"many articles seem to ignore the basics of experimental design"*[4]

This indicates that there is the potential for DOE to have real impact in the modelling and simulation community.

Where the variation between the physical and the virtual DOE differ is the degree of control that the experimenter has over the experiment. In the physical world attempts can be made to decouple variables though there can be interactions between variables that are either unknown or it is impossible to decouple. This is in contrast to a virtual environment where the designer has complete control over all of the interactions between variables. This can lead to only the known coupled variables being expressed in virtual environments. This argument returns to the issue of all models only being an approximation of physical phenomena.

3.3.3 Considerations When Creating a Design of Experiments

The question when creating a DOE is focused around how to develop a simulation that suitably explores and mimics the behaviour of the phenomenon that is being investigated, while identifying biases, and ensuring that only the parameters that will make a difference are varied.

Depending on the style of the author the necessary understanding of the problem that is being investigated is either explicitly stated or implied, regardless understanding is required for the experiment to be meaningful. However the key part of the whole DOE process is the identification of a suitable hypothesis based on accumulated knowledge [84]. This understanding is often where engineering differs from science. Engineers often use experimentation to understand if their potential designs fulfil requirements whereas scientists are trying to understand unexplained phenomena. This can be inferred [5] and note that engineers often have a good understanding of the problem situation to the extent that they can reduce the number of factors that need to be explored.

Across the majority of the DOE literature there are general stages that are recommended. These stages are: designing a model, building the model, verification, and conducting the test [4]. In this work they identify that the difficulty that is often encountered with simulations is that they quickly become complex systems in their own right.

To conduct meaningful testing there are also common recommendations regarding the formation of objectives, identification of variables and the way that they behave, selecting an appropriate way of changing the values of variables in a meaningful way (known as factorial design), and investigating and estimating the experimental error. The specifics of how to go about each of these stages is outside the bounds of this work. From the literature there is no one size fits all choice of which method to use. However some general reoccurring points are discussed in the rest of this section which directly relate to DOE using simulation rather than physical experimentation.

Reductionist thinking can be used to break down the simulation into smaller simulations for the purpose of verification and validation before completion rather than conducting it all at the end. There are issues with this approach as if the system is truly complex by definition, the behaviour of the individual components may not be representative of the behaviour of the components combined.

An important part of the DOE principle is to ensure that the variables that are used are independent of one another. In practice, this level of separation is not always possible. When designing a virtual simulation as part of an experiment, as the number of coefficients in a simulation increases the chances that they are all independent of each other decreases [78].

When it comes to selecting the design points that the simulation is going to use there are complete research projects (e.g. [85]) which focus purely on how to choose such design points to use in the simulation. In such work an attempt is made to reduce the number of times that a simulation is needed to execute to still give a useable output. When conducting DOEs with simulations, the number of variables that can be changed and the number of interactions between them

that can be identified is far greater than traditional sensitivity analyses allows [4]. There are common recommended methods for conducting sensitivity analysis on simulations. These include: response surface methodology, robust parameter design, as well as the traditional changing parameters and observing the effects [5]. There are those who believe that the power of modern computers allow engineers to generate systematic searches of the domains which are being tested [78]. However this is a brute force approach which may mean that the simulation has to be run many times. Such an approach can be timely when considering the large number of high fidelity models being integrated. Whereas it has also been identified that reducing the number of times that a simulation has to be run is beneficial in not only the time that the design takes but also the other resources that are consumed by such a task.

In the case where a full integrated simulation may be very costly to run, there are cases where lower fidelity models can be used to estimate the behaviour of the costly simulation, and hence can be used to select a suitable area for design points [82]. This is very much dependent on the situation being simulated as well as the understanding of problem space. Using an approach such as this allows for techniques that are usually only applied to the final simulation to be used. The techniques that were identified as being the most useful to apply to the low fidelity integrated models were: sensitivity analysis, identification of scale parameters, and identification of possible optimal spaces.

Overall DOE has been shown as an approach to be beneficial to experimentation and specifically simulation. Many academic and practitioner examples can be found in the literature that demonstrate the benefits of the approach. The similarities between the core systems engineering thinking and that which has been demonstrated within the DOE literature has not gone unnoticed while performing this review.

3.4 SIMULATION REQUIREMENTS

Simulation requirements cannot be considered to be identical to the system requirements of the system being emulated. Simulation requirements focus on the testing of the behaviour of the system being designed in a virtual environment rather than the desired behaviour of an entity being designed in its system environment. This is a subtle difference. A fundamental difference between simulation of a designed system and the system itself is that the simulation emulates not only the intended behaviour of the system being designed but also that of the environment that the system being designed is intended to operate within.

If there are any errors in understanding of the problem space, they are inevitably translated into the requirements. Due to the methods used, any errors in these requirements will propagate into designs that are based upon them. For work involving simulations, this understanding of the problem space goes further as many simulation types are sensitive to initial conditions. Initial values are parameterized using the understanding of the problem space. The accuracy of the results of such a test is inherently linked to the level of understanding of the problem space. Hence to increase the accuracy of many simulations it is necessary to have a detailed understanding of the problem space that the system being designed will operate within, before the requirements are written. When considering integrated co-simulations the necessary understanding includes constraints that will be present for the system being designed as well as to the simulation being constructed. Hence the initial understanding of the problem space is vital for an accurate simulation be that a co-simulation or otherwise.

When capturing the requirements for a simulation there will be constraints that will be related to the simulation rather than the system being designed. This is due to the operational environments being different. In specific cases the operational environment for a simulation may be the same as that of the system being designed, however such cases are not the focus of this work as there are existing processes to support such work.

It is proposed that simulations will have requirements that will state: what phenomenon is to be represented, the accuracy of the mimicry, and any constraints of computation, as well as any other specific desired functionality of the simulation. Each of these identified components of a simulation requirement set are explored in detailed below.

3.4.1 PHENOMENON TO BE MIMICKED

There is the primary need for a simulation to mimic a system which has been designed. Hence parts of the simulation requirements detail what the simulation is attempting to mimic. The basis of these requirements in the proposed ‘use case’ come from the system design phase of the product development lifecycle, as

well as the requirements of how the system being designed is intended to operate.

3.4.2 Accuracy of the Mimicry

Part of the proposed method is to use the results of the simulations to inform engineers as to the feasibility of the potential designs so they can make informed decisions. To improve the probability that these decisions are within accepted bounds of risk, a level of accuracy is required for the simulation. There is not a hard and fast level of accuracy for all models and simulations for all occasions as different levels are required for different decisions. This is why the purpose of the test is to be decided upon and specified before the simulation requirements are formulated. As with the purpose known, a suitable selection can be made.

3.4.3 Constraints of Simulation

When conducting *in silico* experimentation there are specific constraints that manifest during simulation and need to be taken into consideration. Despite the increases in computational power nothing in engineering is free and as such there are costs related to each computation. Therefore resources need to be taken into consideration, such as but not limited to: execution time, available memory, processing power, available software, and skillsets of the engineers. Companies may also have set conventions as to how simulations are to be conducted. Any such conventions need to be explicitly captured in the simulation requirements.

3.4.4 Functionality of the System

There are actions of the simulation that are not solely for mimicking the behaviour of the system being designed but rather for the analysis, recording, or modifying the behaviour of the simulation. An example of such functionality is *'The system should record the speeds of the vehicle at a defined time step in the form of a CSV file'*. Other aspects come into the functionality such as how the user will interact with the simulation being designed e.g. *'The user interface of the simulation being designed is to be operated from the command line of a UNIX machine'*. Hence how and why the simulation is to be used has to be taken into consideration when formulating the requirements.

3.4.5 The Effects of Differing Viewpoints

To construct a simulation that produces results that are of a suitable accuracy and fidelity to allow for meaningful decisions is a non-trivial task. This task is further complicated by using existing models and simulations. As the number of component models and simulations increases so too does the number of viewpoints, the result of this is an increase in the complexity of the integration problem. A simple model and simulation task when using existing models can quickly transform from being a simple task to a complicated system and even a complex system. This transformation occurs with an increase in the variety of the component parts and the means by which the parts need to communicate. One

of the foundations of Systems Engineering is to attempt to cope with complexity found in engineering tasks. It is proposed that systems engineering - based approaches and tools can alleviate some of the problem themes that have been identified in section 1.2.4. This is due to systems engineering having demonstrated that it can support the development of complex systems. It is hence proposed that the principles of systems engineering can be applied with the support of specific tools and processes that are specific to model and simulation task.

3.4.6 Requirements Compliance of Existing Models and Simulations

Having a set of requirements for the simulation means that when using existing simulation components it is possible to establish if they comply with requirements of the new simulation being designed. This however raises a fundamental question regarding requirements and systems engineering. In mainstream systems a requirement is either satisfied or it is not, there is no partial satisfaction. However when operating within this problem space it is proposed that for requirements there may be a middle grey area between compliance and non-compliance. The partial fulfilment is categorised by a model or simulation providing part of a requirement but not the whole. When considering if an existing model or simulation is of value to the current simulation being designed, an existing model or simulation may mimic part of a system being designed which on its own does not satisfy the simulation requirements but with additional work or models or simulations it may do so. Therefore the idea that an existing simulation or model may contribute to fulfilling a requirement is captured in this partial statement. This concept has been applied within the processes that have been developed in this research. The way that this partial compliance has been implemented in the process is discussed later. As the partial compliance is a relative term, it is a subjective call from engineers as to the extent to which it is worth considering a component. There are however factors to consider which include: how much work needs to be done to get the model or simulation to comply, do the additional parts exist to allow for two or more models to be connected to comply with the requirements, and is it even possible to connect the model or simulation with other components.

3.5 SYSTEMS ENGINEERING PROCESSES

The most significant criticism of current systems engineering guides is that they are too long and complicated for non-systems engineering experts to follow and put into practice. Therefore this work has attempted to capture the essence of the systems engineering principles while presenting it in a format that engineers from all disciplines can easily absorb and put into practice. In this research various methods for communicating this information were experimented with. It was found that out of the possible options, the most effective method was flow diagrams. This type of diagram is not new in engineering and will be familiar to many engineers from many disciplines. Those who have not used these diagrams in the past will quickly see the similarity between them and the procedural thinking that is used in mathematical representations. Using this representation, each stage of the process is represented by a box. There are limited numbers of different shapes the boxes can take. The boxes of similar shapes denote that they all consist of the same type of operation be that an; action, decision, boundary, validation, verification, source, or a sink state. By keeping the variety of the types of actions small it reduces the complexity of the process [3]. The directional arrows between the boxes direct the user as to the next step. Each box in the diagrams has a textual description found in accompanying documentation. This allows the engineers to use the flow diagrams as a reference point that can be inspected as and when it is needed. By using a visual representation of the relations in the process steps combined with a textual document increases the understanding for the novice user faster than a self-referential textual documentation [86]. It also shows the overall process flow in a few pages of information which can be absorbed at a glance. If the diagram is not self-explanatory then the textual document can be referred to. This will aid in them not getting bogged down in reams of text when looking for which step is next.

The overall model integration task has been broken down into the following process flow diagrams:

- ***Systems Lifecycle*** - An overview of the modelling and simulation process as a whole
- ***Systems Engineering in Integration of Simulation -*** This is a process for the creation of an integrated simulation from many constituent parts
- ***Systems Engineering of Selection of Existing Models -*** Tackles the challenge of selecting the appropriate models and simulations from a repository of existing models and simulations
- ***Defining the Gaps -*** For finding where additional work needs to be done to get from existing models to final solution
- ***Fill Gaps in the Systems Engineering in Integration of Simulations -*** The process of taking the 'new' parts and fitting them between the existing models and simulations

These processes guide an engineer from the desire to simulate, through the integration of existing models and simulation, to the validation of the potential simulation.

3.5.1 SYSTEMS CREATION LIFECYCLE PROCESS

A representation of systems engineering principles has been discussed earlier in section 3.2. This framework has been implemented in the form of a process. The process is shown below in Figure 3.5. This process was designed specifically for the development of a new product or service using virtual testing before physical prototyping. This process takes customer wants and uses them for the development of a product all the way through to the manufacture stage. This process is intended for an organisation that wishes to gain value from and reuse their existing models or simulations for a new product or service. The activities that are present in Figure 3.4 can be seen in the following process, there is a direct mapping between the systems engineering framework and the proposed process.

This systems creation lifecycle process is only concerned up to the point where the product enters manufacture. This is in contrast to other processes in the systems engineering literature that consider everything from the first thought of a system through to its disposal. This work is focused on the task of integrating models and simulations for the purpose of system creation; hence the concern ends with factors that directly affect the integration task. However this process can fit into a larger life cycle concern.

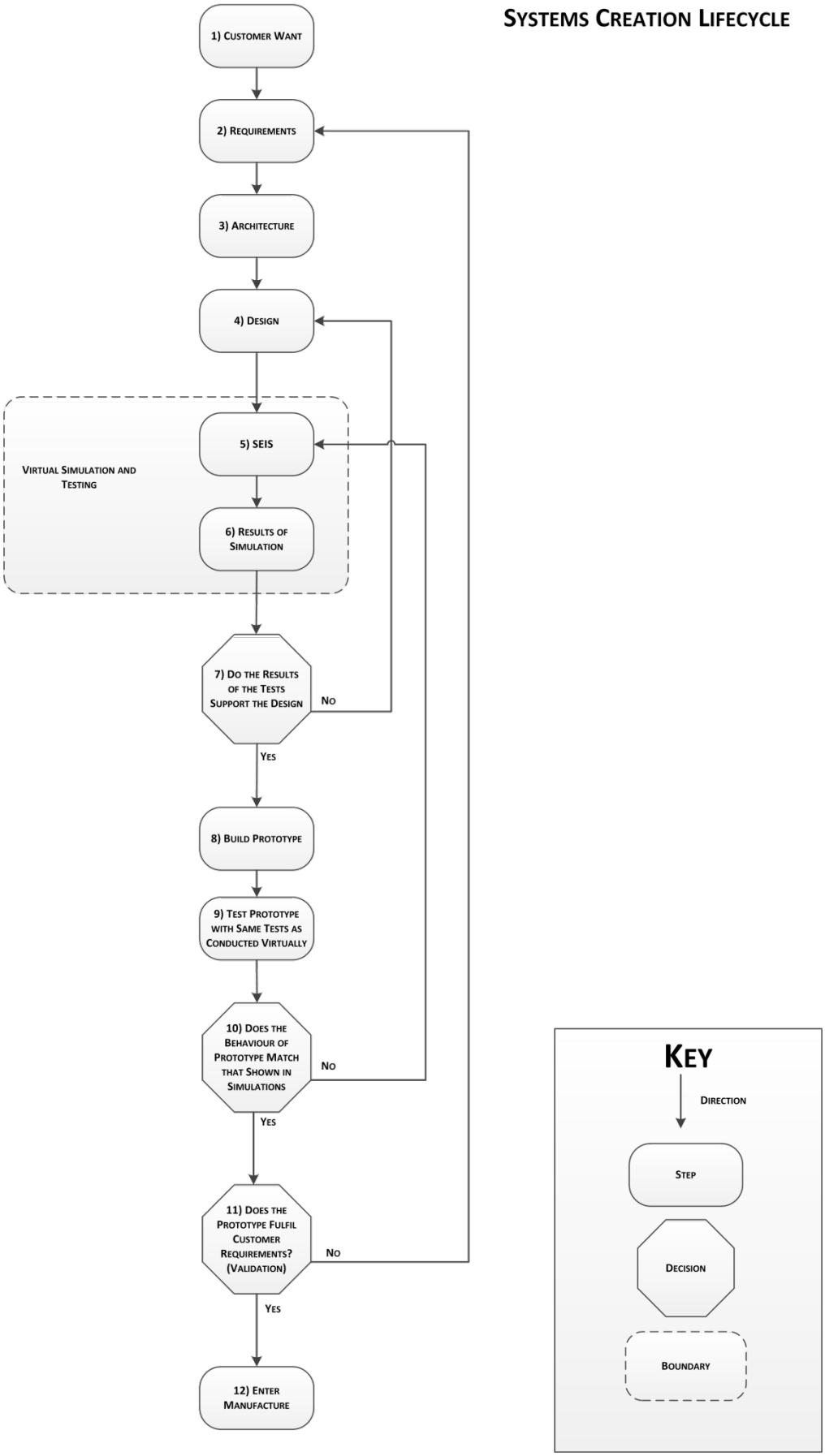

Figure 3.5 Systems lifecycle. This process is an implementation of systems engineering for the development of a product using virtual simulation and testing.

For each stage in this process there are sub-processes, some of which are detailed in this work. There are however some stages that are recognised to be necessary parts of an engineering project, but are strictly are outside the bounds of this research. Any stages that are outside the concerns of this research are identified as such and are only covered briefly.

The systems engineering concepts of checking each stage of the project against a reference point (verification) has the potential to reduce the likelihood of there being a serious error only being found upon completion of the project. Figure 3.5 highlights where virtual simulation and test fits within the larger engineering process. Its position dictates how it is treated. The elements of Figure 3.5 are described in Table 9.1.where each element is given a full textual description.

3.5.2 VERIFICATION REPRESENTATION

To ensure the work that uses the proposed processes is fully focused on what is in the requirements stages, verification testing is recommended. Verification is the testing to ascertain if what is being created is what was intended by those who specified the system. The basis of verification is comparing what has been created against a reference point. The reference point is to be agreed with the stakeholders of that system or sub-system. Any change in stakeholder between the whole system and the component sub-systems is to be taken into consideration. The customer will often not be interested in how the sub-systems are supposed to work but rather their concern is the system as a whole operates. Figure 3.6 shows the basic process of a verification exercise.

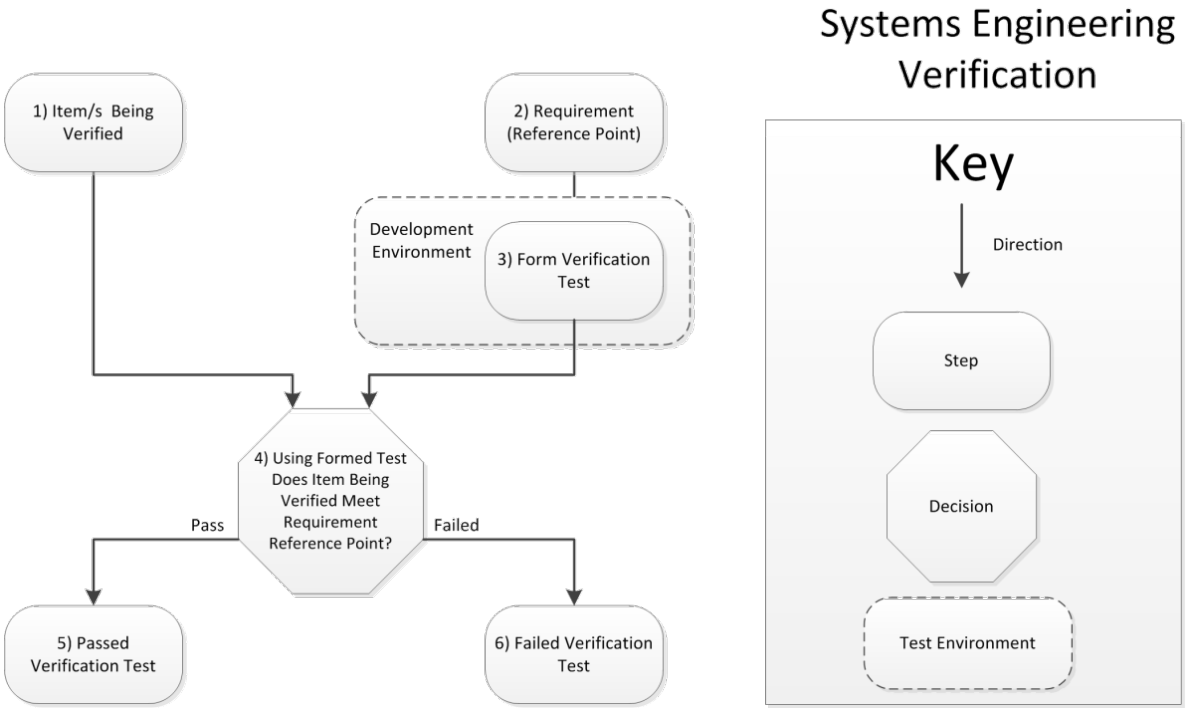

Figure 3.6 Systems engineering verification. This figure shows the basis of all verification that is proposed in these processes. It uses the same flow chart semantics as the other proposed methods in this work.

In the verification process above there is a stage 'form verification test', here the requirements being used for the system being designed have to be written in such a way as a test can be formulated from them. More information on how to write requirements in the proposed method can be found in section 3.4. Using the qualifier in a requirement such as 'simulation must not use more than 1GB of RAM during execution' allows for a precise verification. With this example a test could be conducted whereby the simulation is run and observe through a testing application to ascertain the amount of memory that the simulation calls throughout its runtime. If the simulation uses less that 1GB it has passed, whereas if it uses more than 1GB then it has failed.

If an item under test fails verification it will be necessary for a decision to be made as to if it is necessary to go back and rework the solution until it is in a state where it can be subjected to verification again and pass. Figure 3.6 elements are described in Table 9.2. A full textual description is given for each of the process elements.

3.5.3 VALIDATION REPRESENTATION

The process of validation is similar to that of verification and often the two get confused, even within the literature. The critical difference between verification and validation is the environment where the test is conducted. Validation happens when the system being produced is tested in its operational environment with operational stimulus. This means that the system has to be in a state where it can be tested in its operational environment, which often means that this type of test is only done near the end of systems developmental process. Figure 3.7 below represents the concept of validation testing. Descriptions of each of the process elements can be found in Table 9.3.

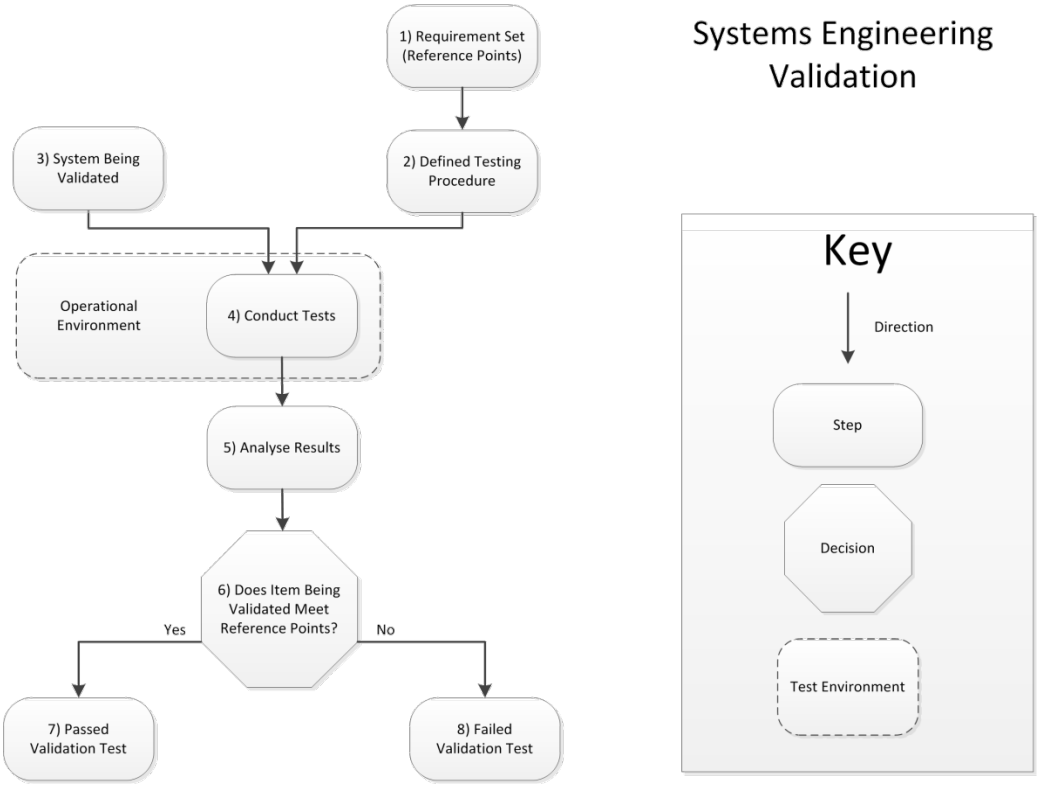

Figure 3.7 Systems engineering validation. Note how the test is conducted within the operational environment of the system being validated.

The definition of the testing procedure comes from the requirements set. Often the high level requirements are used to define what the validation tests are to be. These tests often focus on the behaviour of the whole system rather than that of its component parts.

If a system fails a validation test it often means that there needs to be considerable rework to get it to the stage where it can get retested and pass. Individual verification tests are often far cheaper than validation, not only because they are conducted in the design environment but also because they do not necessarily require the whole system to be present for the test.

3.5.4 Verification and Validation of Simulations

Due to the multiple definitions of verification and validation found throughout systems engineering, ambiguity is compounded when considering verification and validation in the context of models and simulations. It is proposed in this work that it is possible to meaningfully both verify and validate models and simulations. The definitions and explanations of each of these terms are shown in detail in this chapter, and will be used as a reference for these concepts throughout this work.

A reference point is needed to conduct either verification or validation for both models and simulations. To act as a reference point it is proposed that the models and simulations are to have requirements documentation. Having a static requirements document ensures that there is a solid reference that the modeller, and later the model tester, both have access.

It is proposed that the verification of a model or simulation or part thereof can be conducted by comparing its current behaviour and characteristics against the requirements set. This means that the stages of the model can be verified as the design process progresses rather than conducting a monolithic verification towards the end of the work. This has the potential to work well specifically if the design of the simulations is modular. By having the capability to test piece parts of a system, it enables teams of people to work on the same project without the need for all components of the project to be at the same point in development. This will only work as long as the requirements are detailed and are adhered to. The adherence to requirements is in essence what verification is used to ensure.

The validation of models and simulations can be more difficult than verification and in the ever increasingly competitive market place, is often neglected. The proposed method is for the model or simulation to be executed and results captured. The extent to which it is possible the same test as is described by the model or simulation requirements are to be conducted on the actual system phenomena. The results of the system test are to be used as the reference point for the validation of the models and simulations. The question then is a relatively simple one of 'does the model or simulation produce results that are within the bounds that have been specified in the requirements'. If the product that has been defined based off the results of the simulation operates within desired bounds there is often the argument of 'well it works doesn't it'. This mentality is acceptable in a linear lifecycle project when there is no intention of revisiting the project work at a later date. If the product is to be evolved (as is often the case in many organisations) having models or simulations that produce results that are wildly different form the actual results can have catastrophic results if used in subsequent products. If a model or simulation that does not behave as expressed in the requirements is kept, it is proposed that its actual behaviour is recorded. This information could be crucial in other projects that are attempting to reuse models and simulations from previous projects.

3.5.5 The Role of Systems Engineering In Integrated Simulations (SEIS)

The SEIS process is in effect a mapping of the core systems engineering principles (as discussed in section 3.2.4) onto the modelling and simulation task. This process guides the engineer from the desire to test a design through to feedback of results into the design process. Figure 3.8a and Figure 3.8b below show the two halves of the full systems engineering in integration of simulation process.

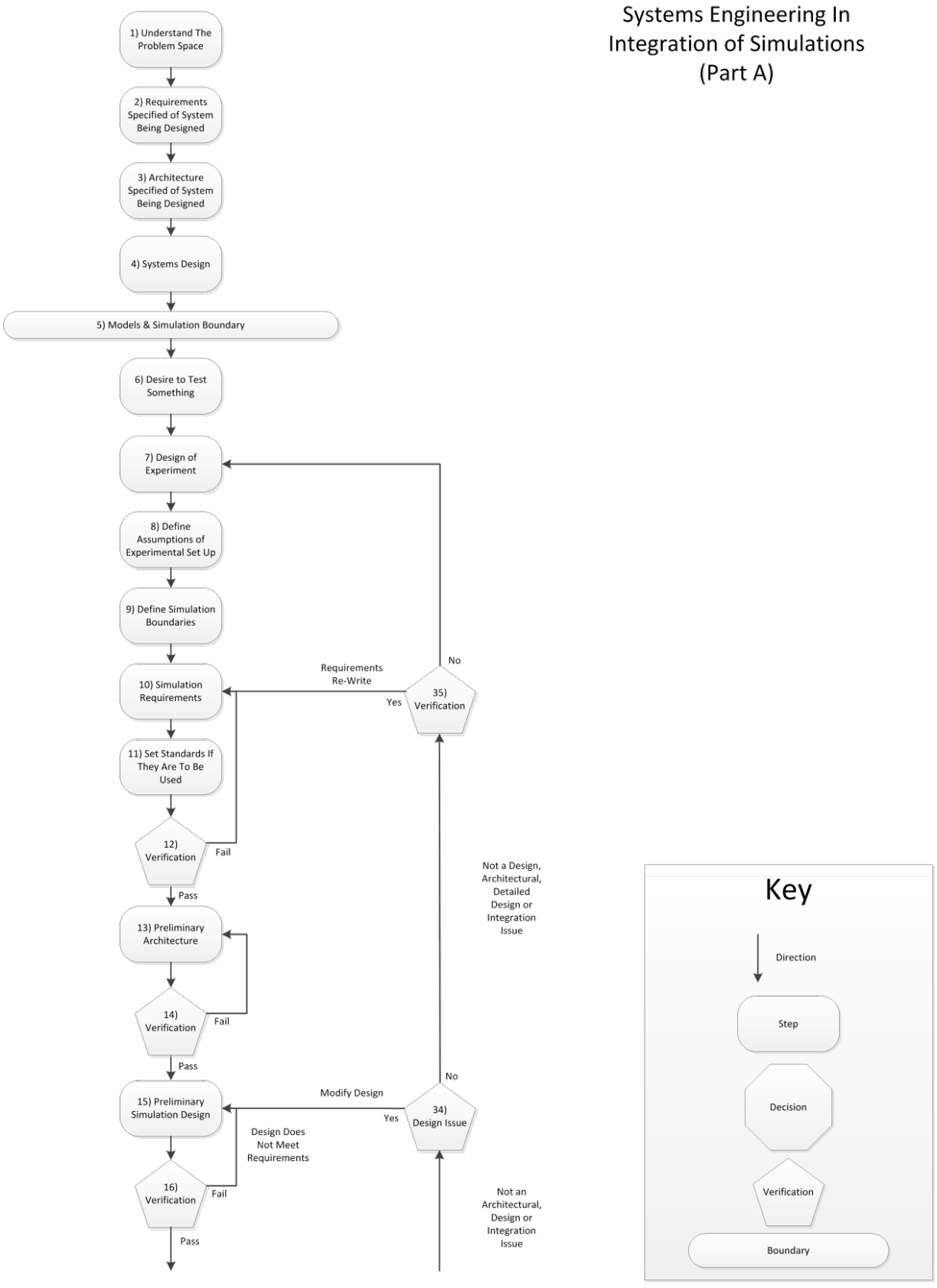

Figure 3.8a Systems Engineering in Model Integration. The first half of the process including what is needed before the simulation tests commence and to the point of the verification of the simulation design. The stages that are before the Model and Simulation Boundary are not strictly part of the process but rather stages that are prerequisite to the simulation process.

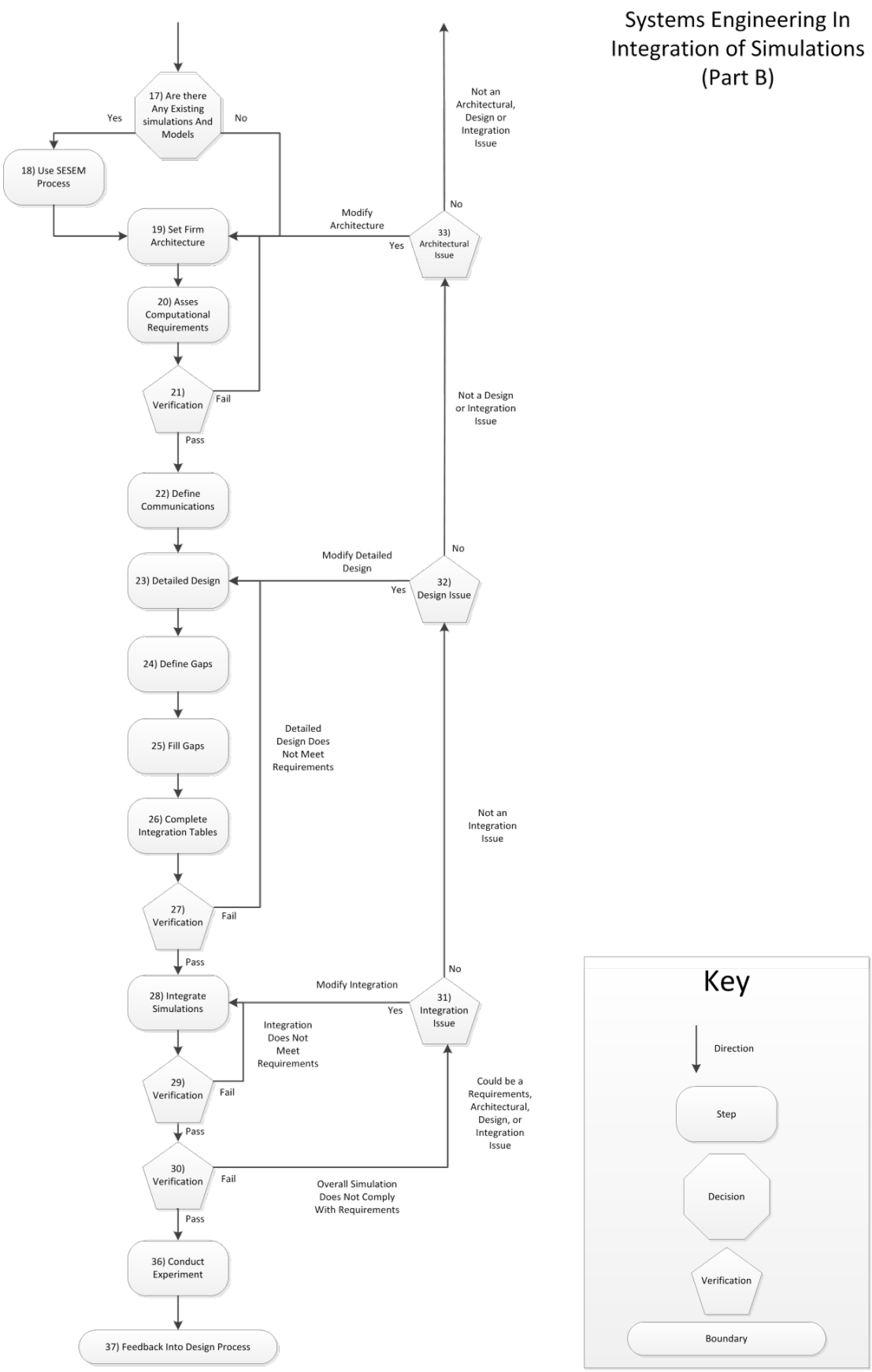

Figure 3.8b Systems Engineering in Model Integration. The Second half of the process extends to the point where the process feeds back into the design process. The two open ended arrows connect with the open arrows of the first half of the process.

The first four stages of this process (understanding the problem space, requirements specified of system being designed, architecture specified of system being designed, and system design) are all stages that form the understanding of the problem space. These parts of the process were found to be common across the majority of engineering projects, even if they were not as clearly defined as shown in this process. The model and simulation boundary is the start of the developed process. Note that in the proposed process (as shown in Figure 3.8a and Figure 3.8b) the verification maps to the requirements, architecture, and design, in the same way as proposed in section 3.2.4.

3.5.6 Systems Engineering of Selection of Existing Models and Simulations (SESEMS) (A Sub-Process)

To integrate existing models and simulations they have to be located, selected, and the viability of their use be ascertained. The selection of models and simulations from a pool of existing ones is not always as straightforward as it first seems. This process is a single step in the SEIS process detailed in the previous section 3.5.5.For this reason the boundaries are defined by the larger process for which this is a component. This process aids in the selection of candidate existing models and simulations based on the requirements of the simulation being designed. The flow diagram for this process can be seen in Figure 3.9a and Figure 3.9b with the textual representation explaining each step is in Table 9.5.

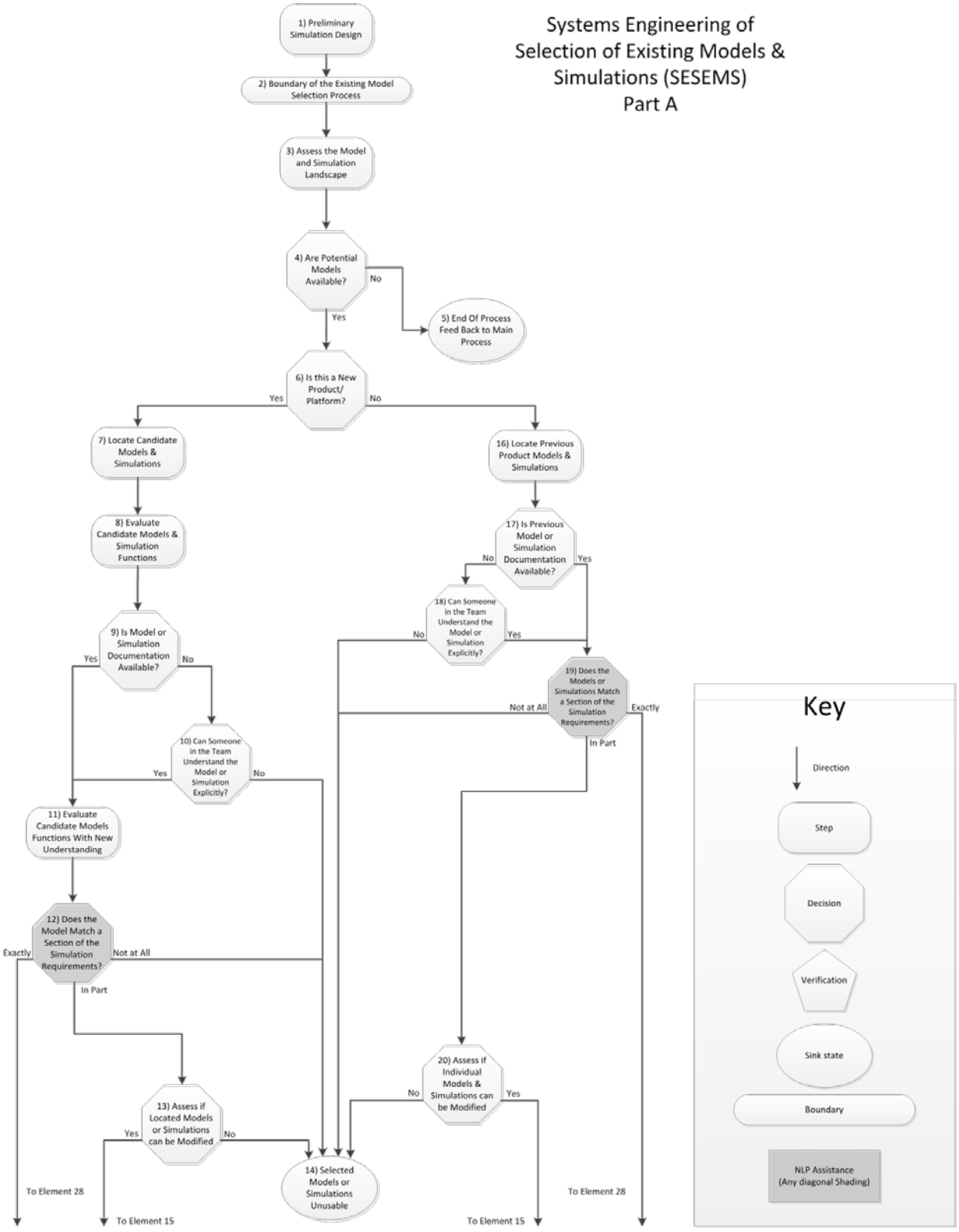

Figure 3.9a Systems Engineering of Selection of Existing Models Part A. Note the darker shaded boxes these are the areas where NLP technologies may be off assistance see section 5 for more information regarding NLP and its potential use in these tasks.

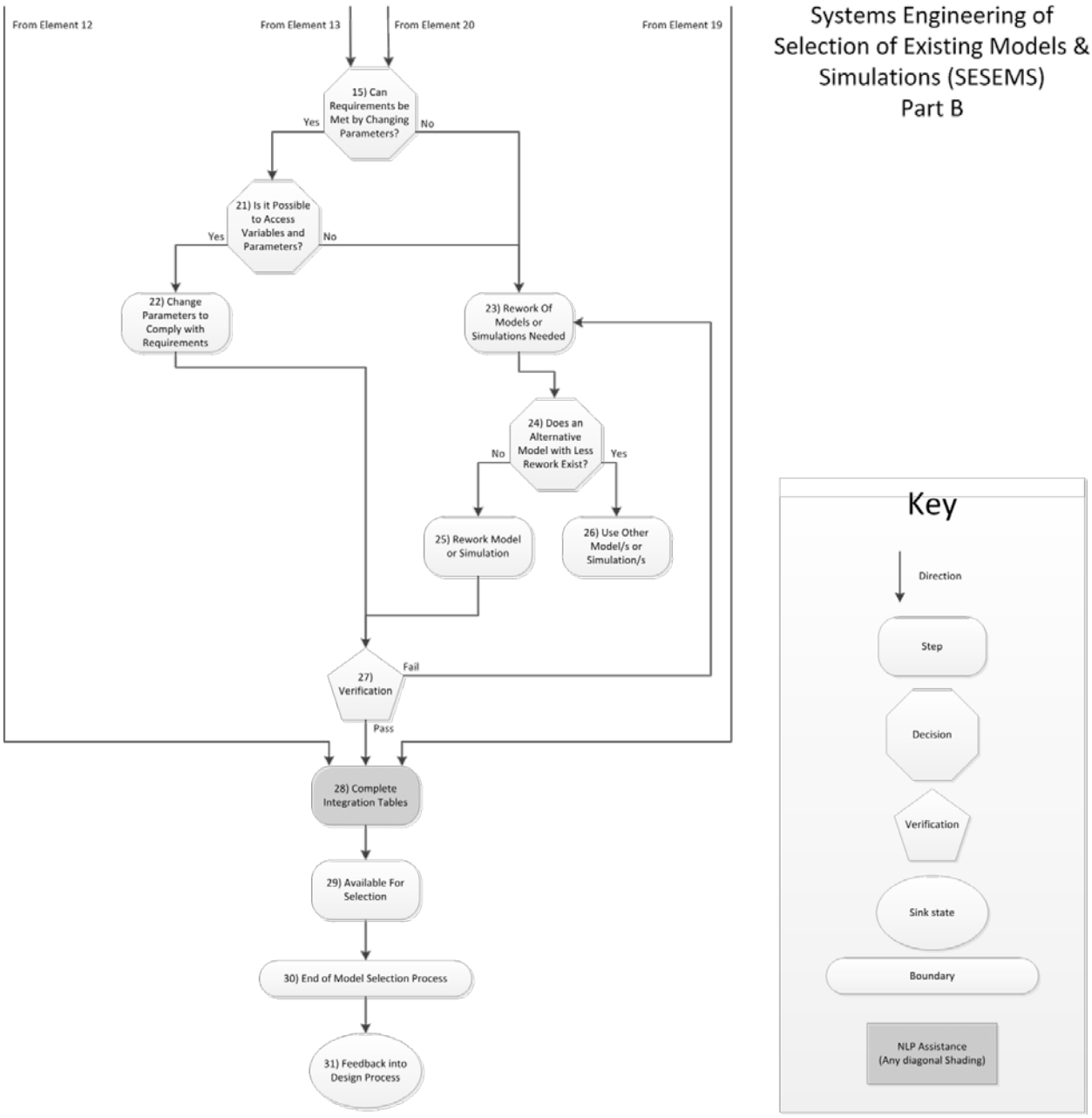

Figure 3.9b Systems Engineering of Selection of Existing Models Part B. Note the darker shaded boxes these are the areas where NLP technologies may be off assistance see section 5.2 for more information regarding NLP and its potential use in these tasks.

An explanation and the reasoning of each stage of the process in Figure 3.9a and Figure 3.9b can be found in section 9.1.

3.5.7 DEFINING GAPS IN SESEMS (A SUB-PROCESS)

As part of the SESEMS a detailed design is structured from existing models and simulations. The situation may arise where the selected existing models and simulations do not cover all that is needed to fulfil the simulation requirements. At this stage it is necessary to define where the gaps are between the existing models and simulations so that they can be filled by another process. The define gaps in the 'systems engineering in integration of simulations' process is shown in the Figure 3.10 below. Each stage of the process is detailed in a textual represented see Table 9.6.

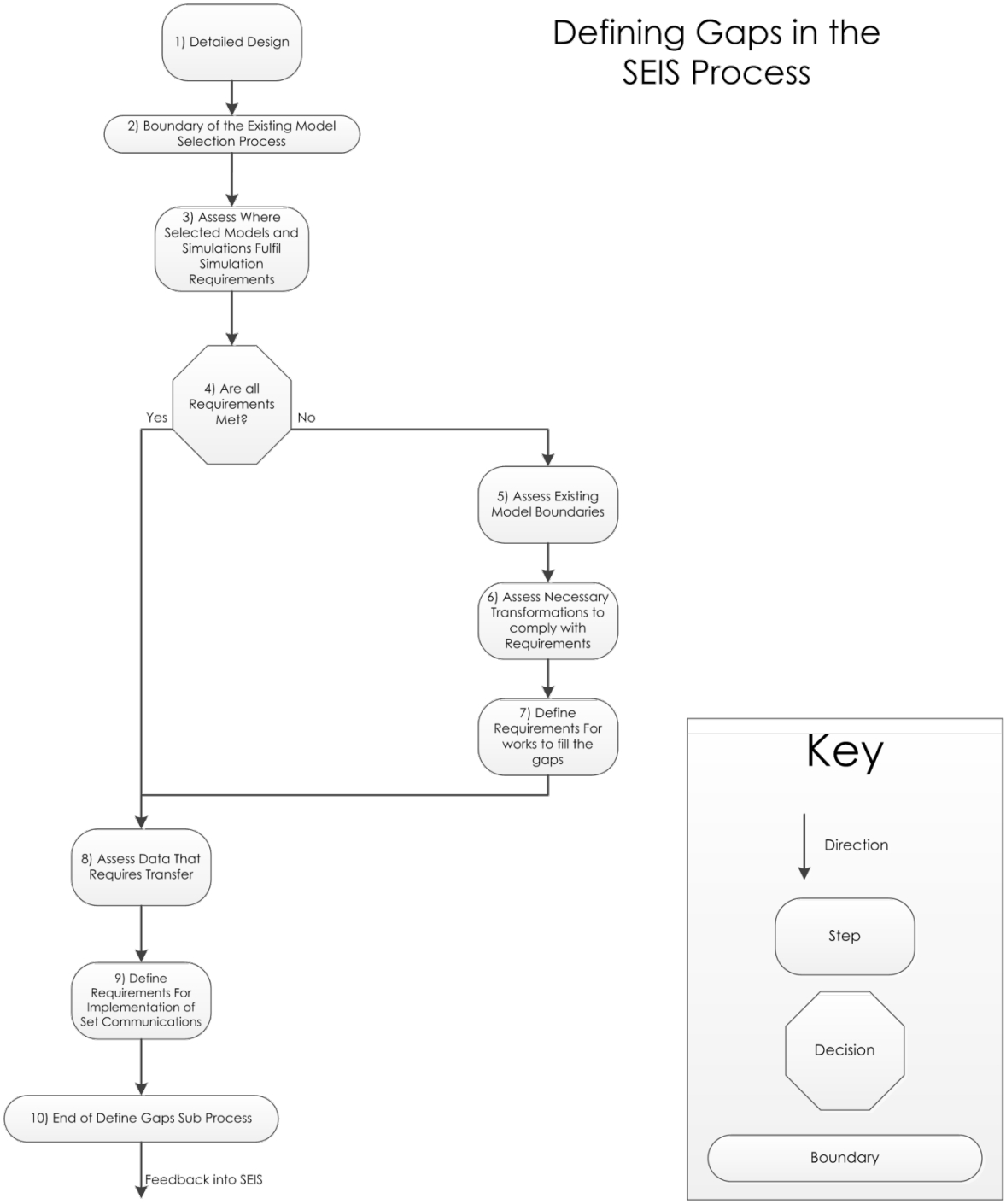

Figure 3.10 Defining Gaps in the Systems Engineering in Integration of Simulations process. The process requires a detailed design and outputs which are used to define the gaps between the available models.

Having a process for defining the gaps gives structure to what can be at first confusing and difficult task. The separation of defining what gaps need to be filled and filling the gaps reduces the likelihood that work is repeated or conducted when not needed. By looking at the gaps it also makes clear the data that needs to be communicated between the models and simulations and the form that the communications will need to take. This process is a way of looking at a constrained problem in a procedural way.

3.5.8 Fill Gaps in the Systems Engineering in Integration of Simulations

Once the gaps between the existing models have been identified the said gaps can be filled with new material. This new material may mean new models, simulations or communications infrastructure. The flow diagram for this process is shown in Figure 3.11. This process is very similar to the overall Systems Engineering in Integration of Simulation process; this is because it is in essence a micro version which is formulated from the same principles.

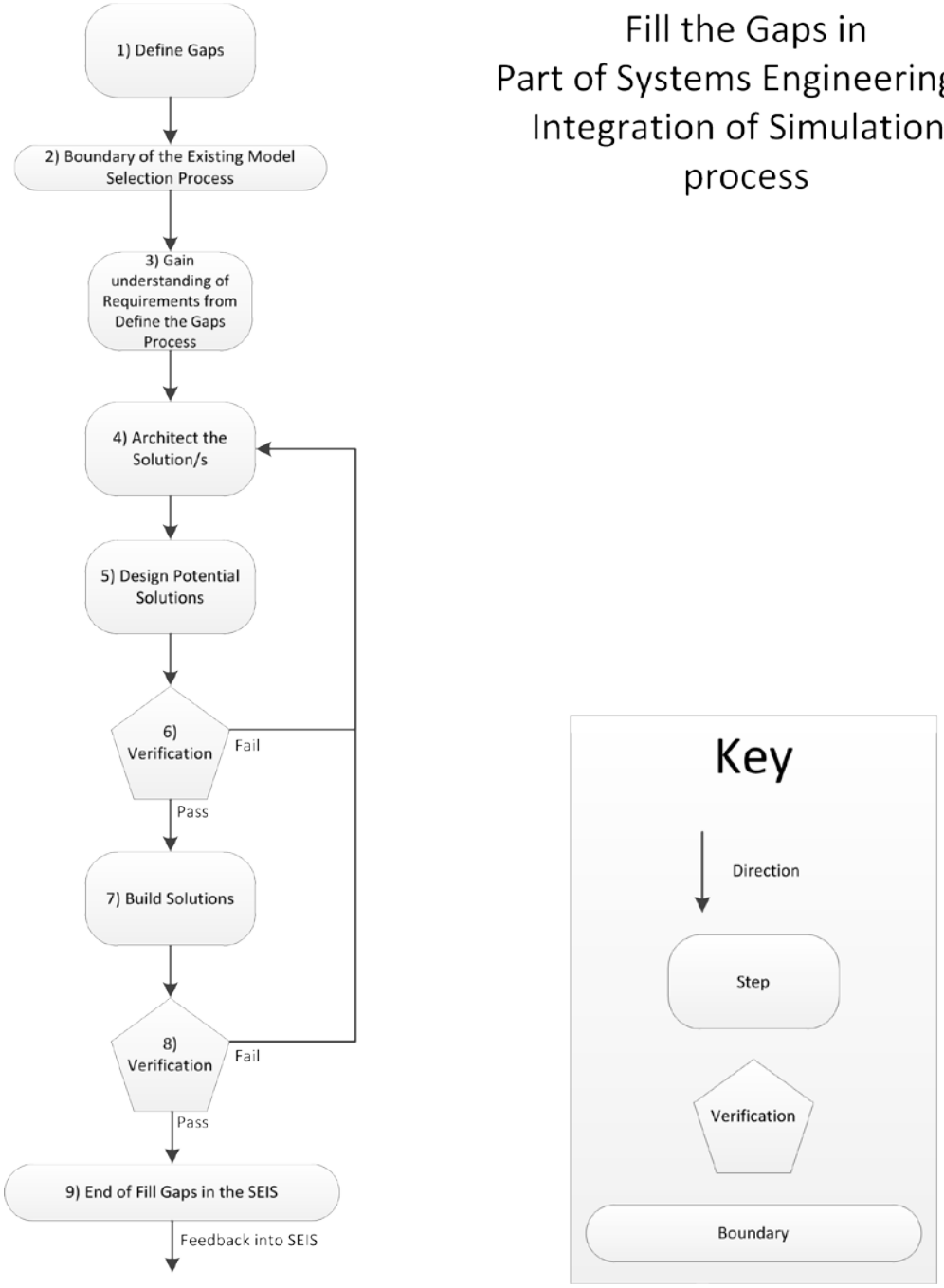

Figure 3.11 Fill the Gaps is part of the Systems Engineering in Integration of Simulations process. These are a design and build process based on the information from the Defining Gaps in Systems Engineering integration of Simulations process.

For further details, the stages of the process which is shown in Figure 3.11 are laid out in section 9.1. For each stage the activities that are to be carried out at that stage are detailed as well as the rationale behind the stated actions.

This process was not included in the main SEIS as it is a contained process that could be executed in many ways. The inputs and outputs are required by the larger process for which this example process was constructed.

3.6 SIMULATION AND MODELLING REQUIREMENT WRITING GUIDE

The intended use and hence the scope of this requirements writing guide is for the production of requirements specifically for tests using models and simulations, hence It is not intended to be a general purpose writing guide. This writing guide takes the aspects of the writing task and breaks it down into activities and characteristics that all of the requirements should have. One of the criticisms of many systems engineering guides is often that they are too long. For this reason each of the sections of this guide are less than one page of text and in total less than 10 pages. The core ideas are expressed with the implementation methods left up to the engineer's best judgment for that situation. For the rest of this section the titles are in bold, the text is *Italicised* and indented whereas the reasoning for that characteristic is given below in non-italic text.

3.6.1 INTRODUCTORY SECTIONS

At the start of the guide there is a section that indicates what a requirement is and how one can be ascertained. This section sets the boundaries of the guide, its purpose, and how it is intended to be used.

Definitions

One of the leading causes for ambiguity and miscommunication is differing definitions of words. For this reason key words that are used in this guide are explicitly stated with definitions.

In the domain of requirements writing the shared consensus of the meaning of words is paramount. For this reason key words that commonly cause ambiguity are defined below.

Model: A representation comprising mathematical functions which is static in nature.

Requirement: A statement detailing what a potential solution is to be capable of achieving, or constraining the solution space which it is to operate within.

Simulation: A simulation is an experiment which uses a dynamic implementation of mathematical models. Simulations produce data sets.

Co-Simulation: The use of two or more simulations that share or exchange values during run time.

Verification: Provides objective evidence that a system or system element fulfils its specified system requirements, architecture, design and behaviour.

Validation: Provides objective evidence that the system, when in use, fulfils its stakeholder requirements, achieving its intended use in its intended operational environment.

With these key words defined users can refer back to this section if they require greater clarity of what is meant by a phrase or word.

Means of Communication

In much of the systems engineering literature the authors spend a considerable amount of time getting caught up in the debates as to which means of communication is to be used rather than how the information is used. To allow for the user to utilise which ever means they feel is the most suitable for the project, this guide does not dictate how to communicate the generated requirements, but will provide the necessary information and characteristics that model requirements should contain. All information within this guide is equally applicable to textual processes as it is to Model Based Systems Engineering (MBSE). This is achieved by focusing on the content of the information that is to be transferred not how this is expressed.

The means of communicating requirements is a contested topic within the systems engineering discipline with strong views held on all sides. This guide was constructed with both textual and MBSE modelling methods in mind. At present neither the textual or MBSE methods are without limitations.

Model Requirements are not System Requirements

The requirements of a system specify the system not the requirements of the model or simulation of the system. The full philosophical reasons as to why this is the case are not discussed, however an abridged version of this information would be a part of any distribution of this guide. A section such as the one below that gives a brief overview is intended to be included.

A simulation of a design is a system in its own right and so requires its own design process. The requirements of the system do not specify how to simulate the intended design and produce tests to verify it. The specification of a simulation system is not a trivial exercise and there are many large scale simulations where this was not realised early on, resulting in much rework.

The conceptualisation, design, implementation, verification and validation of simulation of a complex design are not trivial exercises. Therefore the very systems methods that are being used for the system being designed are as applicable to system of models and simulations emulating it. The use of systems methods also aide in the verification and validation of the models meaning that the models and simulations have potential to hold value after the project has ended.

Moving From Systems Requirements To Modelling Requirements

To produce simulation requirements from a blank sheet of paper can be a difficult task. As a simulation will be mimicking part of the system being designed a logical place to start gaining an understanding of what needs to be done is to look at the requirements of the system being designed. However these need to be complete, concise, and correct.

Before attempting to move from systems requirements to modelling requirements the systems requirements are to be in a state in which the iterations are stable and the complete set have been verified.

For the results of the simulation to be of use the design process needs to be at the detailed design stage and a full potential solution developed.

For a simulation to have meaning and impact on the design process it must test an aspect of the systems behaviour that has been identified as requiring greater investigation. This means the test cases for the system being designed should have been identified and suitable test formulated. These test cases can aid in the development of virtual representations.

Having the test cases also requires a hypothesis as to how the system will function. A hypothesis is needed as this can be used as a first sense check of the output of and resultant simulations.

By having an understanding of the system being designed and the way that it is intended to be tested, before any thought of how the simulation of this system is to operate, ensures that the virtual testing will be reflective of the physical testing that the system will be subjected to during validation.

Verification of Models

The reason for verifying models is to reduce the time spent trying to integrate models that cannot be not verified as being correct.

To ensure that model outputs are 'correct' there is a need to verify the models before relying on their results.

To allow the process of verification and later validation, requirements are need to be specified as to allow for meaningful test cases to be formulated.

If a requirement cannot be tested against in one way or another it is to be reformulated so it can be.

With a robust requirement set and well thought out test cases, meaningful testing can be carried out to ascertain if the requirements for a system have been met or not. Failing to have either a robust set of requirements or well-conceived test case can result in erroneous assertions as to the result of verification testing.

Having the verification of models and simulations allows engineers to check that their work is what it was intended to be.

What is the purpose of the test?
For meaningful investigation into the suitability of a potential design, what is to be tested is to be clearly understood before any attempt to create models or simulations is started.

> *The purpose of the test is the primary drive for the creation of model or simulation. What aspects of the potential design require further investigation and virtual testing?*
>
> *Once an understanding of what is to be tested is gained, the purpose clearly stated, and boundaries of the test to be formulated, the requirements can be written. The statement of boundaries will aid not only in model selection but also to reduce unintentional feature creep.*

By using understanding to constrain the problem that is being worked on ensures that effort is focused on what is absolutely needed, and not exploring areas that are not vital for the development of understanding of the system being designed.

Two Sides of Requirements
To make explicitly clear to the user of the requirements writing guide, requirements are intended to have two distinct uses and both are of value to the project as a whole even if individuals are only involved with the one component.

> *Requirements have two distinct purposes when creating a system both of which are equally as useful and hence important.*
>
> *1) To specify what is wanted from the system being commissioned, be that functionally or operationally*
>
> *2) To function as a reference point so that tests which are formulated to ascertain whether what has been produced fulfils the original intent of the system being commissioned.*
>
> *When requirements are written, the two ways in which it is going to be used needs to be considered. If a requirement cannot be used in both of these ways then it is incomplete and requires rework.*

The dual use of requirements is a key part of the underlying principle of using requirements to guide work on a project. Due to the importance of this singular feature a whole section was devoted to it to signify its importance to any potential users.

3.6.2 Characteristics of Acceptable Modelling Requirements
There are a number of characteristics that all well written requirements have. Each one of the identified characteristics is relied upon in the processes that are proposed in this work. If a requirement fails to have all of these characteristics it can cause considerable problems later on in the systems lifecycle.

The nature of requirements is such that they fall in to two categories: acceptable, in which case they can be implemented successfully; and unacceptable, in which case, if used, success is far from guaranteed.

The characteristics of acceptable requirements are:
Unambiguous and Clear
Concise
Complete
Correct
Traceable
Architectural Compatible
Documented
Tolerant
Realistic.

Attributes of Requirements: Unambiguous & Clear

All requirements are to be written in such a way to reduce ambiguity. By making a requirement unambiguous and clear it ensures that anyone reading it will hold the same ideas as to what information is captured by it as the person who created it. Any jargon is to be explicitly defined within the document. Colloquialisms, contractions, slang, abbreviations and unexplained initialism or acronyms are all discouraged. If possible use the formal syntax of the language being used. If necessary use multiple means to aid in the communication of a concept. It is to be noted that an engineer may only see one requirement in isolation and be asked to create a means by which this is fulfilled without seeing any other requirements.

Attributes of Requirements: Concise

The requirement is short and to the point. Avoid being verbose in descriptions, use only as many words as is necessary to convey the intent of the information while complying with the rest of the acceptable modelling requirements characteristics. However use multiple methods to describe aspects of the requirement if and when necessary.

Attributes of Requirements: Complete

All parts of the requirement are to be in one location and include all information that is needed to understand the specific requirement in question, without need for additional information from other documentation or source.

Each requirement should be in a state where it can be understood without the need to know any other requirements of the system. It has to encapsulate a complete idea be that a function or constraint.

The requirements are all to be complete logical formats be they sentences, models or otherwise.

Before the requirement is implemented it is to be specifically defined as being complete, be that in the requirements documents or otherwise.

Attributes of Requirements: Correct

Every effort is to be made to ensure that all requirements represent the need or constraint in its completeness. Errors in requirements can cause significant problems in the solutions later on downstream. If there is any doubt that a requirement is correct, time and effort is to be invested to ascertain whether it is or not. To increase confidence in those who are using the requirements it can be beneficial to references sources of relevant information which can be included within the requirement documentation.

Verification is an important part of ensuring that the requirement can be defined as correct and so this is another reason why the requirements need to be written in such a way as to allow for verification to be meaningfully conducted.

Attributes of Requirements: Traceability

Each requirement is to be traceable throughout its life. During many requirements capture, manipulation and verification processes it can be all too easy for a requirement to be lost, missed, or duplicated. By having full traceability of each requirement, it can be ensured what has happened to each requirement and which ones have and have not been satisfied by any potential solution.

A means by which requirements can be traced is strongly advised to be used. This guide will not specify a tool or method however the value of using a strict procedure ensures the integrity of the requirements set is ensured.

When requirements are decomposed they should be traceable all the way back to the highest level requirement. Having full traceability also ensures that groups of engineers can talk and work on the same requirements set without issues of all of them working form different sets at the same time.

Attributes of Requirements: Architectural Compatibility

The requirements of models and simulations are not necessary working on a clean sheet but rather piecing together existing models and simulations. If this is the case then there can be issues of ensuring that the requirements for the simulation being designed are architecturally compatible with other requirements. If an architectural standard has been chosen to be used then it may be necessary for the requirements to have a particular format or structure. If this is the case then the standard is to be complied with.

Attributes of Requirements: Documentation

The requirements of a model or simulation form a key part of the overall system documentation. By having a robust set of requirements all other documentation will either be exposed by them or will be built upon them.

Documentation allows not only for verification exercises to be conducted without using the modelling environment but also has the potential to make the model integration process far quicker at a later date.

Attributes of Requirements: Tolerance

Any specific values that are given must have a maximum and minimum. It is advised that no absolute values are to be been used. As in the real world, no absolute measurement of values can be found, therefore a maximum and minimum is to be stated as well as the resolution of the measurement (precision). The resolution allows for the accuracy of the measurements to be determined.

Attributes of Requirements: Realism

All requirements are to be realistic and achievable within the scope of the project. The scope of the project will define the technology, time, cost and other resources available to the project. This is an engineering judgement that will have to take into the consideration the human aspect of the engineering project and the skills that are available in the organisation. All will constrain the feasibility of any potential requirements.

Governing bodies will restrict the solution space and affect the realism of a potential solution and hence need to be taken into consideration.

Procedures and their Implementation

The use of procedures can be somewhat ad hoc which can cause increased variance across the system components being designed. At the requirements writing stage of the project, the level of understanding that should be present in the project team should allow for an informed decision as to if a procedure is needed an what the ramifications of implementation it will be.

There are many well established procedures that can assist in the development of a new system. If a procedure is to be adopted during a project it is advised that it is implemented at the start of the project rather than part way through, as this potentially will reduce the amount of re-work that may be required.

If a procedure is to be used it is to be stated in the requirements as well as any supporting documentation.

Some procedures are recommended wherever it is feasible to do so such as: ensuring that no one person is the sole author and verifier of a

requirement, and a central method of storing all of the requirements is used, and some sort of version tracing implemented.

The implementation of a project-wide process is not one to be taken lightly or too eagerly. If a process is implemented too early in a project it may be done so without sufficient knowledge, and has the potential to cause more harm than good, whereas if the decision to implement a process is made too late then it can be considerable rework to get everything to comply. Incorporating this decision into the requirements process not only ensures that any use of a process is defined explicitly in documentation but critically also ensures that a suitable amount of understanding is captured as to if a process would be of benefit to the project or not.

Model and Simulation Constraints

When considering the generation of requirements for models and simulations it becomes apparent that many constraints need to be captured that are specific to the simulation being designed rather than that of the system being designed. For this reason if these constraints are not investigated before the simulation is designed it can cause significant problems when the simulation being designed is implemented.

When writing model and simulation requirements there are specific constraints that need to be investigated and addressed. These constraints are not applicable to the system being designed.

Model and Simulation constraints that require consideration include:

Simulation Architecture
Total Run Time
Time of Iteration or Step
Hardware Processing
Hardware Memory
Hardware Storage
Specific Use Hardware
Hardware Peripherals
Software.

Model and Simulation Constraints: Simulation Architecture

Using the methods proposed the process of defining an architecture that the simulation is to adhere to and hence its component parts leads to a restriction in the form that the component parts can take. By specifying architecture the complexity of the integration task can be greatly reduced.

Model and Simulation Constraints: Total Run Time

All models and simulations require time to execute. Some simulations may take days to run whereas others take less than a second. Time costs money in the production of a new product. Depending on how vital the

information being modelled or simulated is, it may or may not be worth the investment in time. There is often a trade-off within simulations that the greater the number of data points or accuracy that is needed, the greater the run time. To reduce the chance that models and simulations take a long time to complete, it is beneficial to assign a constraint. The total run time constraint of the entire simulation being designed can be decomposed to all of the component models and simulations.

The value of a total run time is also a trade off with the hardware and software that is available. It may be necessary to readdress the time constraint if there are issues with the capabilities of the available hardware.

Model and Simulation Constraints: Time of Iteration or Step

When integrating models and simulations the time that a component model or simulation requires to complete one iteration or step has a significant impact on the sharing of values between component parts. This concept of time is twofold: the time being represented by the simulation; and, the time that the simulation takes to execute.

Specifying a time that represents the real world is not an arbitrary value that can be plucked at random. If the time value is not the same across all of the models and simulations then issues of sampling and bandwidth (Nyquist theorem) becomes apparent. Having differing real world time steps may be unavoidable if so ensure that ant-aliasing technologies are utilised.

Having a time constraint for delivery of data from the model or simulation in respect to time of computation means that distributed communication may be possible. Knowing how long a model of simulation is going to take to produce a value is valuable when attempting to integrate them together. Defining how long a simulation has to produce a value can be dependent on the hardware and software that is available. It may well be necessary to re-evaluate an achievable time step once such information becomes available.

Model and Simulation Constraints: Hardware Processing

In this context all models and simulations require a computer to process the mathematical representations of the system being designed. The digital processing requires hardware for the model or simulation to run. Different hardware, or even the same hardware configured differently, can have effects on the way to best utilise it and reduce the time that it takes to run a model or simulation.

The way in which code is written for the different hardware and software varies drastically. At the later requirements stage it is advised that the available hardware is specified and decisions made as to what type of processing is going to be used. This will make the later integration far easier.

It is to be kept in mind that it is possible to produce incompatible structures that make model and simulation integration impossible.

Model and Simulation Constraints: Hardware Memory

All models and simulations require memory to operate. In any given hardware set up there is a finite amount of memory available. When specifying an integrated simulation the memory allocation for each component part requires careful consideration.

Model and Simulation Constraints: Hardware Storage

Simulations and models require storage for their own code, any run time engines, as well as the storage for any results. Some simulations produce large quantities of data that require huge amounts of storage. As with memory, storage is a finite resource. Hence its allocation needs careful management. It is advised that maximum limits are set on the amount of storage that any component part can access.

Model and Simulation Constraints: Specific Use Hardware

Some simulations can use specific hardware, such as a Graphics Processing Unit (GPU) or FPGA (Field Programmable Gate Array) to conduct repetitive calculations quicker than on a general purpose CPU. The available hardware therefore has to be considered.

Model and Simulation Constraints: Hardware Peripherals

Some models and simulations use other peripherals to conduct calculations, such as dongles or HMI interfaces. If the simulation being designed is intended to be used in many situations by many different engineers then having required peripherals may not be desirable. If such additional peripherals are not desired then it needs to be written into the requirements.

Model and Simulation Constraints: Software

When writing model requirements software availability is to be taken into consideration. If a new piece of software is needed it should be identified as early as possible. However it is to be noted that if a choice of software has been made for the project at a higher level, it will affect not only the project but also any downstream requirements.

Verification of Requirements

Once a set of requirements has been created they can be verified to ascertain if they comply with the guidance that has been given. The issue of what is to be held as the reference point is dependent if looking to see if the requirement has captured the correct information or if the requirement has been written correctly.

To insure that requirements are acceptable and can be implemented a process of verification is needed. This should take two forms: one checking against the acceptable requirements section; and the second verifying that the requirements reflect what is required from the design of

experiments. By verifying both of these aspects, the way in which the requirements are written as well as against what the intent of the model and simulation is to achieve, reduces the risk of requirements that are erroneous or outside of the scope for the task at hand.

3.6.3 Summary Of The Requirements Writing Guide

It is intended that the requirements writing guide focuses on what information is captured and the validity of the information rather than the means by which it is captured and presented. For all means by which the requirements are captured and presented, the processes and characteristics that are present in this guide will be applicable. This guide is focused around model and simulation requirements using existing models and simulations however the general principles that are used in this guide are equally applicable to simulations being designed from a blank page.

3.7 INFORMATION NEEDS FOR MODEL INTEGRATION

For engineers to be able to integrate models and simulations and ascertain if the integration is meaningful there is information that they will require. For an engineer tasked with integrating a number of models and simulations, they need to have an understanding of the models and simulations as well as an understanding of how to go about integrating them. This section is concerned with what is the minimum amount of information that an engineer needs to ensure that the integration is meaningful, when they have had no previous contact with any of the candidate models or simulations.

From investigation into the types of information that are needed for meaningful integration, reoccurring categories became apparent. These categories form the foundation of the proposed structure. This is not meant to completely remove the engineering judgement in the integration task but rather support it, meaning that pure engineering intuition is not the foundation of the decisions as if two models or simulations are compatible or not. This increases confidence in the resulting integration.

Existing integration standards were investigated as a starting point as to what information has previously been identified as being of use. Out of all standards considered, it was found that at present FMI showed the most promise. For a full review of modelling standards and the way in which they operate see section 2.3.3. Many of the standards work best when they are used during the creation of models and simulations rather than applying them retrospectively to existing models and simulations. FMI is no exception in this case. This is a critical identification as at what point a standard is implemented in the projects developmental time-line has a direct impact on the successfulness of the standards implementation.

3.7.1 THE SOURCE OF INFORMATION FOR MODEL AND SIMULATION INTEGRATION

Most of the relevant integration information can be extracted from model or simulation documentation as well as from the model or simulation itself. This is dependent on the level of detail in the documentation as well as access to the code used to script it. If the documentation of models and simulations is not within the organisational culture then the majority of the relevant information will be in the head of the creators and inaccessible to others. It is most likely that most organisations fall somewhere between the two extremes of too much or too little documentation. This likely means that the relevant information is held in existing models and simulations but it is not in a form that can be readily captured and compared. If such information could be captured and stored in a repository, the repository could be interrogated by various machine learning techniques to find which model or simulations have the highest likelihood of being integrated successfully. This would have further reaching impacts to a business as if an engineer creates a model and it is fed into the repository, checks

could be done to ensure that all of the necessary information is present in the repository. This checking would result in a company still retaining the value of a model or simulation after those who were involved in their creation have moved on.

3.7.2 The Two Sides of Model and Simulation Integration

There are two distinct sides of a model or simulation, the first being the logic and equations that represent the phenomena being mimicked, and the second being the infrastructure that is needed to execute these models and simulations. Both sides are needed to be understood as there are many reasons why either side could cause an inherent incompatibility or integration challenge that has to be overcome. Hence any potential solution for how to capture the necessary information has to reflect both sides of the model or simulation.

Using this perspective combined with information captured in section 2, with first-hand experience, a collection of all of the required information was compiled and sorted, see the mind map Figure 3.12 below. The level of detail is such that an engineer with a working knowledge of modelling and simulation could use this information to discern if two or more models or simulations have the potential to be integrated successfully and in a meaningful way.

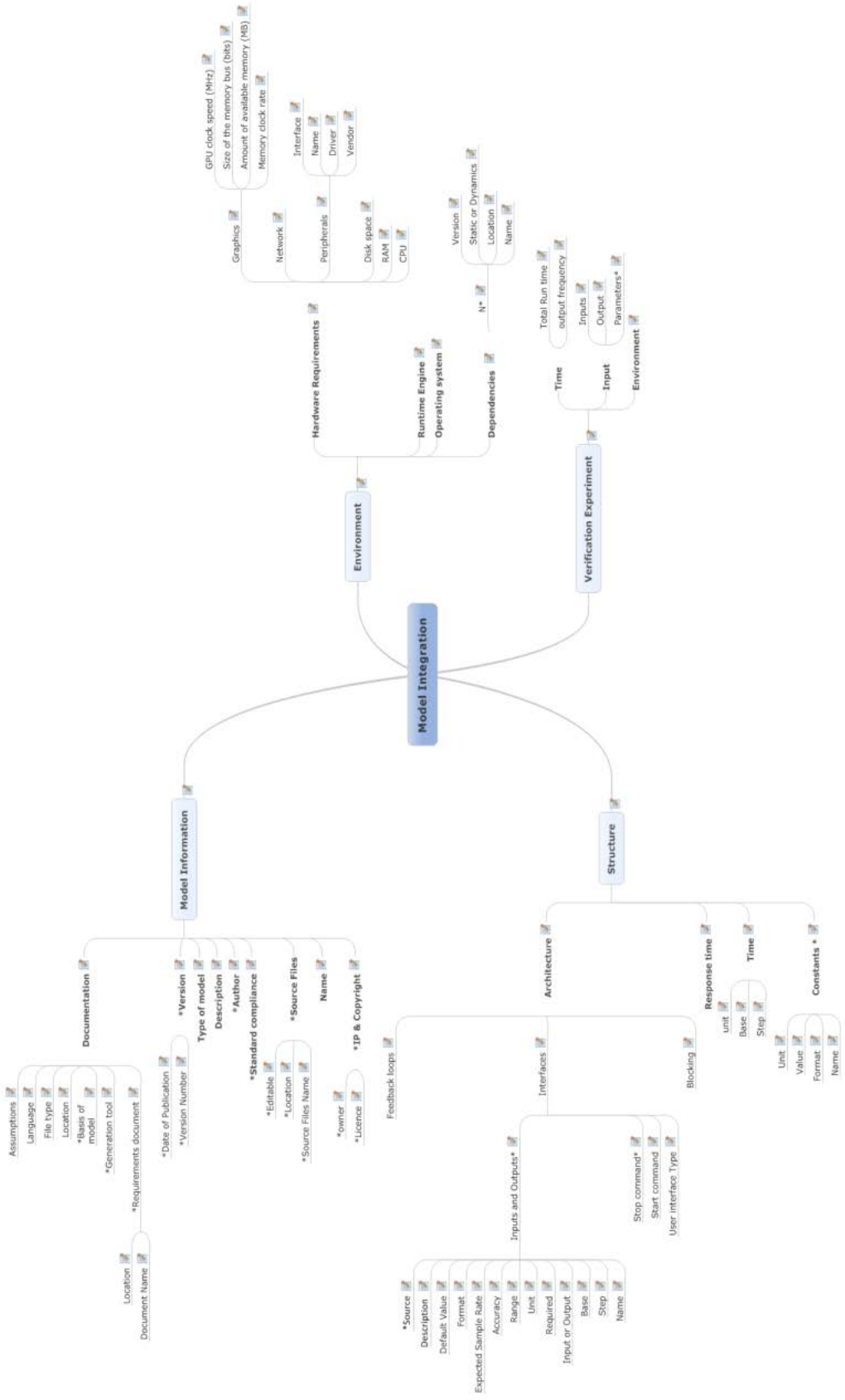

Figure 3.12 A mind map of the required topics of information for meaningful integration. The structure of the required information became apparent once the required information was compiled.

The Environment branch covers the necessary infrastructure that the model or simulation requires to execute. The Structure of the model covers the implementation of the scientific principles. The verification experiments as introduced in section 3.5.5 is captured in the Verification Experiment branch. The Model information covers the description of what is the phenomenon being replicated is and what has been considered and used in its creation. This information branch also covers aspects of intellectual property and distribution agreements.

3.7.3 INTEGRATION TABLES

To capture the information that has been found to be necessary for meaningful integration, a means by which this information could be captured and integrated is needed. To make comparisons between the integration information, the information needs to be in a repeatable form. It is proposed that the information is recorded in a tabular format. The four proposed tables below take the form of the structure that became apparent during the elicitation of the required information; Model Information, Structure, Model Environment, and Verification Experiment. The intended layout of the tables is relatively simple see the Figure 3.13 below.

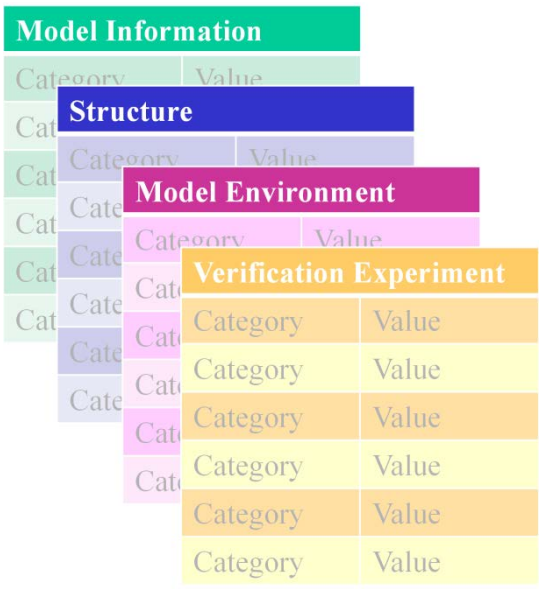

Figure 3.13 The intended layout of the integration tables. The category is defined by the information in the previous section and the value relates to the model or simulation that is being evaluated.

It is intended that the values that are entered into these tables will be in form that is most suitable for that category. Therefore most of the values will be in natural language. By using natural language it allows for flexibility to enable the effective capture of the relevant information. However some are in numeric or Boolean format dependent on the most appropriate data type to capture that information.

By splitting the necessary information into four tables makes navigation far easier than if it was in a single monolithic table containing all of the information. When

filling out the tables having similar groupings of information means that often when looking for one piece of information, another piece becomes apparent and can be captured at the same time.

Instantiating all four of the tables for every model or simulation being considered is not necessary or an efficient use of time. If it is a first sweep across candidate models or simulations it is well worth just filling in the model information table. By filling out the model information table the engineer will capture the fundamentals of the model which will give the semantic reasons why or why not the candidate model or simulation could be used. The structure and modal environment tables are proposed to be used when conducting the integration of the models and simulations. The integration issues that will occur from the differences in the structure and model environment are more likely to be possible to overcome using established engineering methods.

By having the information in a defined structure makes comparisons between the information sets of the two or more models easier to compare and contrast. Organising information in this way potentially allows for various stages of automation. For instance a GUI could be developed that sits at the front end of a database that stores all of the information. With such a database the data can be queried either manually or automatically. Also patterns trends and potential suitability of integration scores could be formulated between the captured models and simulations.

The biggest drawback of using these integration tables is that they can take a considerable amount of time for an engineer to complete. If a level of automation was involved in the process to mine the necessary data from the existing documentation and requirements, the majorly of the information could be captured. Once a suitable amount of data had been captured potentially decisions could also be automated to inform whether the models or simulations are suitable to be integrated.

Having the data in this form also opens up the potential for machine learning to scan over the information and conduct analysis on it. This potential was recognised and explored, see section 5 for more information.

3.8 LEVELS OF ABSTRACTION

It was found during the literature review that the level of abstraction of models and simulation has been identified as being a critical piece of information when assessing the suitability of potential integration. However no way of defining abstraction was found. Rather it is a loosely defined concept that many use but few define. Hence in this research the levels of abstraction are considered to not be a fixed static scale that any model or simulation can be measured against but rather a relative term used to compare two or more models or simulations against each other.

One of the most significant problems when attempting to gain an understanding of a model or simulation is to understand the relationships between the variables and how they relate to other systems and subsystems. When large scale interconnected model integration is intended, architecture is necessary to tame the ensuing complexity and bring a structured order to it. If a hierarchy of the intended system is constructed using decomposition theory a natural order of hierarchy of system, sub-system and elemental units occurs. The models will then have their respective inputs and outputs which in turn have their own variables.

To display how the decomposition of a system using sub-systems can result in a greater variation see Figure 3.14 below. This Figure 3.14 depicts the resulting decomposition situation in its simplest form, each model is considered to only have one input and one output. Each variable is describing a concept; each decomposition describes this concept in more detail. For example speed can be decomposed to starting velocity and end velocity; both relate to movement but the latter contains more information. Hence this links the models and their variables to a level of abstraction.

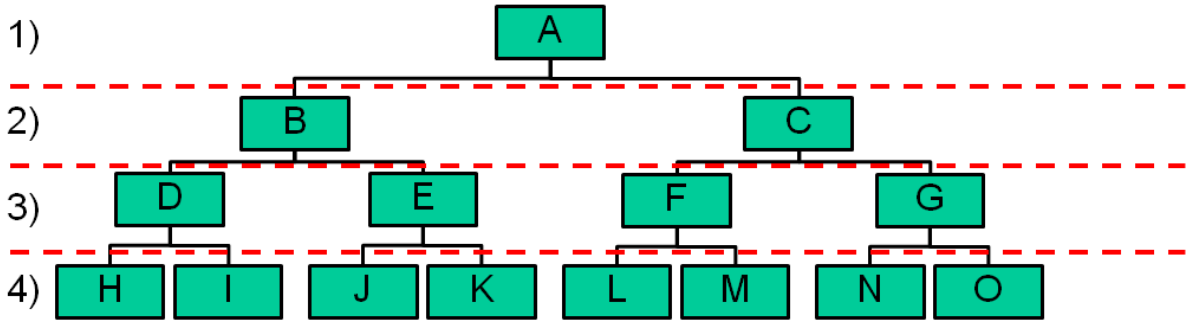

Figure 3.14 Hierarchical decomposition displaying variables and relations between levels. There are four levels of abstraction that have been identified. Level one is the highest level of abstraction and level four is the lowest level of abstraction. The green boxes represent an input or output. The lines between the boxes represent the existence of a relationship.

With this representation it becomes apparent that as the level of abstraction increases, so too does the number of inputs and outputs. Hence there is an increase in the number of internal variables. The 'new' variables have properties that are not captured by the higher levels of abstraction. If this process was continued the final level would be at the very limit of our understanding of the

phenomenon being mimicked which would render any further modelling or simulation futile.

Identifying the levels and pattern of decomposition behaviour is not useful if there is no way of communicating the results. Hence a way to capturingt6 and documenting the relations between variables has also resulted in a textural representation based on the concept of last shared common relative. Using Figure 3.14 as an example the textural representation of the identified relationships is shown in numerical representation.

This textual representation works by mapping the decomposition hierarchical tree from left right, the standard case number relates to the number of variables that are in that section of the tree; a downward arrow represents the jump to the next level. Phi with a numbered subscript, n, indicates that a common ancestor exists (Ψ) with the number of levels needed (subscript, n) to reach it.

[Numerical representation]:
$1\downarrow 2\downarrow 2\Psi_2\ 2\downarrow 2\Psi_2 2\Psi_3 2\Psi_2 2$

This textual representation can be further detailed to capture the names of the variables shown in textual representation.

[Textual representation]:
$1_{(A)}\downarrow 2_{(B,C)}\downarrow 2_{(D,E)}\Psi_2\ 2_{(F,G)}\downarrow 2_{(H,I)}\Psi_2 2_{(J,K)}\Psi_3 2_{(L,M)}\Psi_2 2_{(N,O)}$

To further elaborate how the textual and numerical representations map to the hierarchical decomposition, Table 3.1 breaks down the numerical and textual representations of the system expressed in Figure 3.16.

HIERARCHY LEVEL	NUMERICAL REPRESENTATION	TEXTUAL REPRESENTATION
1	$1\downarrow$	$1_{(A)}\downarrow$
2	$2\downarrow$	$2_{(B,C)}\downarrow$
3	$2\Psi_2\ 2\downarrow$	$2_{(D,E)}\Psi_2\ 2_{(F,G)}\downarrow$
4	$2\Psi_2 2\Psi_3 2\Psi_2 2$	$2_{(H,I)}\Psi_2 2_{(J,K)}\Psi_3 2_{(L,M)}\Psi_2 2_{(N,O)}$

Table 3.1 Hierarchical mapping of numerical and textual representations.

To demonstrate this method further, an example of a ball bouncing between two surfaces has been used. A representation of this can be seen in Figure 3.15 below.

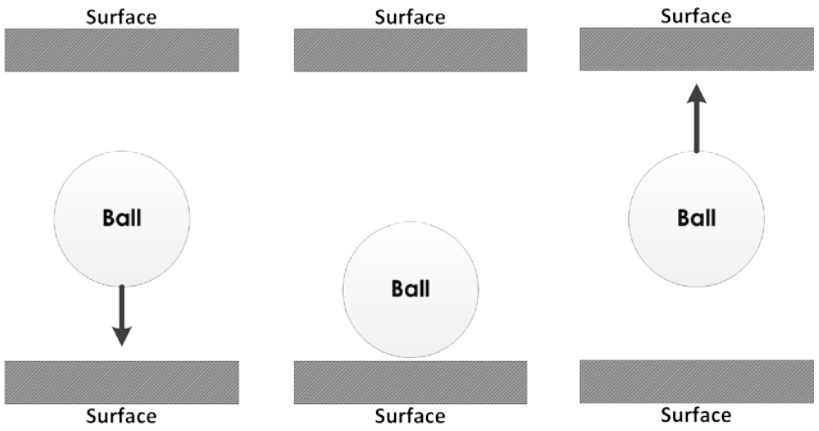

Figure 3.15: A simple representation of a ball bouncing between two surfaces the arrows show the direction that the ball is moving.

This example shows how the hierarchical decomposition of a series of integrated models can be represented using the proposed method. The bouncing ball has been broken down into four levels: textual description; a simple equation whose product is speed; Newtonian equations of motion; and, the energy transfer between ball and each surface. The equations that are used to model this representation can be seen in Table 3.1 below.

ABSTRACTION LEVEL	EQUATIONS
1	Textual description
2	$S = \frac{D}{T}$
3	$u_e(x) = \frac{1}{2} \times k \times x^2$ $v^2 = u^2 + 2 \times a \times s$
4	$K_e = \frac{1}{2} \times m \times v^2$ $u_e(x) = \frac{1}{2} \times k \times x^2$ $v^2 = u^2 + 2 \times a \times s$ $S = u \times t + \frac{1}{2} \times a \times t^2$

Table 3.2 The Abstraction Level of a ball bouncing between two surfaces and the relevant Equations which are appropriate to that level.

The equations used to model the situation shown in Table 3.1 is also represented in levels of abstraction in Figure 3.16 below.

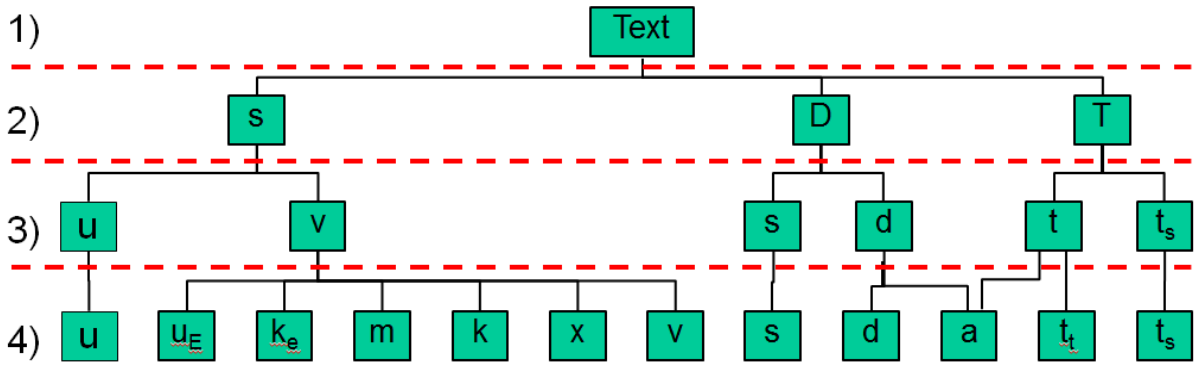

Figure 3.16 Hierarchical decompression of levels of fidelity of a bouncing ball. Four levels of abstraction, the variables that are considered and the relationships between the variables.

Developed textual representation:

[Textural representation 3]:

$1_{(text)}\downarrow 3_{(S,D,T)}\downarrow 2_{(U,V)}\Psi_2 2_{(S,d)}\Psi_2 2_{(t,ts)}\downarrow 1_{(U)}\Psi_2 6_{(Ue,Ke,m,k,x,v)}\Psi_3 1_{(s)}\Psi_2 2_{(d,a)}\Psi_1(1_{(a)})\Psi_1 2_{(a,tt)}\Psi_2 1_{(ts)}$

This analysis highlights how the semantics behind the variables increases in complexity as the levels of fidelity increase. A result of this is that it shows where there is the possibility to pass information between levels of fidelity for example using the value of the model used to calculate V on level three and the six variables used to calculate the same concept on level four.

An interesting output of these analyses is that level of abstraction is captured in a form that can be of use during the integration task. This concept of abstraction is a comparative metric resulting from the produced hierarchy rather than an inherent characteristic of the model itself in isolation. This abstraction metric can aid in the integration process as it enables models from a repository to be compared while still having the flexibility to handle a multitude of modelling realisations. The dangers of taking the variables on face value are exposed as it shows how variables can be at different levels of abstraction, have the same name, data type, units and still represent different information, hence showing the dangers of manual integration based on such limited information. The decomposition demonstrates how engineers use the increase in variables and the interactions between them to model in more detail the same phenomenon at lower levels of abstraction. Being able to refer to specific levels of abstraction as a comparative measure across an integration problem space has benefits to the integration process. This method is of assistance primarily as a means of organising variables and reducing the ambiguity of communication for reference between models and simulations.

The decomposition of variables method brings structure to the integration process without the rigidity that a strict standard enforces. This method also displays all of the variables that are open which facilitates relations to be identified enabling lines of communication to be established. However potentially the most valuable output of this analysis is that it ensures that all

variables (and what they represent) is explicitly known by those conducting the analysis.

One of the identified flaws with this method of navigating and recording the variable space is that it requires an intimate knowledge of the models and simulations. However it has been argued previously that this understanding is needed regardless of the methods which are applied. In a well-documented model or simulation the nature of the variables is defined in the documentation within the simulations and in which case an intimate working knowledge may gained by an individual who was not present during the creation of the model or simulation. This method by is time consuming and would greatly benefit from automation.

3.9 SUMMARY OF SYSTEMS ENGINEERING FRAMEWORK AND PROCESSES

The systems engineering framework was expressed with a series of diagrams showing the progression from a linear implementation of systems engineering concepts to a more holistic approach. The resultant Systems Engineering Framework forms the basis of the proposed processes. This process is decomposed into seven sub-processes all of which are expressed with a flow chart each with accompanying textual descriptions of the process elements.

As part of the proposed process there are integration tables that contain information that is needed to be known about the potential model or simulation components before meaningful integration can be verified. These tables and the tasks involved in completing them is supported by the work conducted in section 5.

A means of comparing abstraction layers using decomposition has been introduced as well as a means of communicating the relationships between assessed models and simulations.

4 CASE STUDIES

4.1 Introduction and the Purpose of Case Studies

When new processes and methods are proposed, examples of how they are used are beneficial to not only demonstrate how they work, but also as a form of validation that they work in practice. In this study, two case studies are presented: the first being one that was used through the development of the methods and formation of the overall process, which is referred to as the developmental case study; the second case study comes from the automotive domain, and is intended to mimic a situation in industry.

The developmental case study is focused on the development of a new squash ball. The study of the movement of squash balls has been the focus of many computer science based case studies. The reason for this is that it is possible to align such a subject to a required level of complexity, to demonstrate a given specific technology. In this case study the complexity has been set relatively low to ensure that the focus of the reader is on the process, not on the intricacies of the specific integration task.

Conducting the case studies is intended as a means of validation as seen in Figure 3.7. Critically this involves testing the system in its intended operational environment. In this case the process will be used as it is intended in industry, hence the second case study is focused around the development of a combined steering and braking control system for a sport utility vehicle. It is feasible that an automotive company would wish to conduct such analysis, and have a repository of existing models and simulations of the individual component parts. These component part models and simulations would also have likely been made in complete isolation to each other. The first iteration of this automotive case study used real world models from an automotive company, as well as in depth material from third party COTS manufacturers. However, due to external constraints to this research, as well as the commercial sensitivity of the models, they could not be published. For this reason publicly available alternatives have been used to illustrate the work that was conducted.

4.1.1 Identified bias

Within any test, sources of bias can affect the validity of results. Within these case studies there is one identified bias that cannot go unmentioned; the individual testing the processes and methods is also the one who created them. This is not only a potential conflict of interest that can affect the evaluation of the proposed methods but also the manner in which they are conducted. To combat the potential for deviation from the proposed processes each stage and the work conducted is explicitly stated for later analysis. This transparency will allow assessment of deviation from the processes if any occurs.

4.1.2 Display of Process elements

The two case studies development of a squash ball section 4.2 and integration of steering and braking 4.3 are implementations of the methods proposed in

section 3. The process elements are represented by numbered headings 4.2.1 to 4.2.32 for the squash ball, and 4.3.1 to 4.3.21 for the brake and steering integration. All sub proses elements have been given unnumbered headings.

4.2 DEVELOPMENTAL CASE STUDY: SQUASH

The purpose of the development case study is to step through each stage of the proposed process. The application that has been chosen is the development of a new squash ball. This will involve a simulation of the movement of a ball around a squash court. This is a constrained problem that can be as complex as necessary to ensure that all facets of the proposed methods are tested, while still being relatable to non-domain experts. The purpose of this case study is to verify that the proposed methods are suitable to be used in the assistance of model and simulation integration of existing component parts.

The full SEIS process (section 3.5.5) is intended to be used when there are: customer wants, system requirements, system architecture, system design, and a desire to virtually test an aspect of the design. For this case study the focus is primarily on the virtual testing of a design by integrating existing models and simulations rather than the design itself. For this reason only the most relevant information relating to the architecture and design of the squash ball is given.

4.2.1 CUSTOMER WANTS

In this case study a manufacturer of squash balls wishes to develop a 'training' ball made out of a different rubber compounds. As a consequence it is known that the 'training' ball will have different behaviour when in play. The parts of interest are the energy transfer between the ball and walls, the velocity at which the ball travels, and the time it takes for the ball to move across sections of the court. This behaviour is expressed in Figure 4.1 below.

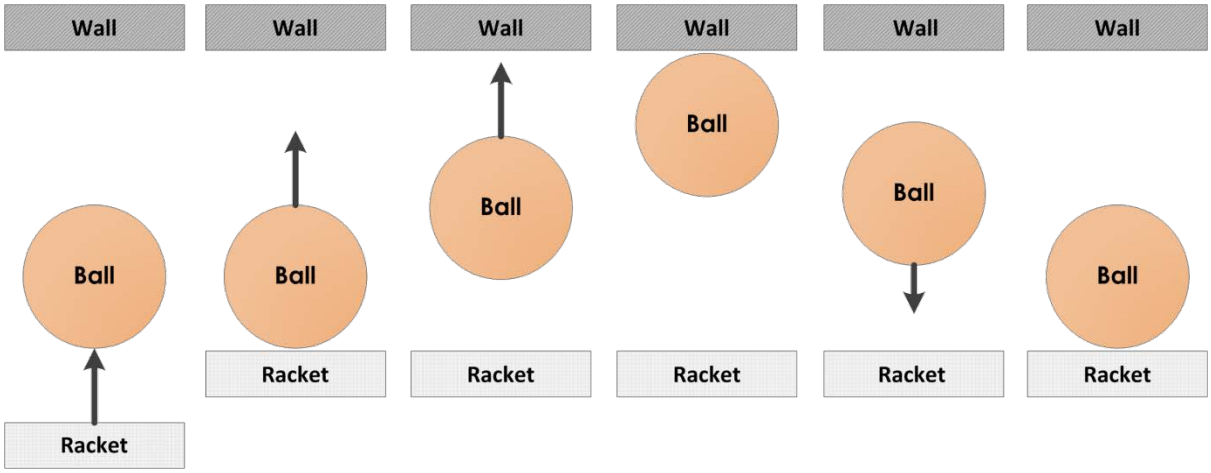

Figure 4.1The behavioural states of the squash ball that are of interest for the evaluation of the design.

4.2.2 System Requirements

There are requirements that are specified by the World Squash Federation (WSF) for balls and courts which are openly publicised and available [87], [88].

Requirements for a New Training Ball

1. The behaviour of the ball at 296.15 kelvin, in the court is to be between 5 ms^{-1} and 10 ms^{-1} faster after rebound from a service speed of 68 ms^{-1} than a regulation competition ball as dictated by WSF.

2. The external diameter, overall weight, and seam strength (if applicable) are to comply with WSF standards.

3. The production costs for a run of 100,000 balls including packaging is to be less than £0.25 per ball.

4.2.3 System Architecture

The architecture of the system that is being designed is that it is a hollow ball formed from various compounds of an undisclosed thickness.

4.2.4 System Design

The design of the ball is of regulation size. The compounds used to construct the ball are to be established through virtual and physical testing.

4.2.5 Systems Engineering In Model Integration

There are prerequisites for the SEIMI process; understanding the problem space, requirements specification of system, architecture of system being designed, and system design, are covered in sections 4.2.1, 0, and 4.2.4 respectively. The transition across the design and simulation boundary which is identified as part of the SEIMI process can now be crossed as all prerequisites have been satisfied.

4.2.6 SEIMI Desire to Test Potential Design

The desire to test a particular aspect has been identified. The desire is to ascertain what the effects of different compounds have on the behaviour of the ball relative to a squash court. It is intended that the outcomes would have direct impact on the material choice for a new test squash ball.

4.2.7 Design of Experiment

The statistical aspect of the design of experiment is not overly relevant in this instance. There are potential compounds that are being concerned for the ball. These are to be tested and the suitability decided upon.

Purpose of the Test

The purpose is to ascertain what are the effects on the behaviour of a squash ball moving around a court when the compounds that it is constructed from are varied. This has been expressed in Figure 4.2 below. The boundaries between the states that the ball is in are proposed to be the areas of greatest interest.

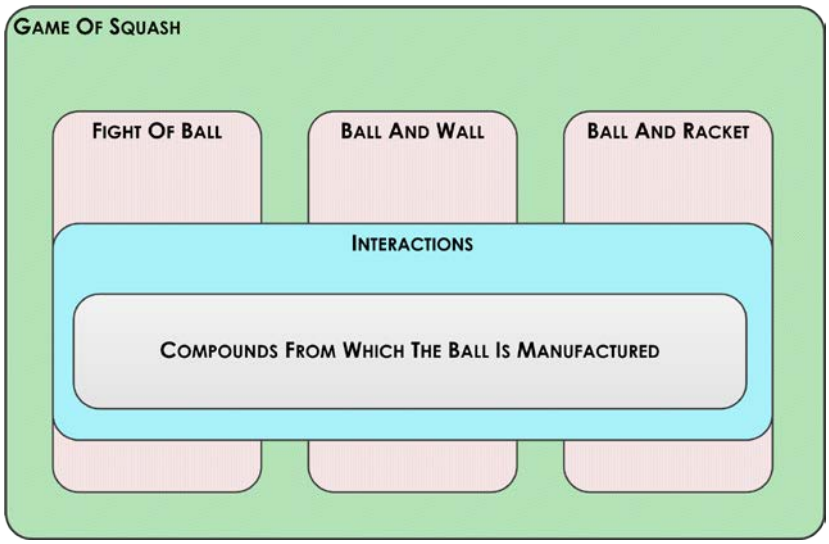

Figure 4.2 Visual representation of the purpose of the test

Hypothesis Which is to be Tested

That changing the compounds that a squash ball is constructed from affects the speed that the ball will return from a wall when hit against it.

Identified Variables

The factors that could affect the behaviour of the ball were explored using the understanding of the situation, and captured in a structured brain storm as shown in Figure 4.3 Brain storm of the potential parts of the behaviour of a squash ball and its interactions.

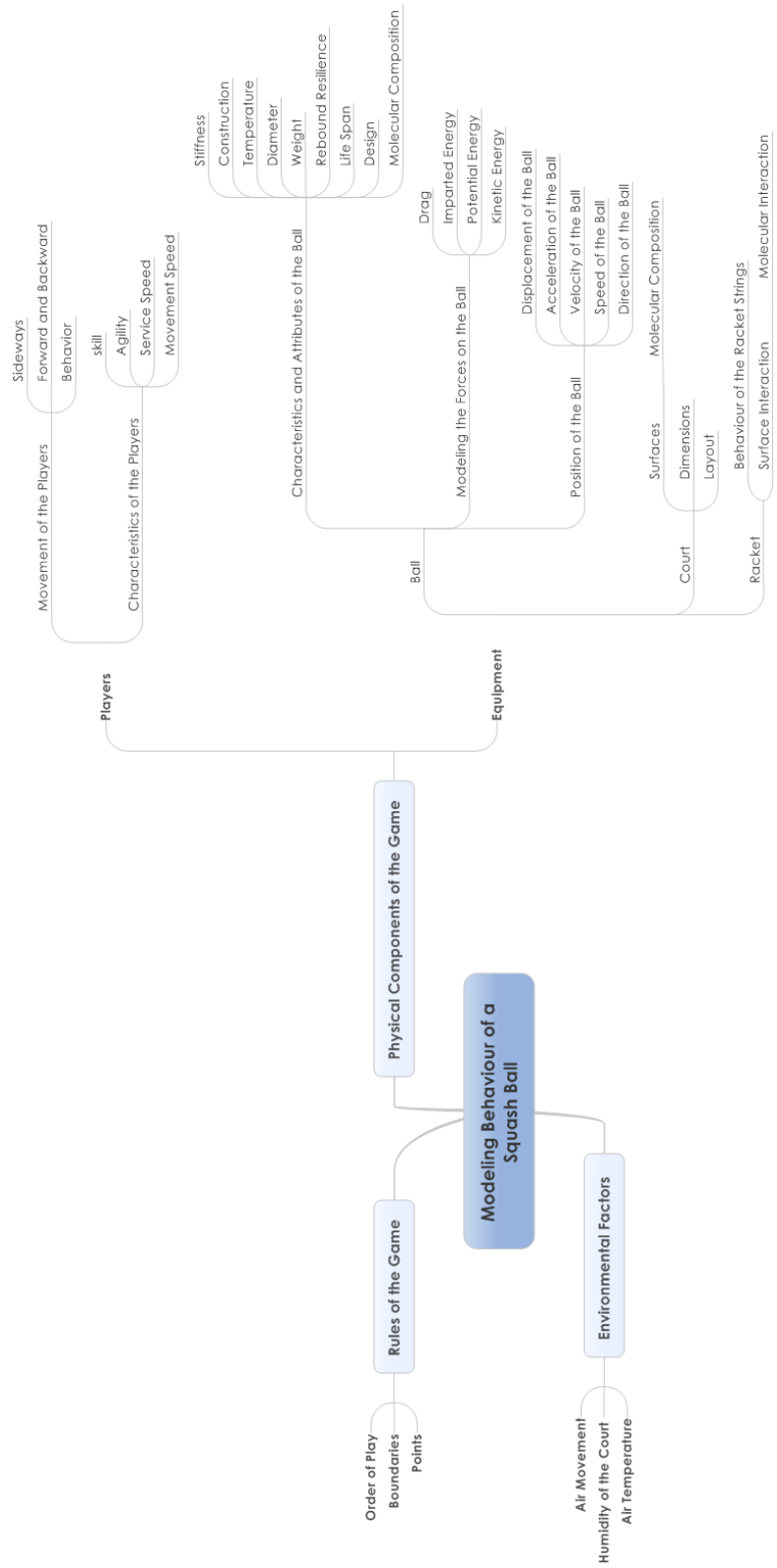

Figure 4.3 Brain storm of the potential parts of the behaviour of a squash ball and its interactions.

With the variables identified there are some that can be held constant and others that will need to be changed to represent the physical phenomena.

The variables that are considered to be held constant:

- Humidity of the air in the court
- Air movement
- Air temperature
- The: skill, agility, service speed, and movement speed of the players
- Diameter of the ball (ball deforms when struck or meets wall)
- Drag coefficient of the ball
- The racket that is used including the behaviour of the racket strings
- Size and lay out of the court
- Rules of the game including; order of play, boundaries, and points.

The variables that may be changed during the experiment:

- Service speed of the ball
- Characteristics of the ball that can change include; stiffness, construction, temperature, weight, rebound resilience, life span, design, and molecular composition
- Imparted, potential, and kinetic energy of the ball
- Velocity, speed, and direction of the ball
- Surface interaction between the racket and the ball

4.2.8 Define Assumptions of Experimental Set Up

There are assumptions that have been made that are to be the same throughout the component models that are either selected or created, as follows.

- The base experiment will hold all elements constant with the only change being the material the ball is made from
- While the ball is moving, it is considered to be a particle
- There is an energy transfer when the ball interacts with each surface.

4.2.9 DEFINE SIMULATION BOUNDARIES

The boundary of what is to be considered as part of the simulation has been captured in Figure 4.4 and throughout below.

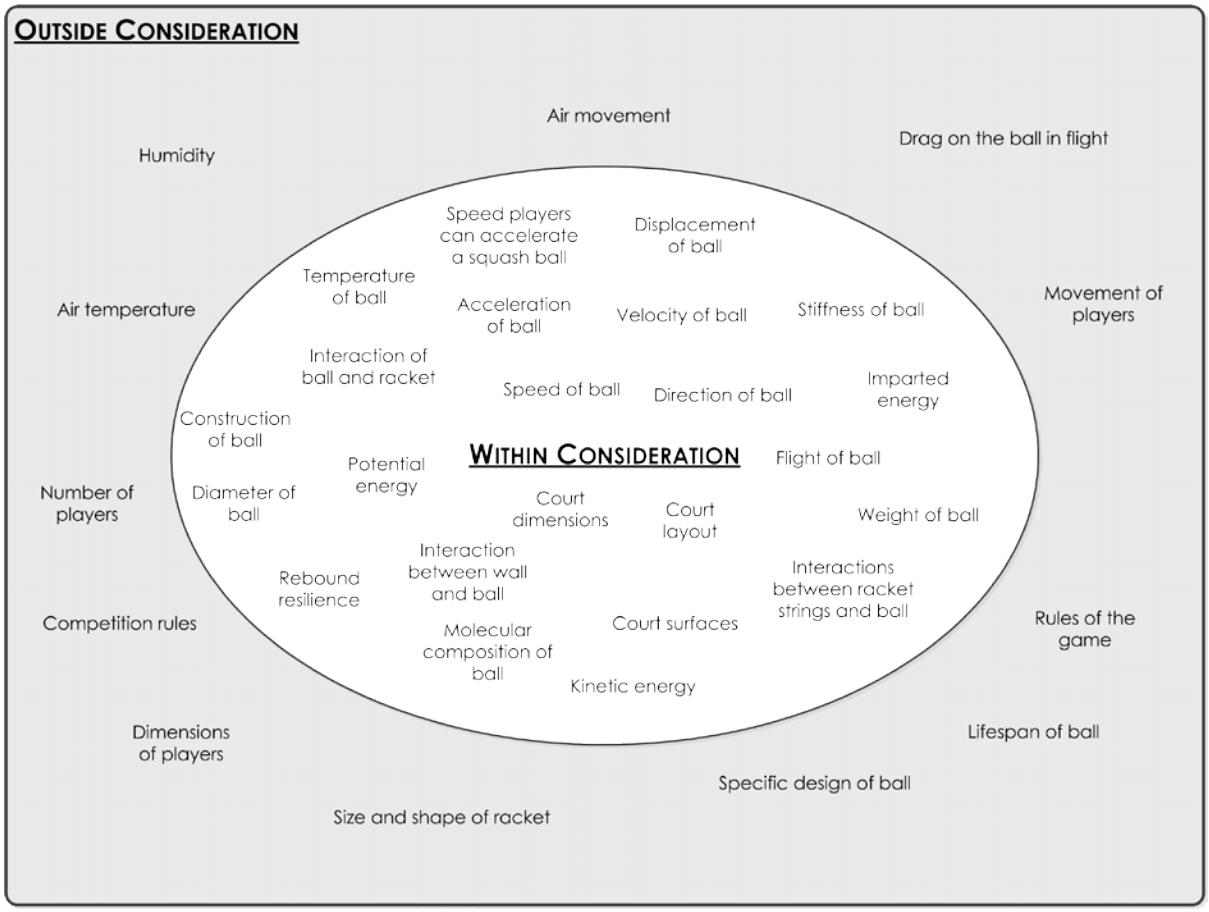

Figure 4.4 Simulation boundaries showing what is within and outside of consideration.

All work is to focus around what is captured within the ellipse in Figure 4.4. All of the factors that have been identified as being outside of consideration can be disregarded or investigated at a later date in a subsequent simulation.

4.2.10 SIMULATION REQUIREMENTS

The simulation and modelling requirement writing guide as defined in section 3.4 was used to create the requirements for the simulation. Due to the system requirements not being sufficient to express the simulation experiment, simulation specific requirements were needed.

Means of Communication

To not distract from the process itself textual requirements have been used in this application as the means by which requirements are captured and communicated.

Requirements

1. *The simulation is to capture the behaviour of a squash ball in flight.*
2. *The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.*
3. *The simulation is to capture the behaviour of a ball interacting with a player's racket.*
4. *The simulation needs to be run multiple times with the only change being the compound of the ball.*
5. *The flight of the ball is constrained by a regulation sized squash court.*

Simulation Constraints

A. The total run time of the simulation should take less than five minutes to fully execute.

B. The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of RAM, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.

C. No specific computational hardware or peripherals are to be used.

D. The modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.

E. The resolution of the time steps across the models is to be at 0.001 seconds.

F. The output results of the simulation are to be saved in a file format that can be interrogated at a later date.

4.2.11 SET STANDARDS IF THEY ARE TO BE USED

An assessment has been made as to if there are any standards that would be of use in this application. It was decided that due to the simplicity of the phenomenon that is to be simulated the use of strict standards would be time consuming for the potential gains that the use of a standard would give in this situation.

4.2.12 VERIFICATION

This is the verification of the requirements as part of the requirements writing stage and is not to be confused with a verification stage of the systems engineering in 'Integration of Simulations' process.

Verification of Requirements

A simple Red Amber Green (RAG) method was selected as the means of assessing the requirements against the requirements writing guide. Within the analysis red defines that the requirement is in direct contradiction and requires logical rework, amber defines that it requires some rework, green defines that it is complicit with the requirements writing guide.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a squash ball in flight.			✓
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.			✓
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.			✓
4) The simulation needs to be run multiple times with the only change being the compound of the ball.			✓
5) The flight of the ball is constrained by a regulation sized squash court.			✓
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) The resolution of the time steps across the models is to be at 0.001 seconds.			✓
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓

Table 4.1 Verification RAG assessment of simulation requirements.

With all of the requirements passing the RAG validation the work can continue onto the next stage.

Verification of Standards

Due to the lack of application of a standard the verification stage concerned with the verification of the choice of a standard can be omitted.

4.2.13 PRELIMINARY ARCHITECTURE

The preliminary architecture for the simulation is depicted in Figure 4.5 below. The architecture breaks down the interactions that have been identified as single components that all operate within a single development environment. The user defined inputs and results are likely to be native functions of the development environment.

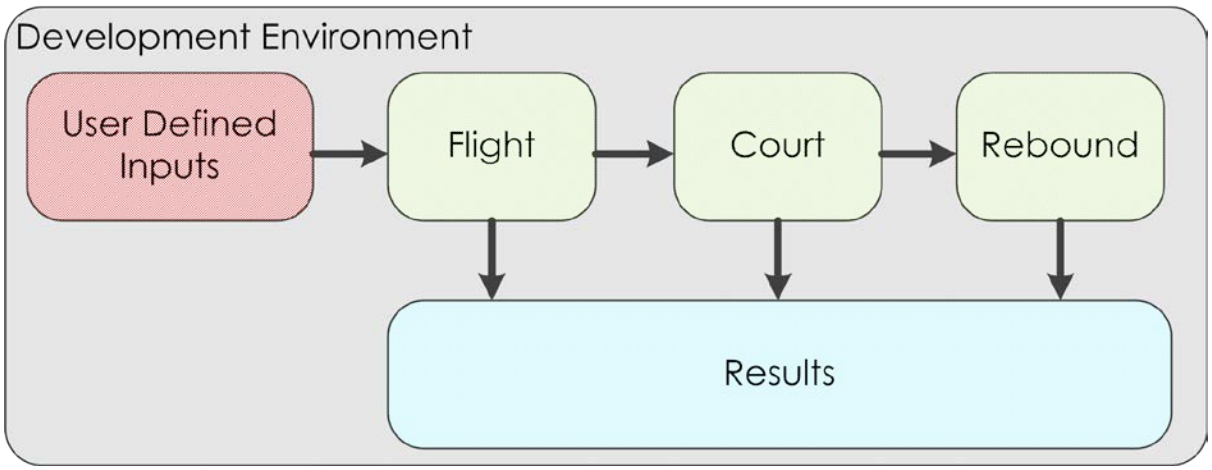

Figure 4.5 Preliminary architecture for the simulation of squash ball flight

This architecture is a simple one that reflects a linear perspective of the phenomena that is to be mimicked.

4.2.14 VERIFICATION OF PRELIMINARY ARCHITECTURE

With a preliminary architecture defined it can be verified against the requirements for the simulation.

A simple Red Amber Green (RAG) method was selected as the means of assessing the architecture against the simulation requirements. Within the analysis red defines that the architecture is in direct contradiction and requires logical re-work, amber defines that it requires some rework, green defines that it is complicit with the requirements.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a squash ball in flight.			✓
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.			✓
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.			✓
4) The simulation needs to be run multiple times with the only change being the compound of the ball.			✓
5) The flight of the ball is constrained by a regulation sized squash court.			✓
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) The resolution of the time steps across the models is to be at 0.001 seconds.			✓
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓

Table 4.2: Verification RAG assessment of the proposed architecture.

With the architecture being found to be complicit with the simulation requirements, a design can be formulated.

4.2.15 PRELIMINARY SIMULATION DESIGN

The behaviour of the parts defined by the architecture can now be broadly defined.

User Defined Input

A GUI (Graphical User Interface) is to be used whereby the user can input parameters and view the results of the simulation. This is to be a single input interface for all of the component parts.

Flight

The flight of the ball through the air and the time that it takes to do so when a particular force is imparted upon it.

Court

The size and shape of a squash court are to be captured as well as the relative position of the ball within the three dimensional shape.

Rebound
The interaction between a ball and a hard surface is to be captured and modelled by this component. The resulting force and speed is to be the output of this component.

Results
From the information received from the flight, court, and rebound components a detailed presentation of the results can be captured, analysed and presented by using a GUI.

These simple descriptions give structure as to what sort of models are to be used. A full solution is not expected at this point.

4.2.16 VERIFICATION OF PRELIMINARY SIMULATION DESIGN

The preliminary design can be verified against the simulation requirements. The RAG assessment method has been used to analyse whether the preliminary design metes the simulation. This RAG assessment can be seen in Table 4.3 below.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a squash ball in flight.			✓
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.			✓
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.			✓
4) The simulation needs to be run multiple times with the only change being the compound of the ball.			✓
5) The flight of the ball is constrained by a regulation sized squash court.			✓
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) The resolution of the time steps across the models is to be at 0.001 seconds.			✓
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓

Table 4.3: RAG assessment of the Preliminary design.

The RAG assessment in Table 4.3 shows that the design meets the simulation requirements and so it is possible to proceed to the next stage.

4.2.17 ARE THERE ANY EXISTING SIMULATIONS AND MODELS?

An assessment of the domain was made and it has been identified that there are existing models and simulations that could potentially be of use. This means that the SESEM process (section 3.5.6) can be used.

4.2.18 SYSTEMS ENGINEERING OF SELECTION OF EXISTING MODELS

The SESEM is a sub-process that is concerned with aiding in the selection of existing models.

Preliminary Simulation Design

This stage can be considered to have been completed.

Boundary of the Existing Model Selection Process

A secondary check has been conducted to assure that all of the prerequisites have been satisfied.

Assess the Model and Simulation Landscape

An assessment has been made of the ways in which the interactions that are to be investigated have previously been modelled and simulated in the past. Information has been gathered as to the mathematical tools and representations which may be of use.

Are Potential Models Available?

From assessing the model and simulation landscape there are potential models that could be of use.

Is this a New Product/ Platform?

It is proposed that in this instance the organisation has never conducted such a virtual investigation and testing of such potential designs. Hence there is no pre-existing simulations that could be modified for this experiment available in the organisation.

Locate Candidate Models and Simulations

An assessment of many possible physical models that could be of use were assessed. The physics models that have been assessed cover:

- Newtonian equations of motion
- Simple harmonic motion
- Equations of inertia
- Kinetic energy and potential energy
- Representations of squash courts dimensions.

It is proposed that in this instance that models exist in forms that can be directly accessed from open source locations.

Is Previous Model or Simulation Documentation Available?
There is significant documentation that is available for the potential models that have been identified. Models that do not have documentation and could not be understood were disregarded.

Can Someone in the Team Understand the Model or Simulation Explicitly?
With the documentation that has been assembled, a greater understanding of what the identified models are capable of has been gained.

The models and simulations which have been selected for further analysis are:

- Documentation of a Particle Moving in Free Space
- Documentation of Energy Transfer Model
- *Documentation of Squash Court In or Out Model.*

Does the Model or Simulation Match a Section of the Simulation Requirements?
Each candidate modes is evaluated against the simulation requirements. RAG assessments have been used throughout.

Particle Moving in Free Space

Table 4.4 below is the RAG assessment of the Particle Moving in Free Space model.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a squash ball in flight.			✓
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.	×		
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.	×		
4) The simulation needs to be run multiple times with the only change being the compound of the ball.			✓
5) The flight of the ball is constrained by a regulation sized squash court.		-	
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) The resolution of the time steps across the models is to be at 0.001 seconds.		-	
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.		-	

Table 4.4: RAG Assessment of the particle in flight model.

As the model in question is concerned with the flight of a particle, it does not comply with requirements two and three, hence it has been assigned a red value. This does not mean that the model is unsuitable for use; it means that the model only covers a section of the behaviour that the overall model is to mimic.

Requirements five, E, and F have been assigned amber values as the models do not at present have the required capabilities, however values could be changed or minor modifications made to make them complicit.

Energy Transfer Model

The table below is the RAG assessment of the Energy Transfer Model.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a squash ball in flight.	×		
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.		-	
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.	×		
4) The simulation needs to be run multiple times with the only change being the compound of the ball.			✓
5) The flight of the ball is constrained by a regulation sized squash court.		-	
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) The resolution of the time steps across the models is to be at 0.001 seconds.		-	
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.		-	

Table 4.5: RAG Assessment of the Energy Transfer Model.

The Energy Transfer Model does not comply with all of the simulation requirements in its current state. The model only represents an energy transfer hence has the potential to comply with requirement two and would need minor alterations to make it applicable. This model however does not comply with requirements one and three. For requirements five, E, and F minor modifications would need to be made for the model to comply. Requirement E would require a parameter change whereas F would require additional code.

Squash Court In or Out Model
The table below is the RAG assessment of the Squash Court In or Out Model.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a squash ball in flight.	×		
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.	×		
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.	×		
4) The simulation needs to be run multiple times with the only change being the compound of the ball.			✓
5) The flight of the ball is constrained by a regulation sized squash court.			✓
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) The resolution of the time steps across the models is to be at 0.001 seconds.		-	
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.		-	

Table 4.6: RAG Assessment of the Squash Court In or Out Model.

The Squash Court In or Out Model does not comply with requirements one, two, or three. However this does not mean that the model may not be of use as it critically satisfies requirement five. E and F would require minor alteration to make it fully compliant with the other requirements.

Assess if Individual Models and Simulations can be modified?
The three selected models can be modified.

Can the requirements be met by changing parameters?
Some of the requirements can be met by a change in parameters. However all of the models would require alterations of code.

Rework of Model or Simulation Needed
For each of the models the changes that would need to be made to get them to comply with the requirements has been captured.

Particle Moving in Free Space:
To make the model comply with requirement five the values of the distances need to be made available accessible as an input to ensure that the distances calculated are within the bounds of a squash court.

To comply with requirement E the time step of 0.001 seconds needs to be set and the model verified as still producing answers that are within expected values.

The saving of results captured by F may not be best captured in this part of the simulation however it could be easily added to the model if needed.

Energy Transfer Model:
For the Energy Transfer Model to be valid in this application the parameters relating to the specific interaction will need to be altered. This includes the characteristics of both the ball and the surfaces specified in a regulation squash court. The parameters for time step requires setting to 0.001 seconds and tested to ascertain that it remains within acceptable accuracy at this time step.

The results of the computation need to be captured in a means that can be saved for later analysis.

Squash Court In or Out Model:
The parameters for time step requires setting to 0.001 seconds and tested to ascertain that it remains within acceptable accuracy at this time step.

The results of the computation need to be captured in a means that can be saved for later analysis as well as making the outputs available at the boundaries of the model such that values can be passed from it.

Does an Alternative model with less rework exist?
From the available models and simulations that have been identified there are no existing available models or simulations. This means that the rework of the models is needed.

Rework Model or Simulation
The necessary changes that have been identified (in Rework of Model Needed) were made to the models and simulations.

Verification of altered models
With the alterations made the models can be verified against the model requirements. The simple RAG method has been implemented again. This allows for the direct comparison between the previous assessments. This verification allows for assurance that the selected models for fill the requirements of the integrated simulation. By having the comparison with the previous verifications allows for assessment of the effectiveness of the rework of the models.

Particle Moving in Free Space

The table below is the RAG assessment of the Particle Moving in Free Space model.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a squash ball in flight.			✓
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.	×		
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.	×		
4) The simulation needs to be run multiple times with the only change being the compound of the ball.			✓
5) The flight of the ball is constrained by a regulation sized squash court.		-	
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) The resolution of the time steps across the models is to be at 0.001 seconds.			✓
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓

Table 4.7:RAG assessment of the modified Particle Moving in Free Space model.

The red values for requirements two and three are not met due to the functionality of the model. However all of the constraints A through F are met.

Energy Transfer Model

The table below is the RAG assessment of the Energy Transfer Model.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a squash ball in flight.	×		
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.		-	
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.	×		
4) The simulation needs to be run multiple times with the only change being the compound of the ball.			✓
5) The flight of the ball is constrained by a regulation sized squash court.			✓
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) The resolution of the time steps across the models is to be at 0.001 seconds.			✓
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓

Table 4.8: RAG assessment of the modified Energy Transfer Model.

The red values for requirements one and three are not met due to the functionality of the model. However all of the constraints A through F are met.

Squash Court In or Out Model
The table below is the RAG assessment of the Squash Court In or Out Model.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a squash ball in flight.	×		
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.	×		
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.	×		
4) The simulation needs to be run multiple times with the only change being the compound of the ball.			✓
5) The flight of the ball is constrained by a regulation sized squash court.			✓
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) The resolution of the time steps across the models is to be at 0.001 seconds.			✓
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓

Table 4.9: RAG assessment of the modified squash court in out model.

The red values for requirements one, two and three are not met due to the functionality of the model. However all of the constraints A through F are met.

Complete integration tables
The integration tables have been completed and can be found in section 9.3.2

Available for Selection
The models that have progressed to this stage are suitable for selection for integration. The models that have been of use can be complied into a list for potential selection.

Models suitable for integration:

Particle Moving in Free Space
Energy Transfer Model
Squash Court In or Out Model.

End of Model Selection Process

This denotes the end of the systems engineering of selection of existing models sub-process.

4.2.19 SET FIRM ARCHITECTURE

With the understanding of the available models and simulations a firm architecture has been set. A representation of this architecture can be seen in Figure 4.6. Note how this is significantly different to what was intended in the preliminary architecture. The changes were due to the structure of the models which were the outputs from the model selection process.

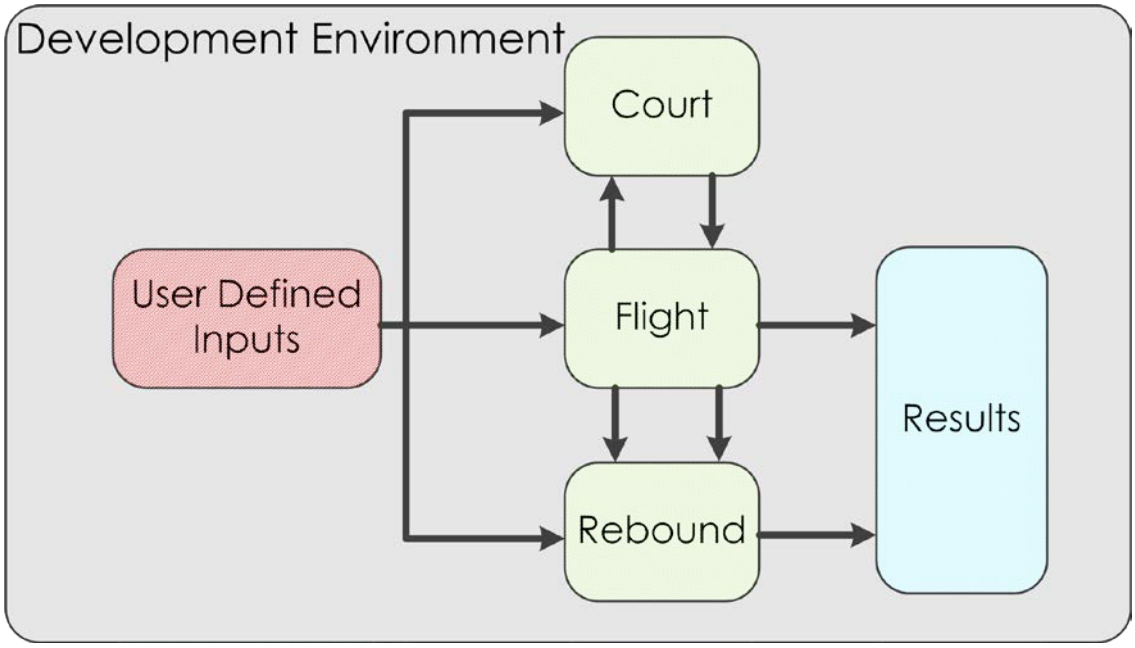

Figure 4.6: A representation of the Firm architecture of the simulation.

4.2.20 ASSESS COMPUTATIONAL REQUIREMENTS

With the selection of models and the architecture of the solution set the computational overheads can be assessed. The requirements A, B, and C specify the computational requirements that the simulation has to work within. The computational overheads are shown in Table 4.10 below.

MODEL NAME	RAM	DISK SPACE	PROCESSOR
Particle Moving in Free Space	2 K	25 KB	Intel® Core™ i5-3380M CPU @ 2.90GHz (4 CPUs)
Energy Transfer Model	4 K	38 KB	Intel® Core™ i5-3380M CPU @ 2.90GHz (4 CPUs)
Squash Court In or Out Model	3 K	41 KB	Intel® Core™ i5-3380M CPU @ 2.90GHz (4 CPUs)
TOTAL	9 K	104 KB	Intel® Core™ i5-3380M CPU @ 2.90GHz (4 CPUs)

Table 4.10: Computational requirements for the component parts of the architected simulation.

The totals are within the computational constraints specified in requirements.

4.2.21 VERIFICATION OF COMPUTATIONAL REQUIREMENTS

The RAG assessment method was used to assess whether if the total demands of the component parts is within specified parameters.

REQUIREMENT	RED	AMBER	GREEN
Total RAM			✓
Total Disk Space			✓
Total Processor			✓

Table 4.11: RAG assessment of the computational overheads of the proposed integrated simulation.

With the overheads of the proposed simulation within the acceptable bounds the next stage can be conducted.

4.2.22 DEFINE COMMUNICATIONS

The types of communication as well as how the communications are to be handled can be specified. In this case all of the communication will happen within the LabVIEW development environment.

4.2.23 DETAILED DESIGN

The specifics of the design can now be specified with individual parameters being set and each communication that needs to occur. Tables have been used to specify the communications between the models this includes the data types to be used. An example of this can be seen in Table 4.12 below. The overall structure of the design adheres to the firm architecture. Using this approach it is clear that there are gaps in both simulation capabilities and integration, these are captured in the next stage. There are relations that have been identified where the information produced by one model requires transformation to get it to a form where the consuming model can utilise it.

Table 4.12: Relationship table capturing the communication paths and data types of the detailed design.

The use of structures as those shown in Table 4.12 is not the only way to capture or display this type of information. It has been used in this case for clarity and transparency reasons.

4.2.24 DEFINE GAPS

This is a sub-process the results of which can be seen below.

Detailed Design

The detailed design using existing models and simulations has been formulated. However it has been already found that there are gaps and transformations that need to be resolved before a fully functioning integrated solution can be made. This detailed design is a prerequisite for this sub-process.

Boundary of the Existing Model Selection Process

From this point on the work conducted is part of the define gaps in the current detailed design.

Assess where selected models and simulations fulfil simulation Requirements

Using the RAG assessment method the three selected models against the overall requirements of the simulation it confirms there are still gaps that need to be filled. The RAG assessment of the three models can be seen in Table 4.13 below.

REQUIREMENT	RED	AMBER	GREEN	MODEL
1) The simulation is to capture the behaviour of a squash ball in flight.			✓	Particle Moving in Free Space
	×			Energy Transfer Model
	×			Squash Court In or Out
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.	×			Particle Moving in Free Space
		-		Energy Transfer Model
	×			Squash Court In or Out
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.	×			Particle Moving in Free Space
	×			Energy Transfer Model
	×			Squash Court In or Out
4) The simulation needs to be run multiple times with the only change being the compound of the ball.			✓	Particle Moving in Free Space
			✓	Energy Transfer Model
			✓	Squash Court In or Out
5) The flight of the ball is constrained by a regulation sized squash court.			✓	Particle Moving in Free Space
			✓	Energy Transfer Model
			✓	Squash Court In or Out
A) The total run time of the simulation should take less than five minutes to fully execute.			✓	Particle Moving in Free Space
			✓	Energy Transfer Model
			✓	Squash Court In or Out
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓	Particle Moving in Free Space
			✓	Energy Transfer Model
			✓	Squash Court In or Out
C) No specific computational hardware or peripherals are to be used.			✓	Particle Moving in Free Space
			✓	Energy Transfer Model
			✓	Squash Court In or Out
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓	Particle Moving in Free Space
			✓	Energy Transfer Model
			✓	Squash Court In or Out
E) The resolution of the time steps across the models is to be at 0.001 seconds.			✓	Particle Moving in Free Space
			✓	Energy Transfer Model
			✓	Squash Court In or Out
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓	Particle Moving in Free Space
			✓	Energy Transfer Model
			✓	Squash Court In or Out

Table 4.13: RAG assessment of the three selected models against the overall simulation requirements.

The RAG assessment shown in Table 4.13 is different from the previous tables as it has all three of the selected models tested against each requirement.

Are All Requirements Met?
From the analysis it is clear that there are requirements still to be met (requirement three), requirements that have been identified to have been partially met (requirement two), and requirements have been completely satisfied (four and five).

Asses Existing Model Boundaries
Using the understanding that is captured in the relationship tables it is possible to assess where the boundaries of the models are and where the overlaps are. Currently, in this instance, it is recognised that there is very little overlap between the models and as such there are many transforms that need to occur.

Asses Necessary Transformations to Comply with Requirements
From the assessment of the model boundaries and the extent to which the requirements have been for filled it is clear there are a number of changes that need to be made in order to make the overall simulation complicit with the requirements.

There are functional simulation requirements that have been fully met, partially met, and not met at all: requirements one, two, and three respectively. However it is to be noted that all of the selected component models are complicit with the simulation constraints. Therefore any changes that are made to the component models must still take the constraints into consideration.

Transformations that need to occur include:

- Tracking of the balls position throughout its transit
- The saving of the important data points
- The modification of the Energy Transfer Model to make it comply with requirement two
- The creation of a model that captures the interaction between the player's racket and the ball
- A means of starting and terminating the simulation
- where the ball starts
- The output of the Energy Transfer Model is to be taken and used to calculate the flight back to the player.

The transformations identified above take the form of notes rather than statements that can be worked from.

Define Requirements for work to fill the gaps

The following requirements have been formulated from the identified translations and specify the required works.

i. All models that concern the balls movement are to track the movement in the terms specified in the Squash Court In or Out models coordinate system.

ii. All models concerning the movement of the ball are to output values that can be passed to the Squash Court In or Out model.

iii. A means of capturing the required data points is to be created. The data to be captured incudes the 3D position of the ball, whether the ball is in or out in the court, the speed of the ball at each time step of the model, and the calculated energy transfer that happens during the interactions with surfaces.

iv. The Energy Transfer Model is to be altered to be made fully complicit with requirement two.

v. A model is to be created that captures the interaction between the player's racket and the ball detailing the position of the ball and the velocity that was imparted to the ball.

vi. The integrated model is to have a defined start and end condition that allows for the transit of the ball from the player to the wall and back to the player.

vii. A model is to be created that takes the output from the Energy Transfer Model and captures the flight back to the player.

viii. The time step specified at 0.001 seconds is to be used as the refresh rate simulation time. E.g. each calculation will represent the change that has happened in the last 0.001 seconds of the balls behaviour.

ix. Only one calculation is to be made during each iteration in a procedural manner.

Assess Data that Require Transfer

As there are different parts of the overall simulation, it will be necessary for the components to communicate and data be passed between them. Using the detailed design Table 4.12 as a starting point, the required additions were made. Due to the size of the table it has been split into three tables Table 4.14, Table 4.15, and Table 4.16.

Inputs		Outputs: Squash Court in or Out																				
		Front Half	Left Service box	Right Service box	Left Quarter	Right Quarter	Back half	Top wall	Front wall	Bottom wall	Tin	Out Bounds side wall left	In bounds side wall left	Out bounds side wall right	In bounds side wall right	Out bound height	In bound height	Back wall out bound	Back wall in bound	Height of line at point x
Particle_Moving_in_Free_Space	Starting Velocity U (mps)																			
	Distance S (m)																			
	Final Velocity V (mps)																			
	Sample Rate (NB samples ofper Time)																			
	X Coordinate																			
	Y Coordinate																			
	Z Coordinate																			
	Stop Loop																			
Energy Transfer model	Mass (Kg)																			
	Velocity (meters per second mps)																			
	Energy transfer efficiency (%)																			
Squash Court in or Out	Total Win																			
	Total Height																			
	Height of Bottom of Service Line																			
	Front wall top of Service Line																			
	Length to short line																			
	Width of service box																			
	Width of back Quarter																			
	Depth of service box																			
	Height of back wall line																			
	X Coordinate																			
	Y Coordinate																			
	Z Coordinate																			
Energy Transfer model and Particle Moving in free space	Time (S)																			
	Distance S (m)																			
	Starting Velocity U (mps)																			
	X Coordinate																			
	Y Coordinate																			
	Z Coordinate																			
	Location of Front wall								FPSV											
	Stop Loop																			
Ball on Racket	X Coordinate																			
	Y Coordinate																			
	Z Coordinate																			
	speed of impact																			
	Energy transfer efficiency (%)																			
Saving Data	Front Half	BSV																		
	Left Service box		BSV																	
	Right Service box			BSV																
	Left Quarter				BSV															
	Right Quarter					BSV														
	Back half						BSV													
	Too wall							BSV												
	Front wall								BSV											
	Bottom wall									BSV										
	Tin										BSV									
	In bounds side wall left												BSV							
	In bounds side wall right														BSV					
	Back wall in bound																			
	Striking Velocity																			
	Remaining energy																			
	Kinetic energy (Jules)																			
	Energy lost (Jules)																			
	Returning velocity (mps)																			

Key	
Term	Symbol
Floating point single value	FPSV
Floating point array	FPA
Integer single value	ISV
Integer array	IA
Boolean single value	BSV
Boolean array	BA

Table 4.14: The communications between the outputs of the Squash Court In or Out Model and the inputs of all other component parts of the simulation are represented.

Inputs		Outputs												
		Energy Transfer model				Particle_Moving_in_Free_Space								
		Remaining energy	Kinetic energy (Jules)	Energy lost (Jules)	Returning velocity (mps)	Loop counts	X Coordinate	Y Coordinate	Z Coordinate	Time (T) (Seconds)	Acceleration (A) (meters per second per second)	Time Sample (Ts) (seconds)	Displacement step St (meters)	Displacement step St (meters) Array
Particle_Moving_in_Free _Space	Starting Velocity U (mps)													
	Distance S (m)													
	Final Velocity V (mps)													
	Sample Rate (NB samples ofcer Time)													
	X Coordinate													
	Y Coordinate													
	Z Coordinate													
	Stop Loop													
Energy Transfer model	Mass (Kg)													
	Velocity (meters per second mps)													
	Energy transfer efficiency (%)													
Squash Court in or Out	Total With													
	Total Height													
	Height of Tin													
	Front Wall Bottom of Service Line													
	Front wall top of Service Line													
	Length to short line													
	With of service box													
	With of back Quarter													
	Depth of service box													
	Height of back wall line													
	X Coordinate						FPSV							
	Y Coordinate							FPSV						
	Z Coordinate								FPSV					
Energy Transfer model and Particle Moving in free space	Time (S)													
	Distance S (m)												FPSV	
	Starting Velocity U (mps)				FPSV									
	X Coordinate						FPSV							
	Y Coordinate							FPSV						
	Z Coordinate								FPSV					
	Location of Front wall													
	Stop Loop													
Ball on Racket	X Coordinate													
	Y Coordinate													
	Z Coordinate													
	speed of impact													
	Energy transfer efficiency (%)													
Saving Data	Front Half													
	Left Service box													
	Right Service box													
	Left Quarter													
	Right Quarter													
	Back half													
	Top wall													
	Front wall													
	Bottom wall													
	Tin													
	In bounds side wall left													
	In bounds side wall right													
	Back wall in bound													
	Striking Velocity													
	Remaining energy	FPSV												
	Kinetic energy (Jules)		FPSV											
	Energy lost (Jules)			FPSV										
	Returning velocity (mps)				FPSV									

Key	
Term	Symbol
Floating point single value	FPSV
Floating point array	FPA
Integer single value	ISV
Integer array	IA
Boolean single value	BSV
Boolean array	BA

Table 4.15: The communications between the outputs of the Energy Transfer Model, the Particle Moving in Free Space model, and the inputs of all other component parts of the simulation are represented.

Inputs		Outputs														
		Energy Transfer model and Particle Moving in free space									Ball on Racket					
		Loop counts	Final Velocity V (MpS)	Velocity incremental V (MpS)	X Coordinate	Y Coordinate	Z Coordinate	Final Velocity	X Coordinate	Y Coordinate	Z Coordinate
Particle_Moving_in_Free_Space	Starting Velocity U (mps)							FPSV			
	Distance S (m)										
	Final Velocity V (mps)							FPSV			
	Sample Rate (NB samples ofper Time)										
	X Coordinate								FPSV	FPSV	FPSV
	Y Coordinate										
	Z Coordinate										
	Stop Loop										
Energy Transfer model	Mass (Kg)										
	Velocity (meters per second mps)										
	Energy transfer efficiency (%)										
Squash Court in or Out	Total Wth										
	Total Height										
	Height of Tin										
	Front Wall Bottom of Service Line										
	Front wall top of Service Line										
	Length to short line										
	Wth of service box										
	Wth of back Quarter										
	Depth of service box										
	Height of back wall line										
	X Coordinate				FPSV				FPSV		
	Y Coordinate					FPSV				FPSV	
	Z Coordinate						FPSV				FPSV
Energy Transfer model and Particle Moving in free space	Time (S)										
	Distance S (m)										
	Starting Velocity U (mps)										
	X Coordinate								FPSV		
	Y Coordinate									FPSV	
	Z Coordinate										FPSV
	Location of Front wall										
	Stop Loop										
Ball on Racket	X Coordinate										
	Y Coordinate										
	Z Coordinate										
	speed of impact										
	Energy transfer efficiency (%)										
Saving Data	Front Half										
	Left Service box										
	Right Service box										
	Left Quarter										
	Right Quarter										
	Back half										
	Top wall										
	Front wall										
	Bottom wall										
	Tin										
	In bounds side wall left										
	In bounds side wall right										
	Back wall in bound										
	Striking Velocity							FPSV			
	Remaining energy										
	Kinetic energy (Jules)										
	Energy lost (Jules)										
	Returning velocity (mps)										

Key	
Term	Symbol
Floating point single value	FPSV
Floating point array	FPA
Inteaer single value	ISV
Integer array	IA
Boolean single value	BSV
Boolean array	BA

Table 4.16: The communications between the outputs of the Energy Transfer Model and particle in free space, the Ball on racket model, and the inputs of all other component parts of the simulation are represented.

Define Requirements for Communication
The decision was made to use one development and execution environment to conduct this simulation. This makes the specification of the communications considerably simpler. The requirements regarding the data communications for this simulation are as follows.

- *All communications are to be conducted using native LabVIEW features and data types.*
- *All communications are to be produced at the rate of 0.001 seconds (simulation time).*

End Of Define Gaps in the SEIS
This denotes the end of the Fill the Gaps sub-process. The next stage is element 25 in the SEIS process.

4.2.25 FILL GAPS

The work was conducted to make the changes to existing models as well as create the new models and component parts. The Fill the Gaps element is a sub-process each stage of which is detailed below.

Define Gaps
The definitions of the gaps process has been completed ensuring that the necessary information is present for this process to be conducted without issue.

Boundary of the Existing Model Selection Process
This denotes the start of this sub-process.

Gain understanding of Requirements from Define the Gaps Process
The requirements from the previous sub-process (Defining Gaps in the SEIS) have been inspected and any issues which became apparent were questioned and resolved. A complete understanding of the requirements is now considered to have been captured.

Architect the solution/s

From the requirements there are three new components that are needed to be added. The overall architecture of the solution is to change as shown in the Figure 4.7 below.

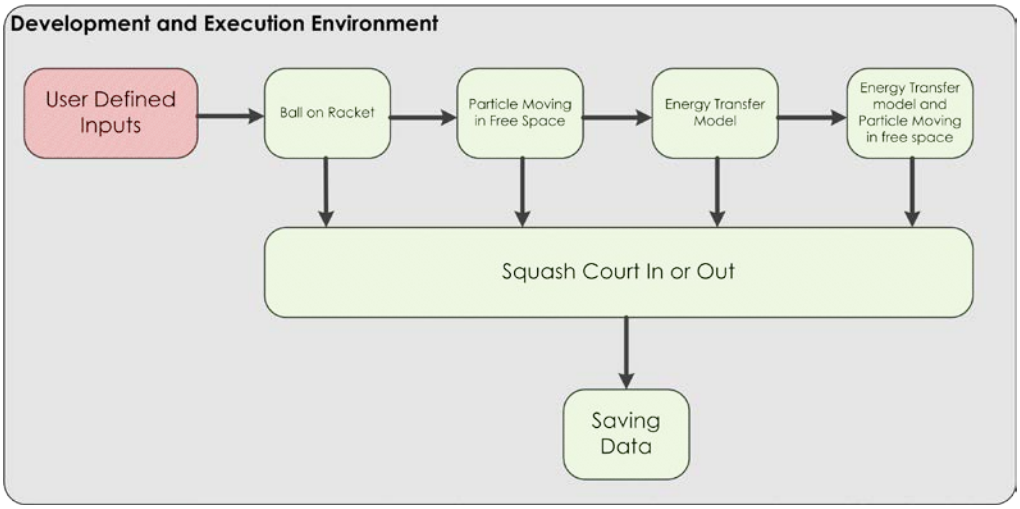

Figure 4.7: The simulation architecture after the Fill the Gaps alterations. Note the change in structure from that in Figure 4.6.

A 'new' component is to capture and save all of the relevant data in a suitable format can be seen in Figure 4.7 above. The inputs of Saving Data are specified in Table 4.14, Table 4.15, and Table 4.16. The output of this model is a CSV file as this allows for multiple software programs to interrogate the file at a later date.

For the ball on racket model the inputs and outputs are specified in Table 4.14, Table 4.15, and Table 4.16. The model itself will also take the role as the starting point of the simulation and will define the starting velocity and position of the ball.

The Energy Transfer Model and Particle Moving in Free Space models inputs and outputs have been defined in Table 4.14, Table 4.15, and Table 4.16. The model will take the values from the Energy Transfer Model and track the flight path of the ball back to the player. This will use the same Newtonian physics that was used as part of the Particle Moving in Free Space.

The Particle Moving in Free Space inputs and outputs will be modified. Each step of the calculation will now also output the three dimensional coordinates of where the ball is at each time step.

Design Potential Solutions

Each of the architected components has been designed in full detail. Using the required functionality as a direction as to how to code them.

Verification of potential solutions

The potential designs is verified against the 'Fill the Gap' requirements as well as the overarching simulation requirements. The overall assessment was conducted using the RAG method see Table 4.17 below.

REQUIREMENT	Particle Moving in Free Space	Energy Transfer Model	Squash Court In or Out	Energy Transfer Model and Particle Moving in Free Space	Ball on Racket	Saving Data
1) The simulation is to capture the behaviour of a squash ball in flight.	G	R	R	G	R	R
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.	R	G	R	R	R	R
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.	R	R	R	R	G	R
4) The simulation needs to be run multiple times with the only change being the compound of the ball.	G	G	G	G	G	G
5) The flight of the ball is constrained by a regulation sized squash court.	NA	NA	G	NA	NA	NA
A) The total run time of the simulation should take less than five minutes to fully execute.	G	G	G	G	G	G
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.	G	G	G	G	G	G
C) No specific computational hardware or peripherals are to be used.	G	G	G	G	G	G
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.	G	G	G	G	G	G
E) The resolution of the time steps across the models is to be at 0.001 seconds.	G	G	G	G	G	G
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.	NA	NA	NA	NA	NA	G

Table 4.17 : RAG assessment of the component parts of the overall simulation after the Fill the Gaps in sub-process part one.

REQUIREMENT	Particle Moving in Free Space	Energy Transfer Model	Squash Court In or Out	Energy Transfer Model and Particle Moving in Free Space	Ball on Racket	Saving Data
i) All models that concern the balls movement are to track the movement in the terms specified in the Squash Court In or Out models coordinate system.	G	NA	G	G	G	G
ii) All models concerning the movement of the ball are to output values that can be passed to the Squash Court In or Out model.	G	G	NA	G	G	NA
iii) A means of capturing the required data points is to be created. The data to be captured incudes (the 3D position of the ball, whether the ball is in or out in the court, the speed of the ball at each time step of the model, and the calculated energy transfer that happens during the interactions with surfaces.	R	R	R	R	R	G
iv) The Energy Transfer Model is to be altered to be made fully complicit with requirement two.	NA	G	NA	G	NA	NA
v) A model is to be created that captures the interaction between the player's racket and the ball detailing the position of the ball and the velocity that was imparted to the ball.	NA	NA	NA	NA	G	NA
vi) The integrated model is to have a defined start and end condition that allows for the transit of the ball from the player to the wall and back to the player.	G	G	G	G	G	G
vii) A model is to be created that takes the output from the Energy Transfer Model and captures the flight back to the player.	NA	NA	NA	G	NA	NA
viii) The time step specified at 0.001 seconds is to be used as the refresh rate simulations time. E.g. each calculation will represent the change that has happened in the last 0.001 seconds of the balls behaviour.	G	G	G	G	G	G
ix) Only one calculation is to be made during each iteration in a procedural manor.	G	G	G	G	G	G

Table 4.18 :RAG assessment of the component parts of the overall simulation after the Fill the Gaps in sub-process part two.

The RAG assessment in Table 4.17 indicates that all of the requirements are fulfilled. Despite the number of Red indicated for functionality there is at least one other model that is capable of for filling the functionality.

Build Solutions

The designs were then implemented and the code scripted.

Verification of build solutions

The solutions that were implemented are to be verified against the requirements to ensure that they were implemented as was intended. The RAG assessment was used to ascertain as to if the requirements were met as intended see Table 4.18 below.

REQUIREMENT	Particle Moving in Free Space	Energy Transfer Model	Squash Court In or Out	Energy Transfer Model and Particle Moving in Free Space	Ball on Racket	Saving Data
1) The simulation is to capture the behaviour of a squash ball in flight.	G	R	R	G	R	R
2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.	R	G	R	R	R	R
3) The simulation is to capture the behaviour of a ball interacting with a player's racket.	R	R	R	R	G	R
4) The simulation needs to be run multiple times with the only change being the compound of the ball.	G	G	G	G	G	G
5) The flight of the ball is constrained by a regulation sized squash court.	NA	NA	G	NA	NA	NA
A) The total run time of the simulation should take less than five minutes to fully execute.	G	G	G	G	G	G
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.	G	G	G	G	G	G
C) No specific computational hardware or peripherals are to be used.	G	G	G	G	G	G
D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.	G	G	G	G	G	G
E) The resolution of the time steps across the models is to be at 0.001 seconds.	G	G	G	G	G	G
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.	NA	NA	NA	NA	NA	G

Table 4.19: RAG assessment of the component parts of the overall simulation after the Fill the Gaps in sub-process part one.

REQUIREMENT	Particle Moving in Free Space	Energy Transfer Model	Squash Court In or Out	Energy Transfer Model and Particle Moving in Free Space	Ball on Racket	Saving Data
i) All models that concern the balls movement are to track the movement in the terms specified in the Squash Court In or Out models coordinate system.	G	NA	G	G	G	G
ii) All models concerning the movement of the ball are to output values that can be passed to the Squash Court In or Out model.	G	G	NA	G	G	NA
iii) A means of capturing the required data points is to be created. The data to be captured incudes (the 3D position of the ball, whether the ball is in or out in the court, the speed of the ball at each time step of the model, and the calculated energy transfer that happens during the interactions with surfaces.	R	R	R	R	R	G
iv) The Energy Transfer Model is to be altered to be made fully complicit with requirement two.	NA	G	NA	G	NA	NA
v) A model is to be created that captures the interaction between the player's racket and the ball detailing the position of the ball and the velocity that was imparted to the ball.	NA	NA	NA	NA	G	NA
vi) The integrated model is to have a defined start and end condition that allows for the transit of the ball from the player to the wall and back to the player.	G	G	G	G	G	G
vii) A model is to be created that takes the output from the Energy Transfer Model and captures the flight back to the player.	NA	NA	NA	G	NA	NA
viii) The time step specified at 0.001 seconds is to be used as the refresh rate simulations time. E.g. each calculation will represent the change that has happened in the last 0.001 seconds of the balls behaviour.	G	G	G	G	G	G
ix) Only one calculation is to be made during each iteration in a procedural manor.	G	G	G	G	G	G

Table 4.20 : RAG assessment of the component parts of the overall simulation after the Fill the Gaps in sub-process part two.

This assessment of the produced components indicates that all of the identified gaps have been filled and that this sub-process is complete.

End of fill Gaps sub-process

This marks the end of the Fill the Gaps sub-process.

4.2.26 COMPETE INTEGRATION TABLES

The integration tables as defined in the section 3.7 are completed for each of the component parts to be integrated. The completed integration tables can be found in section 9.3.2. An interesting point is that the newly generated components do not have an already generated validation experiment. One has to be generated and validated as being representative of the behaviour of the component.

4.2.27 VERIFICATION OF DETAILED DESIGN

The verification of the detailed design takes the form of comparing the design to the original DOE. The second aspect of the verification is to ascertain if the passing of information between the component parts is meaningful.

A simple binary yes or no table was created to display if the original purpose has been captured by the simulation as it stands.

STATEMENT	PASS	FAIL
To ascertain the behaviour of a ball around a standard court when the compounds the ball is made from is varied.	✓	
Changing the compounds that the ball is made from alters the speed that the ball returns off the wall when struck against it.	✓	
The variables that may be changed during the experiment: Service speed of the ball Characteristics of the ball that can change include; stiffness, construction, temperature, weight, rebound resilience, life span, design, and molecular composition. Imparted, potential, and kinetic energy of the ball. Velocity, speed, and direction of the ball. Surface interaction between the racket and the ball.	✓	
The base experiment will hold all elements constant with the compound of the ball will change. While the ball is moving it is considered to be a particle. There is an energy transfer when the ball interacts with each surface.	✓	
The simulation is to capture the behaviour of a squash ball in flight.	✓	
The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.	✓	
The simulation is to capture the behaviour of a ball interacting with a player's racket.	✓	
The simulation needs to be run multiple times with the only change being the compound of the ball.	✓	
The flight of the ball is constrained by a regulation sized squash court.	✓	

Table 4.21: Verification that ascertains if the proposed simulation meets the original purpose of the test.

The results of the overall verification test are shown in Table 4.19. From this test there is the assurance that the proposed simulation still addresses the original purpose of the test.

The next validation test is the inspection of data transfer, ensure that the data types are suitable, and the logic is sound. To ensure that all communications are tested the design tables that were used earlier to specify the communication were used. Each communication was located on the table. If the communication was not on the table it was added. If the communication was meaningful, the data types are either the same, or if conversions are needed the relevant information is retained. A colour coded system has been used for the tables with green representing acceptable communication and Red representing meaningless or questionable communication. The updated tables can be found below Table 4.20, Table 4.21, and Table 4.22.

Outputs

Squash Court in or Out

Inputs		Front Half	Left Service box	Right Service box	Left Quarter	Right Quarter	Back half	Top wall	Front wall	Bottom wall	Tin	Out Bounds side wall left	In bounds side wall left	Out bounds side wall right	In bounds side wall right	Out bound height	In bound height	Back wall out bound	Back wall in bound	Height of line at point x
Particle_Moving_in_Free_Space	Starting Velocity U (mps)																			
	Distance S (m)																			
	Final Velocity V (mps)																			
	Sample Rate (NB samples ofer Time)																			
	X Coordinate																			
	Y Coordinate																			
	Z Coordinate																			
	Stop Loop																			
Energy Transfer model	Mass (Kg)																			
	Velocity (meters per second mps)																			
	Energy transfer efficiency (%)																			
Squash Court in or Out	Total With																			
	Total Height																			
	Height of Tin																			
	Front Wall Bottom of Service Line																			
	Front wall top of Service Line																			
	Length to short line																			
	With of service box																			
	With of back Quarter																			
	Depth of service box																			
	Height of back wall line																			
	X Coordinate																			
	Y Coordinate																			
	Z Coordinate																			
Energy Transfer model and Particle Moving in free space	Time (S)																			
	Distance S (m)																			
	Starting Velocity U (mps)																			
	X Coordinate																			
	Y Coordinate																			
	Z Coordinate																			
	Location of Front wall								FPSV											
	Stop Loop																			
Ball on Racket	X Coordinate																			
	Y Coordinate																			
	Z Coordinate																			
	speed of impact																			
	Energy transfer efficiency (%)																			
Saving Data	Front Half	BSV																		
	Left Service box		BSV																	
	Right Service box			BSV																
	Left Quarter				BSV															
	Right Quarter					BSV														
	Back half						BSV													
	Top wall							BSV												
	Front wall								BSV											
	Bottom wall									BSV										
	Tin										BSV									
	In bounds side wall left												BSV							
	In bounds side wall right														BSV					
	Back wall in bound																	BSV		
	Striking Velocity																			
	Remaining energy																			
	Kinetic energy (Jules)																			
	Energy lost (Jules)																			
	Returning velocity (mps)																			

Key

Term	Symbol
Floating point single value	FPSV
Floating point array	FPA
Integer single value	ISV
Integer array	IA
Boolean single value	BSV
Boolean array	BA

Colour Code

Meaningful	(green)
Not meaningful	(red)

Table 4.22: Verification table showing communications between; Squash Court In or Out, Energy Transfer Model, Particle Moving in Free Space, and Saving Data.

Table 4.23: Verification table showing communications between; Energy Transfer Model, Particle Moving in Free Space, Squash Court In or Out, Energy Transfer Model, Particle Moving in Free Space, and Saving Data.

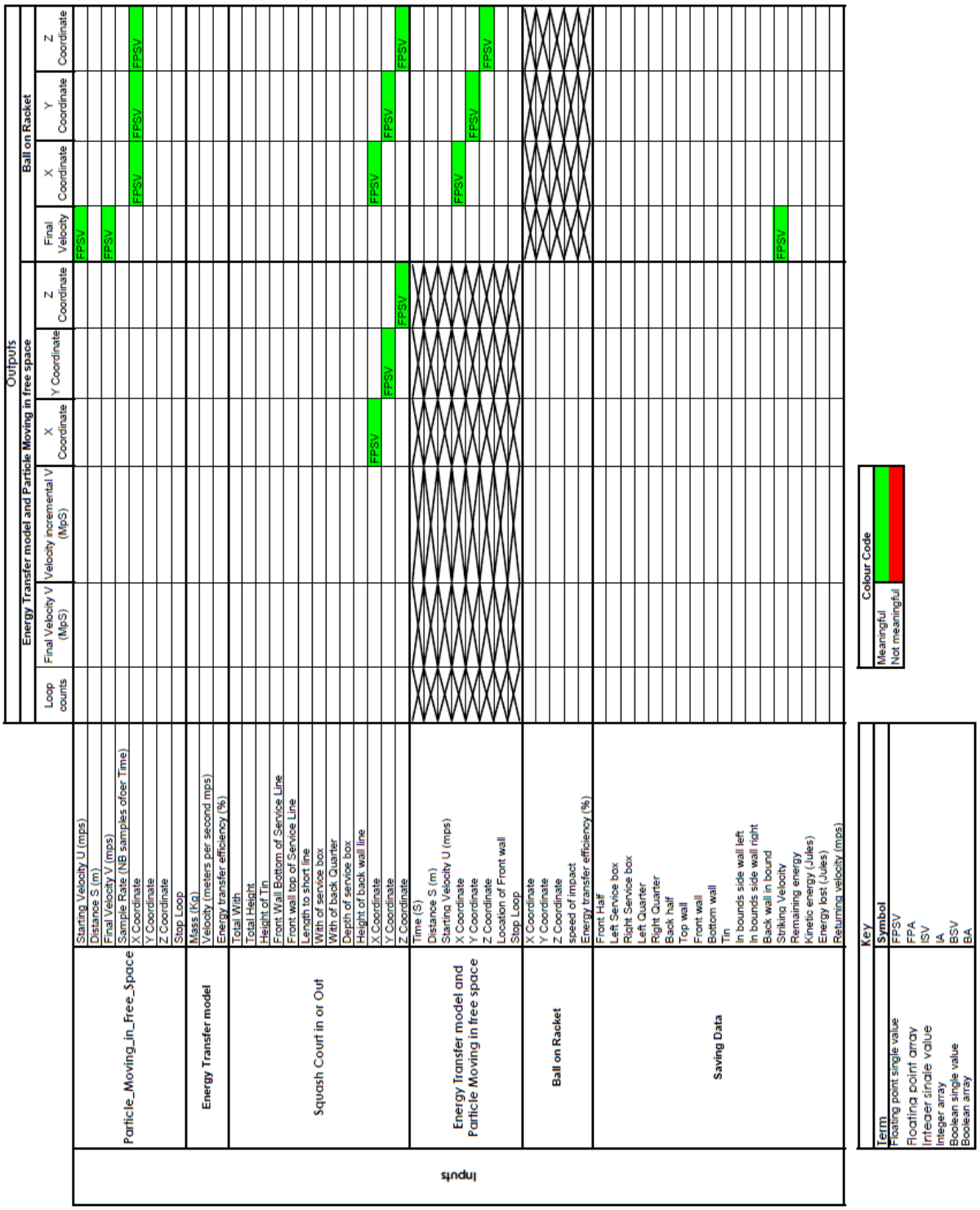

Table 4.24: Verification table showing communications between Energy Transfer Model; Particle Moving in Free Space, Ball on Racket, Particle Moving in Free Space, Squash Court In or Out, Energy Transfer Model, Particle Moving in Free Space, and Saving Data.

The verification tables have not found any issues and so it can be considered that the proposed simulation should integrate in a meaningful way and produce a usable simulation that will address the intended hypothesis.

4.2.28 INTEGRATE SIMULATIONS

The component parts of the simulation were integrated as specified.

4.2.29 VERIFICATION OF SPECIFIC INTEGRATION POINTS

Each of the communications were tested by using the verification experiment that formed part of the integration tables. For each component part of the simulation that was passed values within expected output values and the outputs were assessed. If all of the output values were within bounds specified in the integration tables the integration is considered to pass. If output values are outside the specified values the component and hence the integration is brought into question. Components that fail are investigated as to why there has been a change in the components behaviour once it has been integrated.

To record the results of the verification test the same style of table as has been used for the design process was used combined with a RAG representation. The results of this test can be seen in Table 4.23. In this instant there are no integration issues and so the next stage can be moved to.

		Outputs				
		Squash Court in or Out	Energy Transfer model	Particle Moving in Free Space	Energy Transfer model and Particle Moving in free space	Ball on Racket
Inputs	Particle Moving in Free Space					
	Energy Transfer model					
	Squash Court in or Out					
	Energy Transfer model and Particle Moving in free space					
	Ball on Racket					
	Saving Data					

Table 4.25: Record of the results of the verification of the integrations between the component parts. Red, Amber, and Green represent, significant issue, minor error which can be easily rectified, and no issues, respectively.

4.2.30 VERIFICATION OF INTEGRATED SIMULATION AS A WHOLE

Due to the models being developed individually without concern for this specific experiment, the verification experiments do not align with each other. Therefore an experiment has been formulated to test the integration. The inputs can be specifically defined whereas only general terms of output can be predicted. Hence if the outputs are outside expected ranges, further investigation will be required. The absolute bounds for each model can be found in the integration tables.

Experiment Inputs

The ball will be launched at 25 ms^{-1} (Around 55 miles per hour) at a height of 1.5m from the centre line on the service line towards the front wall.

Expected Outputs
The ball will return to the position where it was launched from.

The speed of the ball will reduce after it has rebounded of the wall.

The time it takes to get to the wall will be less than the time it takes to return to the starting point.

Three dimensional position data will be returned for the location of the ball at each discrete time step.

Results of Test
The inputs were entered into the simulation. The simulation ran and executed. The results were as expected and within expected boundaries. A high level of confidence has been achieved from this verification experiment that the outputs of the simulation are meaningful.

4.2.31 CONDUCT EXPERIMENT
The experiment can now be conducted as specified in the early stages of the systems engineering in integration of simulations.

4.2.32 FEEDBACK INTO DESIGN PROCESS
The results regarding the effects of different materials and the speed the ball returns from the front wall after service. This marks the end of the first case study. The remaining stages of the proposed process will not be conducted. The remaining stages of the proposed processes are outside the bounds of the virtual simulation and test, and move into the physical domain. The work so far conducted as part of the case study demonstrates the key relations between an experiment and the means by which to implement it using existing simulations.

4.3 AUTOMOTIVE CASE STUDY: ABS AND STEERING

This automotive case study reflects the methods that have been developed in section 3. This validation test will consist of the systems engineering processes that have been developed to improve the problem situation that was identified in section 1.2.4.

The automotive case study uses models from a sponsoring organisation. To maintain the nature of commercially sensitive information, some aspects of the models chosen are abstractions from those in use. These abstractions have an effect on the validation function which is discussed below.

The first iteration of this case study used real world models from an automotive company as well as in depth material from third party COTS manufacturers. The test revolved around the selection of a new potential COTS brake system being integrated to an existing platform.

The proposed processes were followed and the following architecture was developed.

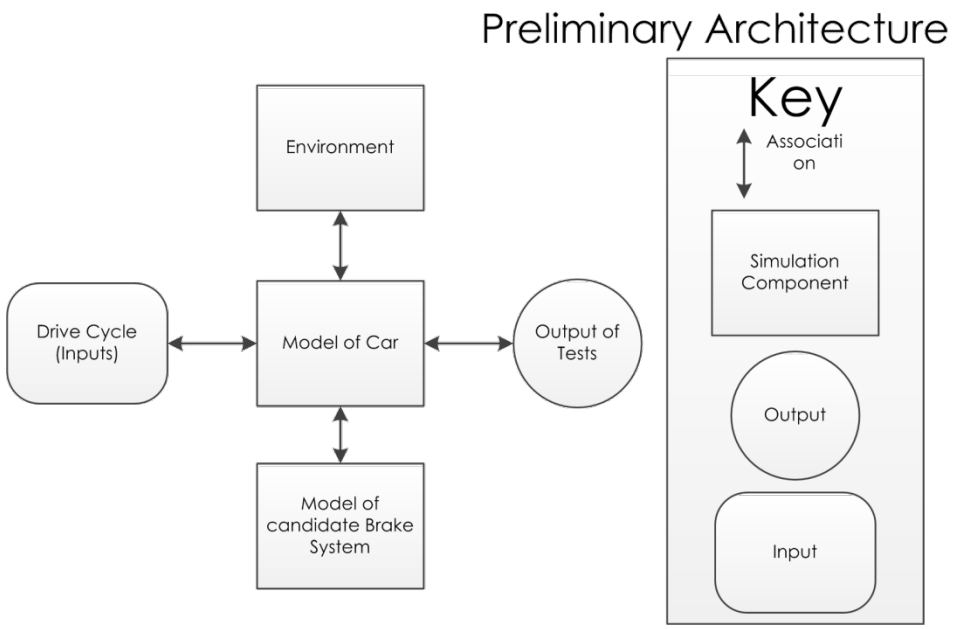

Figure 4.8: The preliminary architecture of the simulation designed to test the effects of a new potential brake system on the longitudinal behaviour of a car.

When the proposed methods were applied issues arose around elements of the SESEM process elements 15, 16,17, and 18 which highlight issues regarding the available documentation of third party components. After discussions with two groups of practising automotive engineers who had intimate knowledge of the respective system models, fundamental issues became apparent which would have required complete rework to align assumptions made. This in itself is a valuable result of testing the processes, as initially the models appeared compatible for integration, but then a non-automotive engineer was able to

identify semantic incompatibilities. These incompatibilities were then validated by practising automotive engineers which in turn validates the proposed processes.

The automotive experiment which is to be conducted in this case study draws from experimental work produced by theme one of the JLR EPSRC PSi Project [89]. As part of these case studies, existing Matlab Simulink models as well as other models found in the literature were used as a platform to test other technologies. Critically for this case study, the component models were created independently of the PSi project and there is significant documentation that can be interrogated with the NLP POC see section 5.4. Model selection, documentation, and their use was achieved independently to this integration study. Importantly the results of the theme one integration have been validated by automotive engineers who specialise in these component systems. The pool of simulation components is to be the same, the means by which the suitability for integration is to be assessed is significantly different from theme one, and the resultant integration may be different. This will allow for a clear validation point.

4.3.1 The Purpose of the Test

The validation test of the case study steps through each stage of the proposed methods. Once the proposed methods have been competed the entire process as a whole is analysed. This enables the suitability of each stage to be analysed as well as the process as a whole.

The test is conducted as if an engineer working in the automotive sector was using the proposed methods.

4.3.2 The system being simulated

The task that will be simulated is one where the engineer has been tasked with producing a simulation to assess if it is possible to use an existing controller to link the braking and steering systems together. The effects on the vehicles handling performance are to be investigated when these two systems are linked by a single controller.

4.3.3 Potential bias

Testing bias can affect the validity of any results. It is hence vital to identify the bias that may affect the testing. There is one bias that cannot go unmentioned: the individual testing the process is also the one who developed the process in the first place. There is hence the possibility that additional steps not dictated in the methods section will be used or they will be modified during the execution of the process. All efforts will be made to only conduct the stages that are laid out in the method section. All work will be explicitly stated and recorded for later analysis.

4.3.4 CUSTOMER WANTS

It is proposed that an automotive company wishes to investigate, through simulation, the possibility of linking together the steering and braking systems under one controller. With these two sub-systems combined under one controller, the effects of road handling are to be investigated. It is required that existing component systems are to be used. Hence it is also considered that existing models are to be used as the basis for the investigation. The steering and brake systems isolated as well integrated using a controller is illustrated in Figure 4.9 below.

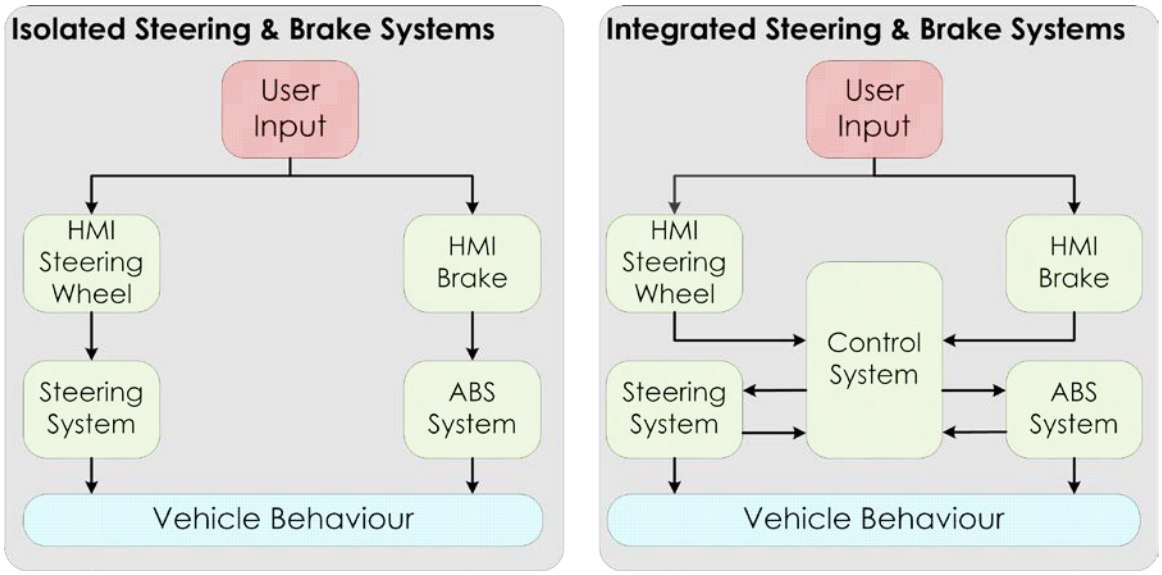

Figure 4.9 The steering and brake systems isolated (left). The steering and brake systems integrated using a control system (right).

4.3.5 SYSTEM REQUIREMENTS

This work is being viewed as an investigation as to whether linking braking and handling is a desirable course of action for future work. However there are still system requirements that need to be met.

Requirements for a Combined Braking and Steering System

1. The brakes shall retard the longitudinal motion of the vehicle.
2. The braking system will prevent the locking of the wheels during braking.
3. If braking on a surface with a low coefficient of friction, the wheels should not lock.
4. The brake system modulates the brake line pressure independent of the user controlled pedal.
5. The brake system is to bring the wheel to the slip range that is optimal for braking performance.
6. The steering system shall change the angular direction of the vehicle.
7. The steering system will amplify the inputs form the user.

4.3.6 System Architecture

The architecture of the system being designed is that of a sports utility vehicle. The vehicle has four wheels, internal combustion engine, gearbox, a rack and pinion steering, and independent suspension for each wheel. It is intended that the brake and steering systems are linked under one controller.

4.3.7 System Design

The implementation of the architecture is intended to use all existing physical components. The individual components are tried, tested and well established. The new aspect to this design is the way in which they are to be connected and controlled.

4.3.8 Systems Engineering In Model Integration (SEIMI)

This denotes the start of the virtual simulation and testing. The SEIS process stages 1-4 can be considered as being covered by the systems lifecycle process the information of which can be seen in the previous four sections. From this point forward the SEIMI process is considered to be within the modelling and simulation boundary. The transition across the design and simulation boundary can now be crossed as all prerequisites have been satisfied.

4.3.9 Desire to Test Something

The desire to test a particular aspect has been identified. This was apparent from the very start of the project. The desire is to ascertain what the effects are of joining the brake system with the steering system on the overall vehicle handling.

4.3.10 Design of Experiment

This experiment is to be a comparative test between two mutually exclusive events. The first being where the steering system is given an input and the brake system is not interconnected or even activated. The second case is where the

steering system is given the same input, however the braking system is interconnected and capable of being activated to aid with turning. The effects of braking while steering is to be investigated by comparing it to steering while not braking and any resultant change in trajectory.

Hypothesis to be Tested
By linking the steering system and the braking system under a single controller will result in a reduction in error between the user input and the vehicle trajectory.

Error in this case is considered to be the difference between the user input and the trajectory that the vehicle takes.

Factors within the experiment to take into consideration
As part of the experiment there are factors that need to be taken into consideration:

- The steering inputs are intended to be a single sine input that is often used to simulate lane changing manoeuver
- The characteristics of the vehicle including the size or the body, and wheels are to be changeable in the models
- The experiment is to be able to be repeated at different longitudinal speeds of the vehicle
- All constants and parameters which are not concerned with the activation of braking systems are to be the same between tests.

Variables That May Change During Tests
There are variables that have been identified that may change through the test:

- The steering input values
- The speed of the vehicle
- The retardation force of the brakes (during braking test).

The variables that are considered to be held constant:
Some identified variables are considered to be held constant across both test cases.

- The size shape and mass of the vehicle
- The weather
- The coefficient of friction between the tyre and the road surface
- The road surface
- The overall steering input profile
- The range of braking modulation
- The range of possible steering angles
- The initial speed of the vehicle before steering input initiated
- The incline that the vehicle is traversing.

4.3.11 *Define Assumptions of Experimental Set Up*

As part of this experiment there are assumptions that need to be taken into consideration across all of the component models.

- The vehicle is a three dimensional object with a length breadth and height.
- Gravity is to be considered to be 9.81ms^{-2}
- The tyres can slip on the road surface
- The vehicle is moving and the surface it is sitting on is static.

4.3.12 Define Simulation Boundaries

The boundary of what is to be considered as part of the simulation has been captured in Figure 4.10 below.

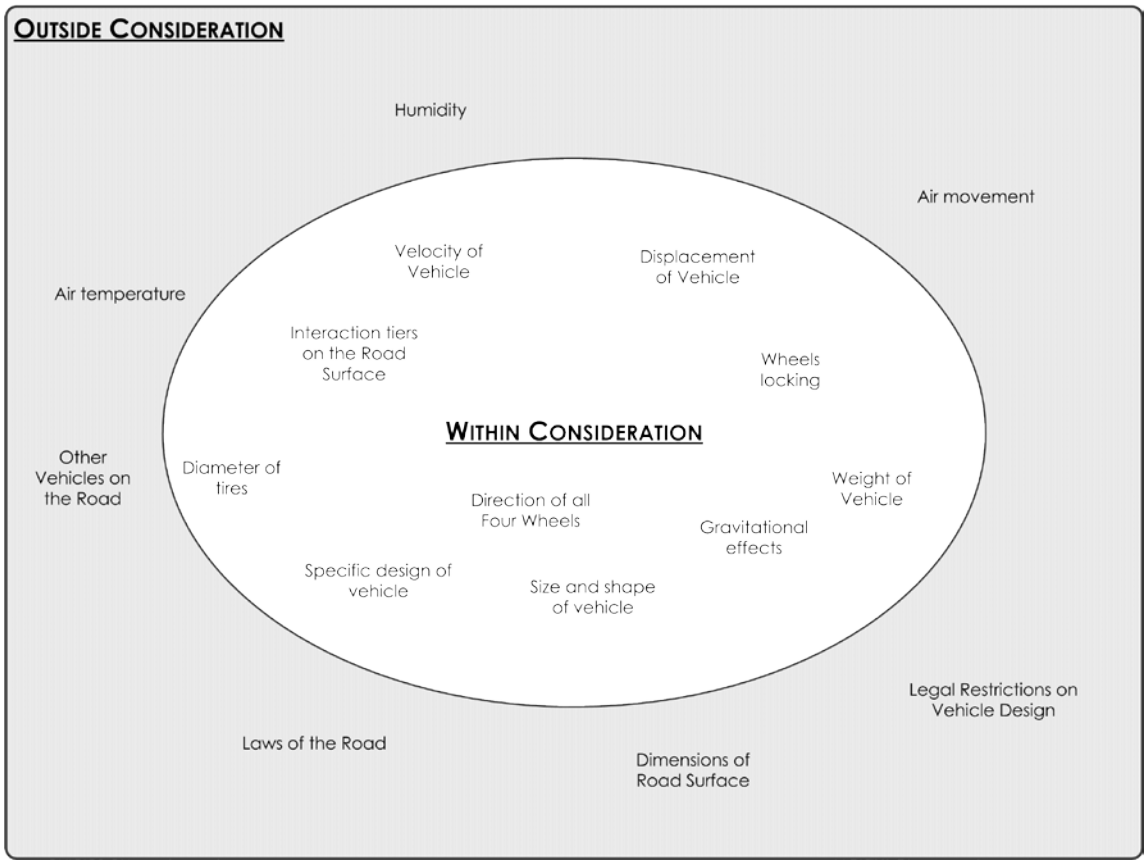

Figure 4.10: Simulation boundary showing what is within and outside of consideration.

All work is to focus around what is captured within the ellipse in Figure 4.10. Each of the factors that have been identified as being outside of consideration can be disregarded or investigated at a later date in a subsequent simulation

4.3.13 *Simulation Requirements*

The simulation and modelling requirement writing guide as defined in section 3.4 was used to create the requirements for the simulation. Due to the system

requirements not being sufficient to express the simulation experiment, simulation specific requirements are needed.

Means of Communication
To not distract from the process itself, textual requirements have been used in this application.

Requirements

1. *The simulation is to capture the behaviour of a vehicle which has a combined ABS and steering system.*
2. *The simulation is to capture the behaviour of the vehicle with a sinusoidal steering input.*
3. *The simulation needs to be run multiple times with the speed of the vehicle changing across operational speeds from 10kH^{-1} to 115kH^{-1}*
4. *The model is to contain; Driver input, ABS System, and steering system.*
5. *The outputs of the component systems are to be recorded.*

Simulation Constraints

A. The total run time of the simulation should take less than five minutes to fully execute.

B. The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.

C. No specific computational hardware or peripherals are to be used.

D. The modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.

E. If LabVIEW or Matlab is used, only a single license may be used.

F. The output results of the simulation are to be saved in a file format that can be interrogated at a later date.

G. All component parts are to be in the public domain.

4.3.14 SET STANDARDS IF THEY ARE TO BE USED

Various standards have been investigated, however a decision was made to not implement any of them.

4.3.15 VERIFICATION OF REQUIREMENTS

Verification of Requirements
This is the verification of the simulation requirements as part of the requirements writing stage and is not to be confused with a verification stage of the Systems Engineering in Integration of Simulations process.

A simple Red Amber Green (RAG) method was selected as the means of assessing the requirements against the requirements writing guide. Within the analysis Red defines that the requirement is in direct contradiction and requires logical re-gwork, Amber defines that it requires some rework, Green defines that it is complicit with the requirements writing guide.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a vehicle which has a combined ABS and steering system.			✓
2) The simulation is to capture the behaviour of the vehicle with a sinusoidal steering input.			✓
3) The simulation needs to be run multiple times with the speed of the vehicle changing across operational speeds from 10KH-1 to 115KH-1			✓
4) The model is to contain; Driver input, ABS System, and steering system.			✓
5) The outputs of the component systems are to be recorded.			✓
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) If LabVIEW or Matlab is used, only a single license may be used.			✓
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓
G) All component parts are to be in the public domain.			✓

Table 4.26: Verification RAG assessment of simulation requirements.

With all of the requirements passing the RAG validation the work can continue onto the next stage.

Verification of Standards

Due to the lack of application of a standard the verification stage concerned with the verification of the choice of a standard can be omitted.

4.3.16 PRELIMINARY ARCHITECTURE

The preliminary architecture for the simulation is depicted in Figure 4.11 below. The architecture captures the components systems that have been identified.

Some indication as to the communications that will be needed have also been captured.

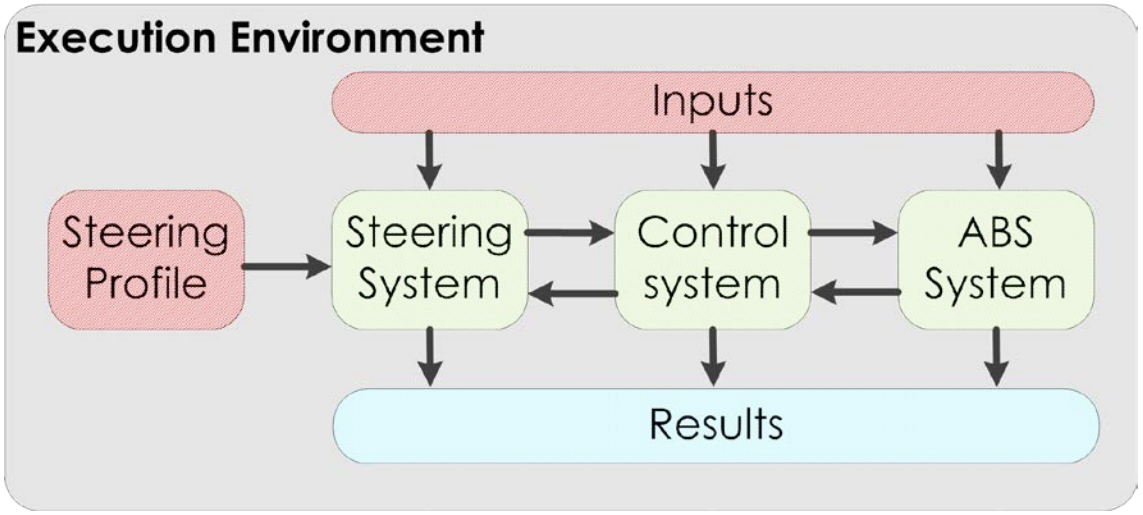

Figure 4.11: Preliminary architecture for the simulation.

The preliminary architecture intends that there will be communication between the various component parts. However the steering profile is unlikely to be the only input. To record the results of the component parts there will be the need to pass the results to a component to do so.

4.3.17 VERIFICATION OF PRELIMINARY ARCHITECTURE

With a preliminary architecture defined it can be verified against the requirements for the simulation.

A simple Red Amber Green (RAG) method was selected as the means of assessing the preliminary architecture against the simulation requirements. Within the analysis Red defines that the requirement is in direct contradiction and requires logical re-work, Amber defines that it requires some rework, Green defines that it is complicit with the requirements.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a vehicle which has a combined ABS and steering system.			✓
2) The simulation is to capture the behaviour of the vehicle with a sinusoidal steering input.			✓
3) The simulation needs to be run multiple times with the speed of the vehicle changing across operational speeds from 10KH-1 to 115KH-1			✓
4) The model is to contain; Driver input, ABS System, and steering system.			✓
5) The outputs of the component systems are to be recorded.			✓
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) If LabVIEW or Matlab is used, only a single license may be used.			✓
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓
G) All component parts are to be in the public domain.			✓

Table 4.27: Verification RAG assessment of preliminary architecture.

With the architecture being found to be complicit with the simulation requirements the next stage can be moved onto.

4.3.18 PRELIMINARY SIMULATION DESIGN

The behaviour of the parts defined by the architecture can now be broadly defined.

User Defined Input

A user interface is to be used whereby the user can input parameters and view the results of the simulation. This is to be a single input interface for all of the component parts.

Steering profile

The inputs steering profile is to be a sinusoidal waveform.

Steering Control
Model that represents a steering system, which takes user inputs and assists the user alter the angle of the wheels. The user inputs are too represented by inputs from the steering profile component.

ABS System
Model that represents the behaviour of an existing anti-lock brake system. The model is to capture the behaviour of the modulation of the brake pressure to stop the wheel from locking completely.

Control System
A controller that takes inputs and gives outputs to and from other system components as required.

Results
A component that takes inputs from other simulation components and saves them in a suitable format.

4.3.19 VERIFICATION OF PRELIMINARY SIMULATION DESIGN

The preliminary design can be verified against the simulation requirements. The RAG assessment method has been used to analyse whether the preliminary design metes the simulation requirements. This RAG assessment can be seen in Table 4.26.

REQUIREMENT	RED	AMBER	GREEN
1) The simulation is to capture the behaviour of a vehicle which has a combined ABS and steering system.			✓
2) The simulation is to capture the behaviour of the vehicle with a sinusoidal steering input.			✓
3) The simulation needs to be run multiple times with the speed of the vehicle changing across operational speeds from 10kH-1 to 115kH-1			✓
4) The model is to contain; Driver input, ABS System, and steering system.			✓
5) The outputs of the component systems are to be recorded.			✓
A) The total run time of the simulation should take less than five minutes to fully execute.			✓
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.			✓
C) No specific computational hardware or peripherals are to be used.			✓
D) The modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.			✓
E) If LabVIEW or Matlab is used, only a single license may be used.			✓
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.			✓
G) All component parts are to be in the public domain.			✓

Table 4.28: RAG assessment of the preliminary design.

The RAG assessment in Table 4.26 shows that the preliminary design meets the simulation requirements and so it is possible to proceed to the next stage.

4.3.20 ARE THERE ANY EXISTING SIMULATIONS AND MODELS

The elements that are composed as parts of this system are well established. It is recognised that in a genuine engineering setting there would be existing models and simulations that could be of use in this situation. To emulate this repository a literature review was conducted and it is well known that models exist. This will impact the SESEM process but that will be explained in due course.

4.3.21 SYSTEMS ENGINEERING OF SELECTION OF EXISTING MODELS SESEMS

The SESEM is a sub-process that is concerned with aiding in the selection of existing models. As the potential for existing models being of use then this process can be implemented.

1 Preliminary Simulation Design
The overall goal of the system being designed as well as the simulation being designed is well understood. There is also an intended architecture and design for the simulation being produced. The preliminary requirements for the SESEM process can considered satisfied.

2 Boundary of the Existing Model Selection Process
A secondary check has been conducted to assure that all of the prerequisites have been satisfied. From this point on it is considered that the SESEM process is the structure that is being followed.

3 Assess the Model and Simulation Landscape
An assessment has been made of the ways in which the interactions that are to be investigated have previously been modelled and simulated in the past. Information has been gathered as to the mathematical tools and representations which may be of use.

4 Are Potential Models Available?
From the assessment of the modelling and simulation landscape there are a number of potential models that could be of use.

6 Is this a New Product/ Platform?
The component sub-systems that forms the bases of the proposed system are well established and so too are the platforms. Hence the decision to consider that this is not a new product or platform has been made.

16 Locate Previous Product Models
An assessment of the various off the shelf models that could be of use have been collected. The models that have been located include:

- Modeling an Anti-Lock Braking System
- Modified Anti-Lock Braking (ABS) Model
- Vehicle Body
- Power-Assisted Steering Mechanism
- Simple 2D kinematic vehicle steering model
- Tyre simple
- Tyre (Magic Formula)

There is more than one tyre and steering models that could potentially be of use. However at this stage having more than one model is not a hindrance.

17 Available Model Documentation
For each of the identified model a simple assessment of whether there exists sufficient documentation available has been conducted. If the documentation is not available or has insufficient detail, is there someone in the team that can understand the model explicitly has been ascertained. The results of this assessment are captured in Table 4.27 below.

Model Name	Documentation Available	Team Understanding	Suitable
Modeling an Anti-Lock Braking System	✓	–	✓
Modified Anti-Lock Braking (ABS) Model	✘	–	✘
Vehicle Body	✓	–	✓
Power-Assisted Steering Mechanism	–	–	–
Simple 2D kinematic vehicle steering model	✓	–	✓
Tyre simple	✓	–	✓
Tyre (Magic Formula)	✓	–	✓

Table 4.29: Representation of the assessment of the documentation availability. Three symbols are used; ✓ full, – partial, and ✘ not at all.

Any models or simulations which do not have sufficient documentation or available understanding are deemed unsuitable. It is worth noting that Power Assisted Steering Mechanism only has limited documentation whether what is present is sufficient will be discerned later in the process. The team understanding is partial for all of the selected models.

19 Does the Model Match a Section of the Simulation Requirements?

For Each candidate model it is evaluated against the simulation requirements. A RAG assessment of each of the potential models has been captured in the Table 4.28 below.

REQUIREMENT	Modeling an Anti-Lock Braking System	Vehicle Body	Power-Assisted Steering Mechanism	Simple 2D kinematic vehicle steering model	Tyre simple	Tyre (Magic Formula)
1) The simulation is to capture the behaviour of a vehicle which has a combined ABS and steering system.	A	R	A	A	A	A
2) The simulation is to capture the behaviour of the vehicle with a sinusoidal steering input.	R	R	A	A	A	A
3) The simulation needs to be run multiple times with the speed of the vehicle changing across operational speeds from 10kH-1 to 115kH-1	G	G	G	R	G	G
4) The model is to contain; Driver input, ABS System, and steering system.	A	R	G	A	A	A
5) The outputs of the component systems are to be recorded.	A	A	A	A	A	A
A) The total run time of the simulation should take less than five minutes to fully execute.	G	G	G	G	A	G
B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.	G	G	G	A	G	G
C) No specific computational hardware or peripherals are to be used.	G	G	G	G	G	G
D) The modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.	G	G	G	G	G	G
E) If LabVIEW or Matlab is used, only a single license may be used.	G	G	G	G	G	G
F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.	A	A	A	A	A	A
G) All component parts are to be in the public domain.	G	G	G	R	G	G

Table 4.30: RAG assessment of the selected potential models. Red (R) does not comply, Amber (A) partly complicit, and Green (G) fully complicit.

20 Assess if Individual Models Can be Modified

Not all models can be modified for a number of reasons that are detailed in section 6.3. Each of the selected models are assessed as to if they can be modified. The extent to which the can be modified is not in conjecture at this point it is full access modification or nothing.

MODEL NAME	CAN MODEL BE MODIFIED
Modeling an Anti-Lock Braking System	✓
Vehicle Body	✘
Power-Assisted Steering Mechanism	✓
Simple 2D kinematic vehicle steering model	✘
Tyre simple	✘
Tyre (Magic Formula)	✘

Table 4.31: Assessment of whether the potential models can be modified. The assessment is a simple yes (✓) or No(✘).

14 Selected Models Unusable

For the issues with the documentation and or the ability to modify the existing components the following existing models are considered unusable.

MODEL NAME	MODEL USABILITY
Modeling an Anti-Lock Braking System	✘
Vehicle Body	✘
Power-Assisted Steering Mechanism	✘
Simple 2D kinematic vehicle steering model	✘
Tyre simple	✘
Tyre (Magic Formula)	✘

Table 4.32: Assessment of whether the potential selected models are usable. The assessment is a simple yes (✓) or No(✘).

None of the identified models are compatible for the intended simulation. If a model cannot be modified it has to meet all of the requirements exactly. For ‘Vehicle Body’, ‘Simple 2D kinematic vehicle steering model’, ‘Tyre simple’, and ‘Tyre (Magic Formula)’ require modification to make them complicit with the requirements.

With the understanding of two remaining models that is present in the documentation issues became apparent. The ‘Modeling an Anti-Lock Braking System’ is unrealistic and only meant as a demonstration model. The power assisting model documentation does not cover the assumptions that were used during its creation. As well as many of the components are ideal in nature and

not representative of the systems that would be implemented in the design. The SESEM process dictates that in this instance it will then be necessary to locate more new models or follow the path of creating new models.

This case study was initiated as being a facsimile of the work that was conducted with industrial models and simulations. In the industrial situation there were only a finite number of models and simulations within the repository that was available for this work. The result of the analysis that was conducted in the industrial case study reached the same point in the process as the selected models and simulations did in the academic automotive example. It is recognised that in the academic automotive example it is conceivable that other model or simulations could be found, however this was not the case in the industrial situation. The process dictates that in such a position if existing models cannot be located then new models are to be made. To produce such models and simulations subject matter experts would be needed to create the models and simulations, which would comply with the simulation requirements. This changes the task from integrating existing models and simulations, to developing a new simulation which may not even need to be integrated. Developing a new simulation from a blank slate is not the focus of this research, as that is a different topic. For this reason this case study did not progress any further.

4.4 Evaluation of Methods

The two fully worked case studies provide a reference point to which meaningful evaluations of the proposed methods can be made. Throughout the work that was conducted for the case studies strengths and weaknesses of the proposed methods became apparent. These identified strengths and weaknesses are discussed below.

4.4.1 Strengths of the Proposed Methods

From conducting the case studies, clear strengths of the proposed methods became apparent when compared with the current identified methods as discussed in section 2. Many of the identified strengths of the proposed methods originate from the structure that they bring to the integration task. This structure breaks down the seemingly chaotic integration task into repeatable stages that otherwise rely on the engineering experience and understanding of those taking part in the simulation task.

The way in which the process has been created is such that it does not dictate how to conduct each task, but rather it is more of a requirement for the thought process of the user. This allows for the user to select the most appropriate tool for the job for their domain to conduct the required analysis at that stage. As demonstrated in the two case studies, different tools were used for the same process element dependent on the information that needed to be processed. In different situations some tools will inevitably be more suitable than others and having this flexibility increases the potential area where this process remains relevant.

Having stages in the integration task make it possible for more than one engineer to work on the task simultaneously. In many cases it is possible for multiple engineers to work on the same task element at the same time. For case study two, elements one to six were effectively conducted by more than one engineer as the premise of the test came from negotiations with industrial engineers and academics while the rest of the stages were completed solely in an individual academic environment. The decision stages of the processes (traditionally a problem for group work) do not necessarily rely on just an individual making the decision as they could be conducted in a meeting setting or even using another systems engineering tool. At any of the stages it is possible for the process to be handed to another engineer as long as they are also familiar with the process and all of the previous stages have complete sets of the recommended documentation. This also allows for breaks in the work to be possible. While conducting case study two there was in fact a three week gap. However it took very little time to pick up where it had been left off. Having such a process allows for greater flexibility of engineers time.

Specifying the purpose of the simulation at the start acts as a reference point throughout the rest of the process. This reference point is used to formulate the

basis of the verification to which the whole simulation is subjected at various points. This static reference point has been of benefit as it allows for the simulation to be formulated in layers of complexity and functionality, meaning that the sources of the component parts to potentially be widely different, while keeping focus on what the simulation is attempting to achieve. In case study two there was the real potential for the work to progress and produce a simulation regardless, however the defined purpose of the test ensured that the work remained focused and hence halted. Without the static reference point this may not have been the output.

The requirements writing guide allows for structure and similarity across the requirement sets. This was found to be beneficial across both case studies as it improves not only the quality of the requirements but also the effectiveness of more simple analytical tools. Having the simulation and constraints separate ensures that consideration is not only focused around functionality but also to the computational capabilities of the resources available. For the second case study the simulation constraints could have caused a real problem if certain potential simulation components were used.

The integration tables have demonstrated within the case studies to be one of the most valuable outputs of the proposed methods. The tables capture of the semantic information needed for meaningful integration. They are only completed once an understanding has been gained regarding the model which will be the subject of the tables, and only if the likelihood of the model being of use is high, waiting to this point to complete the tables is due to the resources required to conduct the task is considerable. Having such information stored in readily readable tables means that any two models with completed tables can be assessed to ascertain if they are semantically similar enough for potentially meaningfully integration. Critically this makes it possible to assess whether two or more models have the potential to be meaningfully integrated without the original modellers being involved. This is all before actually investing the time into not only the integration task but also the time needed to ascertain if the results of the integrated simulation are meaningful. This result is a considerable reduction in the resource overhead that it takes to produce a meaningful (with high confidence), integrated simulation when compared to current methods if rework is required. The integration tables facilitate many models being compared with the information stored in them being uniformly structured removing the need for integration engineers to hold all of the relevant information in their own working memory. Even in the limited example of case study one, the amount of information that was required to have available was such that the tables became of considerable benefit.

If the model integration process does not go any further than the generation of the integration tables it does not mean that the work is of no use. It was found that in both case studies that the integration tables were far more thorough and concise when compared against the documentation that was supplied with the

models. Once the integration tables are generated they can be stored for others to potentially use at a later date. The semantic information held within them will still be relevant for later work.

The work that was conducted in case study two highlights the processes ability to locate the potential integration issues of existing models on a semantic level. This is of even more significance as it is capable of this even when the person following the process is not a domain expert of the potential simulation components. This validates that the engineer does not necessarily need to be from the domains of the potential models to be able to assess if integrating them could be meaningful or not. This capability was also found in case study two, interestingly at the exact same point of the process. This indicates that the understanding of the models and the comparison against the simulation requirements is the first real sorting stage of whether or not a potential component is suitable.

The work involved with the setting up of the experiment for the first twelve elements of the Systems Engineering in Integration of Simulation process took a considerable amount of time. However having a designed experiment is of significant use when formulating the verification of the simulation. The output of the first twelve stages of SEIS give a static reference point which can be used for the verification task later on. Having a reference point means that if there is a variation from said reference it can be identified and corrected. The reference point critically allows for verification to be more than just the validity of the data transfer between components being acceptable. Without such a well-established reference point the validity of the verification testing could be brought into serious doubt.

4.4.2 Weaknesses of the Proposed Methods

While conducting the case studies it was not without issue and weaknesses of the proposed processes became apparent. Many of the issues stem from the increase in time that it takes to get to the point where programming starts to take place. This is due to the trade-off between the time that is invested in getting the integration correct the first time, rather than the time after the simulation is completed and the inevitable rework that has to be done.

When conducting the proposed methods it became apparent that the amount of effort and time that goes into a stage is not represented by the textual output. This could be a potential issue when managing engineers using these processes. Elements of the SEIS process where this is particularly prevalent are 1, 7, 11, 19 and 26. For the SESEM process it is elements 3, 8, 10, 11, 18, 23, 24, 25, and 28. From the number of tasks that require considerable thought and work with little visible output may trouble some managers.

The task of locating previous models in this process is but one single stage however it is a non-trivial one. It would benefit from further research and

guidance as to how the engineer could go about locating these existing models and simulations. This stage was one of the reasons why the second case study ran into the problems that would later halt the progress of the work.

When conducting the SESEM process during both the case studies a potential issue became apparent regarding stages 12, and 19. At this stage existing models are held against the requirements for the simulation being designed. There is hence the potential need for requirements for the potential design. As the overall requirements of the simulation may not be fully applicable in this situation as the potential components are unlikely to be applicable at that stage. The user has to make the decision as to whether the potential component is of worth to continue using and investing in potentially reworking of the component.

4.4.3 The Effectiveness of the Proposed Methods

The application of the proposed methods provides structure to the integration process and produces a situation where there is both a potential solution and a means of conducting meaningful verification. This however does come at a cost of time and resources. In case study one the methods proved that they could be used to produce a simulation form existing models, whereas in case study two the processes highlighted that there were real serious semantical differences and potential problems with using the identified models. This indicates the effectiveness of the proposed methods.

4.5 SUMMARY OF CASE STUDY TESTING

The two case studies demonstrate how the proposed methods in section 3 can be implemented. The first case study is a fully worked example from product concept through to the point where a physical prototype can be made and the virtual testing validated. The second case study represents an automotive example where the selected component models and simulations were identified as having significant semantic differences. The extent of these sematic differences resulted in any potential integration being meaningless.

The critical findings from the two case studies are: the value of having a defined reference point that can be used through development for verification and validation, the ability to reliably identify semantic differences between the potential component models and simulations, and that the methods allow for non-domain experts to be able to make an assessment of the suitability of potential integrations before the work is put in to integrate and test the full simulation. However the greatest weakness of the proposed methods is the time it adds to virtual simulation and test. However there is the indication that automation technologies such as NLP could vastly reduce the time it takes for current repetitive deductive tasks, see section 5.

5 NATURAL LANGUAGE PROCESSING

5.1 NATURAL LANGUAGE PROCESSING

In this work the language that we speak, read and write to convey meaning is defined as natural language. It is to be noted however that not all language can be considered to be written using natural, more on this topic later. Hence this paragraph is considered to be natural language. This research focuses on the written rather than spoken word. Written words are the primary method used to convey ideas within the identified problem space. Reading natural language into a computer, conducting analysis, and gaining additional information is the basis of Natural Language Processing (NLP). This is not as straightforward as it may first seem. Though there are many PhD studies that research solely NLP, it is used as one of many methods in this study.

The general stages of an implemented NLP application, when machine readable files are available, are shown in Figure 5.1 below. However to get to a functional NLP application, it will require training or knowledge encapsulated in code to ensure the output of the analysis stage is meaningful. Using a computer to perform the activities shown in Figure 5.1 is not a trivial exercise.

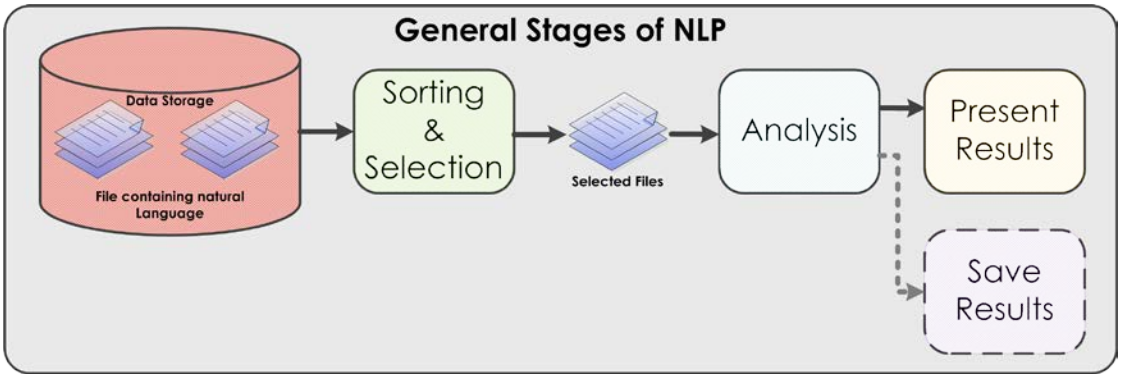

Figure 5.1 The basic stages of NLP when machine readable files are available. The arrows denote the progression of the stages. There are some applications of NLP that only present the results to the user whereas other applications save the results for further analysis this is denoted by a dotted line.

Within the NLP literature there are two distinct trains of thought that become apparent. The first is in academic research that looks to further the capabilities of NLP, though they may not have an immediate application. In contrast the second comprise the more application-based studies and as such tend to use well established technologies that have a clear application focus. In this review both perspectives will be investigated.

The challenge of NLP originates in the way in which computers work. The inputs to a computer are analysed and set operations are conducted dependent on the inputs that it receives. When considering natural language as an input to a computer, the range of possible inputs is vast and the meanings inconsistent (as discussed later). Although natural language is currently recorded using computers this does not mean that the meaning of the words are understood and the overall text comprehended. One of the more familiar examples of NLP is in the grammar checking that word processors conduct. The purpose of such

grammar checkers is to highlight to the author sentences that are not compliant to a particular set of grammar rules. Such functions in word processors take the natural language, analyse it, and display to the user potential errors. In some of the more advanced programs, it takes the identified error and offers potential corrections.

5.1.1 Current Capabilities of Natural Language Processing Technology

To assess the current capabilities of NLP is a difficult task, as in pockets it has become very successful and accepted as the method of choice such as language translators, whereas in other domains there are still research challenges halting potential applications.

Current Strengths

If large training sets are available the similarity between texts can be identified. The development of general purpose taggers has reached a stage where the results have a high degree of accuracy, meaning that each word and piece of punctuation can be identified. The speed by which NLP tasks can currently be performed is such that thousands of documents can be analysed in seconds.

Current Caveats

Often NLP applications are limited to only being effective in a selected domain. Whenever the domain changes the methods require retuning or rewriting. This results in some applications getting a bad reputation as they are being used in situations for which they were not designed. Dependent on the analysis, they may appear to be much slower than other computation processes. This is due to the sheer number operations that some NLP applications compute. Expectations of users as to the capabilities of NLP applications is an interesting topic in itself but due to the perceived simplicity of the tasks many people believe that NLP must be far more developed than it actually is. This can result in disappointment or undervaluing the achievements of NLP projects and their applications.

5.1.2 Understanding and Comprehension

One of the key challenges that still faces NLP is the comprehension of what the individual words mean and then what is meant when these words are assembled in a specific order with specific punctuation. This stems from computers being deductive rather than inductive machines. Logical descriptions of language always fall short of the complexity that is possible when using it. Hence any machine that uses such a purely logical progression will eventually fail to capture a particular nuance or combination that was not considered when the system was implemented. For such an understanding of what words mean would require a level of general artificial intelligence (AI). There are issues with creating AI as well, specifically such general purpose AI that would needed to handle the multitude of possible words and phrases that would not be captured by any training program or by through direct programming. There is the deeper issue of

conceptually knowing or defining how (or if) a computer understands the text [90]. This then raises an interesting problem as how is it possible to conduct analysis if there is currently no way of getting a computer to comprehend what is meant by the words. The issue then becomes how to analyse the semantic meaning of the text without understanding what the semantics are in the first place.

There is something that is surprisingly uncommon in the linguistic field, namely a clear definition as to what is meant by natural language semantics, which is,

> *"the meaning expressed within Natural Language at the morphological, lexical, syntactic, and discourse levels."*[91].

As a definition this may not appear that clear due to the vocabulary that is used. Morphology is considered to be the structure of the words, as morphemes are the smallest meaningful parts of language. Therefore morphology is the way in which the morphemes are combined and structured. Lexical level refers to a language's vocabulary. Syntactic levels are the rules which are used to assemble words in a meaningful way. This definition is clear and concise and is one that will be applied in this work when considering natural language. It is to be noted that this is not the same definition of 'semantics' when considering models and simulations.

With the definition of semantics defined for language there are issues with semantics that can be discussed without confusion as to what the issues are. The capture of the semantics of a language is one of the more difficult aspects of NLP. The English language has significant semantic content. Individual semantics will not be discussed in this study; however the effects and issues caused by the differences will be covered.

5.1.3 NATURAL LANGUAGE PROCESSING TECHNOLOGIES

Over the last ten years there has been something of an explosion in NLP research that has resulted in a mix of technologies that have been applied to conduct different forms of analysis on different linguistic domains. There are those who use set theory and pattern analysis to analyse [90]. For the words to be sorted into sets, the words first need to be categorised into pre-defined categories. This task of categorisation is often completed by using a tagger. Across the literature many different taggers have been designed and tested. However, often the inner workings of the tagger are not fully disclosed. When a sentence is tagged each individual word and piece of punctuation is analysed and given a tag which represents a pre-determined category for example 'jump' may be tagged as 'VB' meaning that it has been categorised as a verb. Many methods then analyse the types, frequency, and relative positions that the tags appear in the text.

One method to develop taggers that have flexibility is to use fitness based mathematic constructs trained using suitably large domain specific data sets. There have been some studies where there has been limited success such as when a trained tagger from one domain is retrained in another without rule re-scripting and the output results remain valuable [92]. This is a purposeful finding as it shows that as long as significantly large corpora are available a core set of algorithms could be successfully deployed across multiple domains. From a business context this would mean that any tools (implementation of algorithms) would be purchased once and then trained on their data sets allowing for the tool to be used in more than one department or subject area. From the literature the magnitude of the training sets have many thousands of entries. The cost of producing such corpora is sizable and hence many are in a general format. Some specific corpora that are open to public use are few and far between. There are even fewer engineering specific corpora and none could be found that were applied to the topics of this thesis.

Some research projects such as those from the Stanford Natural Language Processing group use a mixture of rule-based, statistical, and deep learning methods. This is an example of a group using different tools to suit the different applications that they are researching. They have demonstrated that the different NLP technologies all have their strengths and weaknesses as such they apply what they see to be the most appropriate tool for the job [93].

There have been attempts made to capture natural language in ontologies. However capturing all of the semantics of the natural langue causes somewhat of a challenge as these semantics need to be accurately mapped onto the strict semantics of the ontology. This results in what some refer to as an unnatural language that does not necessarily capture all of the semantics that the natural language has [91]. They go further to discuss how these semantic gaps are currently one of the most significant obstacles for meeting their customers and users' expectations of the NLP capabilities. As English is an ever changing and evolving language the semantics that have been captured will need to change and evolve the language to remain 'correct'. This further complicates the capture of the semantics. As the creation of the mappings is also a timely task such changes would also be a significant time investment for any organisations looking to maintain a system which uses this technology. There are issues surrounding the extent to which ontologies can capture the meaning of text [91] as there are limitations to what information can be held in current ontologies. It is still a challenge as to how to capture the semantics of the effects of connectives. Due to these difficulties in mapping semantic information into ontologies there are still many research projects that are concerned with attempting to capture natural language.

There is an area of NLP that is concerned with ascertaining if the extent to which text is semantically similar [94]. Comparison studies often require large manually constructed data sets to allow for a meaningful validation of any proposed

method. Such data sets are not often readily available. Due to the resources that need to be invested into the creation of such data sets they are often not released into the public domain. However within the literature there are examples where data sets are made available for other researchers to test their algorithms against. In one such example they used 2000 sentence pairs which were manually rated for similarity. They then set up an open call for other researchers to test their methods against the data set. The data set was split into two sections 50% train and 50% test. Out of the 35 teams that took part

> *"The best results scored a Pearson correlation of over 80%, well beyond a simple lexical baseline with 31% of correlation."*[94]

This demonstrates that there are methods available which can give a relatively reliable metric as to the semantic similarity between texts.

When there are new large data sets there are those who propose the use of logical rules and tagger as a means of NLP. These have the advantage of not needing additional data sets to train statistical methods. Success has been achieved in a study where a tagger was used in conjunction with 79 logical rules [95]. This application was focused around extracting requirements information from project documentation. This indicates that when there is only a limited source of data this approach can be of use. It also demonstrates how the applicability of NLP can be tested on a domain before the investment is made for statistical methods and the required data sets.

This short review provides a snapshot of the applications for which a variety of NLP technologies have been applied. However there are certain themes that have become apparent. Statistical methods that use elements of learning often prove to give the most reliable results. However such statistical methods often require vast amounts of manually constructed data that is a resource heavy exercise in terms of cost and time.

5.1.4 Language and the Challenges it Brings to Natural Language Processing

The study of linguistics is a rich and well established field in its own right. Some of the analysis and research has a real impact on attempting to formulate NLP algorithms. One of the key areas is the discussion as to if English is descriptive or prescriptive in nature. A detailed discussion of descriptive and prescriptive terms can be found in Section 5.2.1. There is a seminal text that is referred to throughout the linguistic community *"Halliday's Introduction to Functional Grammar"* [96] In this work the English language is represented in a formal logic with particular focus on the way in which clauses are constructed and the behaviour of classifications of words behave within them. Such work indicates that even with the descriptive nature of the language there are still logical constructs that reflect the underlying structure of the language. This work has been cited as being an example where formal logic rather than statistical methods can be

used to analyse the English language. It is however a logical progression, as if the language changes over time some of these rules and logical statements may become less applicable or even irrelevant as the language develops.

One of the identified problems within engineering text be that from requirements, specifications, documentation, or otherwise, is the ambiguities that result from natural language. For applications of NLP in engineering the most common example is requirements analysis.

The types of ambiguity that are the cause of the issues have been identified as *'lexical'* (words can be more than one type e.g. noun and verb),*'structural'* (knowing which words relate to which concept previously mentioned), *'semantic'* (words having more than one meaning), *'pragmatic'* (the possibility of a collection of words meaning two or more different concepts), and *'referential ambiguity'* (knowing what is the subject) [90]. These types of ambiguity are present in natural language and even with the best efforts of requirements engineers, it is difficult to eradicate all ambiguities. This also highlights a significant challenge that any analysis tool has to overcome.

The issue of domains is an interesting one, as each domain has its own definitions and grammatical structures. Within the literature the adaption of existing NLP tools to new domains has been identified as being a costly and time consuming activity [92]. This shows both the lack of flexibility of current methods as well as the scale of differences between the language used between domains. When reviewing the literature it became apparent that many researchers focus on a specific domain or sub-domain to reduce the variation, and hence complexity of the task that the NLP has to contend with. Some sources (e.g [92]) state that it is well known that by reducing the size of domain that an NLP application is designed to work within is a means of improving the accuracy of the analysis and hence results produced.

5.1.5 Formal Languages and Natural Language Processing

There are examples of formal languages that have been developed that lend themselves more favourably to language processing, an example of this can be seen in the OMGs Semantics of Business Vocabulary and Business Rules [97]. However as soon as a formal language is used it is no longer natural language and so loses the flexibility and richness of natural language. This can be problematic at the early stage of a project and requires all involved to be fluent with the formal language. From a practical point of view even if formal languages are used, translation at some point needs to be made between natural language and formal language [95]. This shifts the burden of ambiguity from across a project to those who implement the formal language. Suitable verification needs to be implemented to ensure that the semantics of both the natural language and its formal representation is representative. Formal languages are not significantly referenced in this research as they were not identified to be part of the issues that were defined in the initial problem space

(See Section1.2.3). There is a field of research that looks at mapping natural language into formal languages as well as expressing complex designs with formal languages, however they are outside the scope of this work.

5.1.6 USE OF LANGUAGE IN ENGINEERING DOCUMENTS

As has previously been identified, language is not used uniformly across all domains or even the same way in the same domain. It has been proposed in standards such as ISO/IEC/IEEE 29148 [27] that there is a structure which requirements take. The standard states that there are three forms that requirements are to take:

1) [Condition][Subject][Action][Object][Constraint]
2) [Condition][Action or Constraint][Value]
3) [Subject][Action][Value]

In the domain of language analysis, these three forms all contain a noun and a verb in the same phrase. The noun-verb phrase is also referenced throughout the systems engineering literature. It is proposed that NLP may be capable of identifying requirements by searching for noun-verb phrases. It is however recognised that not all phrases that have a noun and a verb in are going to be requirements but it is one potential way to identify a set of noun-verb phrases a subset of which may be requirements.

Within each engineering domain there is a sub-set of language that is used by those within the domain. To enable any NLP process to be effective across all domains it may necessary to map what the words that that domain use and the definitions that are used. The ways in which the domain languages overlap are shown in Figure 5.2 below.

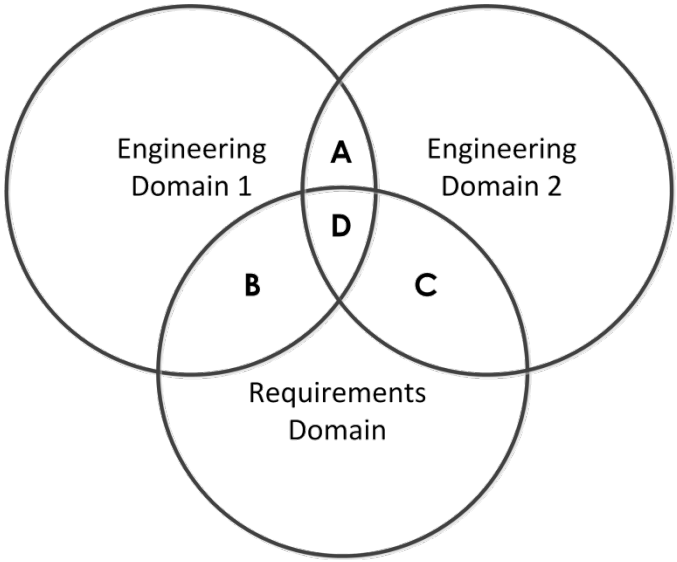

Figure 5.2: The overlap of engineering discipline languages.

The mapping of the engineering domains may be potentially more of a challenge than it first seems. It is relatively straightforward for terms that do not overlap between domains as in this case could be considered to have a relatively stable definition however the issues arise when the word falls in any of positions found in Figure 5.2 denoted by letters A,B,C and D. Position D represents the words that have three distinct meanings that each engineering domain has assigned to it. Part of the challenge is assessing where words fall within multiple definitions and then how to use surrounding semantics to ascertain what the author intended the definition to be.

5.1.7 Current uses of Natural Language Processing in Engineering Projects

NLP technology has reached a maturity where it is being used in many aspects of engineering. Due to the way in which engineering is domain-based, some of the applications are more wide spread than others. Often the applications of NLP are for information extraction from a repository [92].

The extraction of data from natural language has shown itself to be a valuable tool for the creation of databases [90]. Such applications take natural language and enter the appropriate information to the fields of the database. This is a useful tool when there are many thousands of fields that need instantiating and there are is a significantly large amount of natural language that needs processing. There is also a link between the automated input of information to ontologies.

There has been many research projects that have focused around the use of NLP technologies and system requirements engineering [98]. This has reached the extent where there are COTS requirements tools that contain NLP tools such as being able to help with the identification of duplicate requirements. However this still provides a research challenge where they are actively researching new ways of improving the identification equivalent requirements [99]. It is now at the stage where the semantics of the requirements is the subject of the similarity scoring rather than just looking at the occurrence of the same words.

5.1.8 Current Available Natural Language Processing Tool Libraries

Due to the rapid expansion of NLP research over the past 10 years there have been a number of tools that have been made available for others to use. There are many small libraries that exist as the output of specific projects; however there are two that dominate the NLP literature: the Stanford Core NLP; and the Python Natural Language Took Kit. These are the two libraries that have been researched in depth for this work.

Stanford Core Natural Language Processing

The majority of the information regarding the Stanford Core NLP tool set is from their detailed website [93]. However there are papers that have been published about its history and development [100]. The Stanford Core NLP is a suite of NLP

tools that has been developed to be accessible to multiple programing platforms, and as such has been implemented in Java. The toolkit was designed specifically with the intention of being easy to use, without the need to learn large amounts of proprietary commands and language [100]. It has later been translated for Python, Ruby, Perl, F# and .NET. It was first released to the public in 2010 since which the functionality has periodically increased.

The Stanford Core NLP currently has the capabilities of: tokenizing text, identification and removal of any XML tags if any are present in the text, determining the true likely case of text, part of speech tagging, gender identification of names, various tools for analysing tags, syntactic and semantic analysis tools.

This toolkit has been gaining popularity in both the academic and commercial sectors for NLP [100] over the last few years. However it does not have a large online community which shares information and problem solving advice. This may come in time if the popularity of the tool continues to develop.

Python Natural Language Toolkit

The Python Natural Language Toolkit (NLTK) has become somewhat of a 'go to' tool for first exposure to NLP. When downloaded for free it, comes with over 50 corpora and lexical resources such as wordnet, as well a suite of text processing tools such as taggers, tokenisation, and semantic reasoning. The toolkit is compatible with windows, Mac, and linux operating systems [101].

The first public release of the toolkit was in 2001 [102] and there have been regular updates since there is a thriving online community that supports the NLTK. This community has produced numerous tutorials and exercises for novices to work through. The online community also offer helpful support to each other when problems arise.

There is extensive documentation for each of the functions which can be modified by the user. The documentation extends to a book [103] which has been made available free online. The book has been written to not only explain how the NLTK can be used but also NLP and linguistic concepts. This book contains step by step examples of how to use the various functions with explanations as to what is happening.

Throughout the time that the NLTK has been available for public use there has been a steady stream of academic papers that have been published. These cover many different domains as researchers apply the tool kit to many different situations.

5.2 THE APPLICATION OF NATURAL LANGUAGE PROCESSING

The methods discussed have identified automated tools for NLP, the proposed methods would entail a computer to be able to process natural language. Getting a computer to process natural language is not as simple as it first seems. Despite computers being used to harness words by typing them into a document, this does not mean that the computer understands what is meant by the words or phrases composed. Despite word processors having spell and grammar checkers which give the illusion of understanding, it is actually programmed to look for key words and uses either statistical probability or logic to suggest what may be more correct options. Translation programmes are a realm of interesting research in which the goal is to get a computer program to understand what is the sentiment behind what is being said and use this as the basis for a translation. This understanding of sentiment is needed for effective translation as there is not always a one to one mapping of the meaning of a word into another language.

5.2.1 DESCRIPTIVE VS PRESCRIPTIVE LANGUAGES

Within languages there are two broad classifications that most languages fit into, descriptive and prescriptive. A descriptive language has the property of the definitions and grammatical syntax of words changing over time as well as between users. In a prescriptive language all of the definitions and grammatical syntax of the words are strictly prescribed and adhered to by those using it, improper use is defined to be erroneous. Both forms of language have their strengths and weaknesses. The strengths of a clearly defined prescriptive language are that all parties know exactly what is meant by what is being communicated with a low level of ambiguity. However a rigid structure makes describing new concepts difficult or even impossible as there is no way to capture them. The contrast of this is that with a descriptive language the definitions of words are not held the same and those who use it all have their own idea as to exactly what each word means. This flexibility allows for new concepts to be conveyed however it comes at the price of ambiguity between individuals. With a descriptive language two people can receive a sentence of the same words in the same order and have a completely different understanding as to what was meant by the words and hence the sentence a whole. The result of this is an ambiguity between the author of sentence and the person interpreting it. The English language is considered to be largely descriptive in nature. To demonstrate how much the language has changed texts from as little as 50 years ago have distinct differences in syntax and definitions of words, which are evident even to the lay reader.

Within the problem space that this project is looking at it is understood that there are multiple languages present however the decision has been made to focus on the use of English as the primary language used within the defined problem space.

Whether English is a descriptive or prescriptive language is contentious in the literature. This discussion is a reoccurring theme in a series of papers that discusses the use, teaching, and understanding of the English language within Europe [104]. It is interesting to note that the Oxford English Dictionary, which has become the *de facto* for definitions of English words, does not have a single definition of words but rather is a historic dictionary whereby the etymology of each word is given rather than a single static definition [105]. Such evidence that English as a language is evolving and is continuing to do so indicates that despite the wishes of those who want it to be prescriptive, in practice it is not. Which leads to an interesting situation, with descriptive languages if everything is changing then any algorithms that are used to analyse it will need to be able to cope with this.

5.2.2 Implications of Descriptive Language on Engineering Projects

English is interesting in that there are many different genres, disciplines, and regional uses that use the same words in very different ways. This can result in two people both considering that they themselves to be speaking English to not be able to communicate effectively as both use words and grammar that the other does not know or understand. This is not even taking into consideration idioms and cultural references. It has been recognised that even within engineering the different disciplines can have significant communication issues even when specifying requirements [106]. A leading cause of ambiguity in communication between engineering disciplines is their use of the same words with differing definitions. This can lead to the situation were two engineers believe that they are in agreement with one another when in fact they can have very different ideas as to what they are discussing in the first place. It can get far worse as at least in this instance both parties are involved in the project from a technical perspective and their core understanding of the physics of the problem is likely to be similar. Whereas when engineers talk with other disciplines or customers their viewpoints may be suitably different and so they are effectively talking different languages. This is the reason why requirements' engineering is such a specialist role and one that does not translate well between different market sectors.

5.2.3 Implications of Descriptive Language For Natural Language Processing

Having a descriptive language to process using a computer raises issues as to how to deal with the same words in the same order can mean many different things depending on context and the nature of the communication. The way that many projects and the applications that they produce handle this issue is to reduce the variation and focus on a particular area or sub-set of the language as a whole. Such an approach is also taken in the work conducted in section 5.2.5 where the focus is on the language uses in the automotive modelling and simulation domain.

Having a descriptive language often means that rules, if present and accepted by the majority of those who use the language, will not be adhered to by all users in every sentence. This makes any form of logical construct prone to missing some of the occurrences of what it is intended to detect. Hence using computers which can be considered as logic machines to analyse such ambiguity raises many issues.

The feature of a descriptive language that the definitions change over time also means that any logical construct must also either be able to change over time or be recognised that the effectiveness of the program will diminish the further from the intended date of the documents being analysed.

5.2.4 Rule and Sentiment Based Natural Language Processing

Within the field of NLP there are two categories that most methods can be sorted by, rule-based, and sentiment-based.

Rule-based NLP uses a logical construct to analyse the text looking for specific occurrences of words, phrases and the order in which they are used. The strengths of this approach are: speed of development, comparatively fast execution for simple algorithms, Intuitive in construction, and the results can be verified manually. However there is the caveat that if a set of rules is applied to descriptive language there will inevitably be exceptions to the rules and information will be missed. To increase the accuracy, logical methods are often tailored to a specific writing style or domain.

Sentiment-based NLP attempts to gain an understanding of what the text being analysed is attempting to convey. These methods use statistical analyses and or machine learning techniques. The strengths of this approach are that sentences with different words and structure which have the same meaning can be analysed, compared, and indicate a relation to the same objects and or actions. These methods however require large initial data sets which need to be manually constructed, constantly verified, and outputs corrections are needed to ensure continued accuracy. Simple tasks can be much more computationally heavy when compared to rule-based methods.

There has been research into using a combination of rule-based and sentiment-based analyses which have shown some success. This uses rule-based algorithms to feed the sentiment-based NLP or uses rule-based NLP to verify the sentiment-based results. It is recognised that sentiment-based NLP is the direction that the majority of research is going and it has the flexibility to be implemented within an organisation and for it to automatically grow with the said organisation keeping its analysis accurate over time.

For this research a Proof of Concept (POC) is needed to show that the methods proposed in section 3 can be automated or assisted by the use of NLP. For the decision as when to use rule or sentiment-based analyses, there are a number of

project constraints that have to be taken into account. These constraints are not strictly part of the problem space as was defined in section 1.3. These constraints include the following.

- There is a limited pool of available existing industrial textual model and simulation documentation which is suitable for NLP analysis.
- Access to development tools and environments which are available to use is limited. This is due to the nature of the research as well as the organisation being used as a case study.
- Time is limited for the development and testing of whether NLP can be used and is of benefit in this use case.
- The POC is only intended to identify if this technology is of potential use in this use case, and not an industrial tool.
- Due to the systems engineering aspect of this work, all outputs must be in a form where verification can be conducted; the technological POC is no different.

With the five constraints identified above the decision was made to use rule-based rather than using sentiment-based NLP for the development of a POC.

5.2.5 Natural Language Processing Proof of Concept

To ascertain if the NLP has the potential to aid in the methods proposed in this work a POC was constructed. The purpose of this POC is to investigate if the capabilities of this technology can in some way aid in the capture and analysis of the textual information that form parts of the proposed processes.

The tasks where NLP has been identified as being of potential benefit are as follows.

- Ascertaining if the documentation of a potential model indicates if there is a match between the capabilities of the model and a section for the simulation requirement.
- Compilation of the information needed for the completion of the proposed integration tables from the model documentation.
- The comparison of completed integration tables, to ascertain if the models are semantically similar and hence suitable to integrate.

To produce this POC the Stacked Systems Engendering method was implemented coupled with a modified spiral approach for iterative progression of the software. The ‘understanding’ of the problem phase of the Stacked Systems Engineering is covered in section 3.2. However time and effort still need to be invested into investigating languages and scripting tools.

5.2.6 Natural Language Processing Proof of Concept Requirements

The task of producing a NLP POC is an investigation into the capabilities of the technology in a new domain. As this is a new application the exact capabilities of the technology are unknown. For this reason the requirements were organised such that each was the staging for its successor. It was intended that the requirements would start with simple tasks and work up to being challenging. Hence it was never envisaged that all of the requirements would be fully satisfied.

Operational Requirement of the Application

To ascertain if NLP technologies can aid in analysing the suitability of integrating existing models and simulations from their documentation.

Functional Requirements

Functional requirements are considered to be the capabilities of the system being developed, and what they will be capable of when complete. What this means for the POC is that the functional requirements are the reference points as to if this NLP technology can provide the functionality that is needed to be of use in this problem space. Examples of functional requirements include the following.

- The application is to read in text files.
- The application is to identify the word types (Noun, Verb, Connective, etc.) of all words in the document being analysed.
- The application is to identify sentences with Nouns present in text.
- The application is to identify sentences with Verbs present in text.
- The application is to identify any Noun-verb phrases present in text.
- The application is to Identified Nouns, Verbs, and Noun-verb phrases.
- Noun-verb phrases are to be saved to a file that can be read back into the application.
- The application is to have user-defined repository of key words and phrases.
- The application is to identify key user-defined words or phrases.
- The application is to identify the sentences that the identified keywords are part of.
- The application is to be capable of counting the frequency that types of words are present in the text.
- The application is to compare if two or more documents have the same words in and report the frequency across the documents.

- The application is to compare two or more documents to ascertain if they contain the same keywords and the number of times that they occur in each document.
- The application is to identify sentences with Nouns with predeterminers.
- The application is to compare two or more documents to ascertain if they contain Noun, predeterminer phrases.

Non-Functional Constraints

The requirements specified below are concerned with matters that are not the functionality of the application but rather state the environmental constraints that bound the potential solution space. In this case the constraints are predominantly concerned with what it needs to be developed in and executed on. Many of these constraints are due to the project that this work is a part of and the resources available. Examples of non-functional constraints include the following.

- The application is to read in '.txt' files where the text has no formatting other than space, tab, and carriage returns. [No bold underlined italicised etc].
- The application must operate on a mid-range engineering laptop with 8GB RAM, 250 GB memory, 500 GB hard drive, and a 2.9GHz quad core processor.
- The application should not take any longer than one hour to analyse two documents for all of the required algorithms.
- The development environment used must not require any more computational power than that available for execution of the developed application.
- The development environment and application is to operate on a windows 7, 64-bit operating system.
- All uses of any software are to be fully legal for academic research use.

Languages and Development Environments

The POC has to be created in a development environment using a computer language. An investigation was conducted as to how current and past NLP applications were developed and how the researchers reported their experience using different combinations. Consideration was given to the fact that allowances would have to be made to the differences in use case.

The decision as to which language and development environment to use was made by first assessing the strengths and weaknesses of the languages. Once a language was chosen a decision as to which development environment to be used was made.

The languages that were considered to be used for the development of the POC where; C, C#, Math script, G, and Python. These languages were chosen for consideration because of their use within the literature, the basic properties that they have, and that there is a range of different types of language which can be considered.

C is a procedural language that has been the foundation of many of the computational developments since 1972 [107]. The result of this heritage is well established coding practices with many sources of free help and advice. There are only a limited number of libraries that are freely available for NLP using C. This means that almost all intended functionality of the POC would require coding from basic string manipulation functions. This would be a time consuming endeavour.

C# is a development of C but whereas C is procedural C# is object-oriented. This makes parallel processing possible and requires different programming patterns to be used [108]. There has been some NLP work produced using C#, however the libraries that are freely available are somewhat limited.

Math script is a form of language that is used by development environments such as MATLAB [109]. Math script is not a language as such but rather a name for a collection of similar inputs to various mathematical tools. This language has common features and syntax between tools. However functions can be tool and version dependent. Often basic function calls are the same across environments with changes required for tool specific functionality. These languages can be optimised for mathematical operations rather than string based manipulations. There are some string functions or ways to manipulate strings with some effort being put into conducting NLP using math script however due to the tools that interpret it not being optimised for text execution can be slow and algorithms cumbersome to implement.

G is the graphical language that National Instruments use in their products [110]. It is a graphical based language that uses the concept of data flow. This language is quick to pick up and many find intuitive to use. It has string operators and some NLP functions however this is far from its intended use as a data acquisition tool.

Python is an open source language that dynamically defines its variables on the fly. This language has been found by far to be the most commonly used language for NLP applications. There are numerous open source toolkits available which have high quality help files available for guidance on design patterns for textual analysis. Due to the means by which Python has been developed there are some libraries that will only work with some versions of Python. The two main variants have been developed side by side, Python version 2 and Python version 3. The differences between them are such that some functions will behave differently in the two versions. As the leading tool kit for NLP

in Python uses version 3 from this point in this work when referring to Python it is considered to refer to Python 3. More information on the differences between Python 2 and 3 can be found at the Python official web page [111]

To compare the possible programming languages a Pugh matrix [112] was used, this can be seen below in Figure 5.3 below. The categories that have been used to evaluate the different languages are: number and quality of existing NLP functions and libraries, quality of native string (or text) manipulation functions, documentation for language available online, availability of online help, memory allocation usage and control, user familiarity, availability of free online training, quality of documentation available for existing NLP functions and libraries, and quality of documentation of base functions.

All categories are scored between 0 and 5; this means that there is no middle value and there is a minimum possible total score of 0 and a maximum of 45.

CATEGORIES	C	C#	MATH SCRIPT	G	PYTHON
Number and quality of existing NLP functions and libraries	1	2	1	2	5
Quality of native string (or text) manipulation functions	2	2	1	1	3
Documentation for language available online	4	4	3	3	4
Availability of online help	4	4	3	4	4
Memory allocation usage and control	5	5	2	3	3
User familiarity	2	1	3	5	2
Availability of free online training	4	4	2	2	4
Quality of documentation available for existing NLP functions and libraries	2	2	1	1	5
Quality of documentation of base functions	4	4	2	2	4
TOTAL	**28**	**28**	**18**	**23**	**34**

Figure 5.3 Pugh matrix for the selection of the language to conduct the NLP POC. Each score is between 0 and 5 representing least and greatest desirability respectively.

Python received the highest score from the Pugh matrix analysis of the identified languages that could be used for the POC. This decision was sanity checked and it was decided that Python was to be the language used.

With the language having been selected there are a number of different development environments that could have been used to script the POC application.

An investigation was undertaken as to the different development environments that could be used with the Python3 language, the environments that were found were, Notepad, Visual studio, IPython and Spyder. These are all readily

available Python scripting tools and are all capable of producing scripts that a compiler can read and use.

A Pugh matrix was used to aid the selection of a possible development environment to use. The categories that each environment was scored against were; availability, functionality, quality of documentation, IT support (from the organisation that this research is being conducted within), computational overheads, available guidance and tutorials, user assisting functions, and debugging tools.

CATEGORIES	NOTEPAD	VISUAL STUDIO	IPYTHON	SPYDER
Availability	5	5	5	5
Functionality	2	4	3	5
Quality of documentation	3	4	3	3
IT support	5	5	2	2
Computational overheads	4	2	2	1
Available guidance and tutorials	2	4	4	2
User assisting functions	1	5	3	5
Debugging tools	1	4	3	4
User familiarity	4	4	2	1
TOTAL	**27**	**37**	**27**	**28**

Figure 5.4 A Pugh matrix to aid in the selection of scripting tools for POC development. Each score is between 0 and 5 representing least and greatest desirability respectively.

Using the outcome of the analysis of the Pugh matrix, Microsoft Visual Studio was chosen for the development environment to produce the NLP POC. Other environments were used for troubleshooting issues with Microsoft Visual Studio. When installing Python, packages can be installed that increase functionality of the development environment as well as giving access to function libraries. The packages that were installed were bundled in the Anaconda installer [113]. This contains over 150 pre-built and tested Python packages [114].

By using such an installer it can be guaranteed that all of the packages that are needed are installed in the correct order and all of the paths are set to the right addresses. Having these packages installed gives the option to use functions that have been developed such as graphs, charts, as well as a whole host of NLP capabilities. Having these capabilities to hand vastly reduces the development time of programming, which will be of benefit for the POC.

One of the key packages that was used was the Natural Language Toolkit (NLTK). This is a series of Python functions and corpora that were created in 2001 by Steven Bird and Edward Loper from the University of Pennsylvania. Since 2001 it has been developed by many and is now one of the leading toolkits for NLP using Python [101].

To use some of the functionality of the NLTK package additional installs of corpora, taggers, and dictionaries are required [115]. A Python script is installed with the NLTK when run it will produce a GUI where the additional information can be selected and downloaded from an online repository. It is advised that if this work is being replicated that all of the environment and additional information is downloaded before attempting to run any of the code that has been developed.

5.2.7 Process For The Development of Proof of Concept Code

With the understanding of the problem and the requirements having been captured, the next two stages of the Stacked Systems Engineering method are architecture and design. An approach which merges these two components was decided upon. The architecture would predominantly be defined as a central calling block that calls functions and passes data. This allows for functions to be written in isolation and brought into the main program. A diagram of the architecture as it was intended can be seen in Figure 5.5 below.

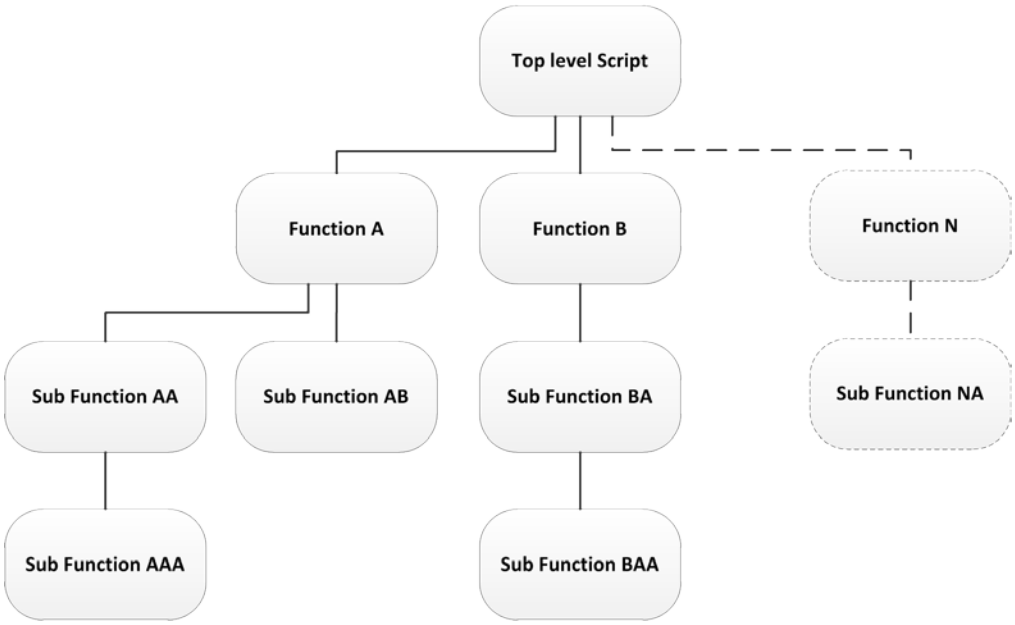

Figure 5.5 The planned architecture for the NLP POC. A top level script orchestrates the calling of functions which are formed from other component functions. The solid lines denote a data exchange. The dashed lines denote the expandable nature where additional functions can be added.

With the incremental increase in functionality that was intended for the POC, and with base functions building upon one another to give increasingly complicated behaviour, a spiral process was used. This spiral process was developed using the work in [77] as inspiration. The developed spiral process can be seen in the diagram below. The diagram uses the idea of an iterative process winding around the same actions however the nature of the tasks has changed significantly from the process proposed in the literature [77].

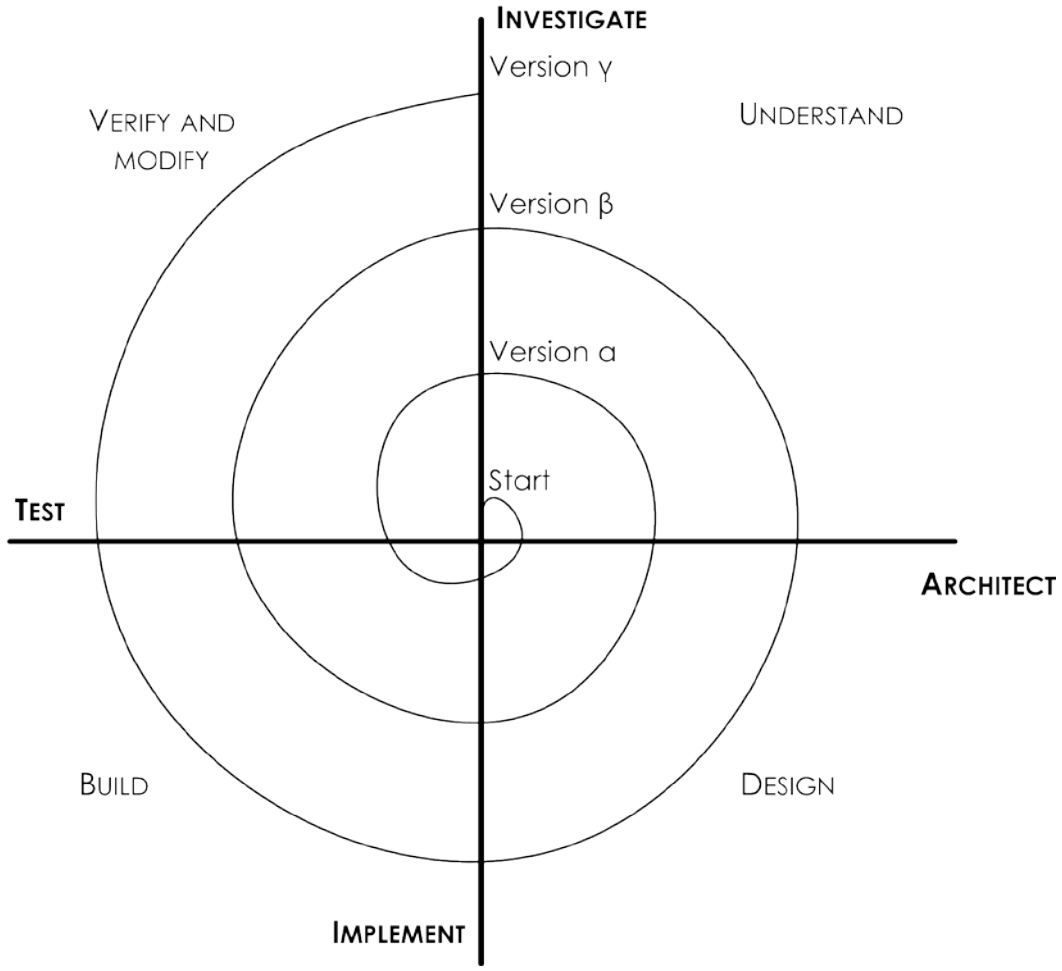

Figure 5.6 This is a new take on the traditional spiral. It has an Investigate, Architect, Implement, and test, as the constituent parts of the process. At the end of each circumnavigation a new behaviour is available to use.

The constituent parts of the spiral process shown in Figure 5.6 above require explanation as to what actions and decisions are carried out.

Spiral Process: Investigate

To investigate current technologies and tools to ascertain what is currently possible with given constraints. This understanding is used with the desired capabilities of the overall application to assess what is possible. Consideration is given to what has gone before (capabilities wise) and how a new capability could be built upon. At the end of this stage an idea as to what the function is going to be capable of and the way in which it is to be implemented is known. By the end of this stage tests are specified as to ascertain if the intended function behaved as desired.

Spiral Process: Architect

With a new capability and technology selected, consideration is given to the how the function will be put together. This includes how it will use core functions and previously developed functionality. The interfaces are also set at this stage of the process. By setting the interfaces at this point it allows for considerations as to how the application will handle the data that it will receive, produce, and the

means by which it will be represented. By the end of this stage a full plan of what is going to be produced and the form that it will take will be in place. By the end of this stage, once the architecture has been set, a full design is to be created.

Spiral Process: Implement

With the architecture and design of the intended functionality known it can now be scripted and complied. This is the stage where the code for the application is written and built. The implement stage ends when the function is a point where it can be integrated into the existing application ready for testing.

Spiral Process: Test

This stage is essentially the verification of the new functionality that has been produced in this iteration as well as how this new code interacts with any existing code. The verification point comes from the decisions and specification which is set in the investigation stage of the spiral. Modifications can be made to the code at this stage to bring the behaviour into line with what is desired, expected, and specified. Modifications are only made to the implementation not the architecture at this stage. The results of testing can be used in the investigation stage of the next iteration. At the end of the test phase a new version of the code will have been developed.

Evaluation of the New Spiral Process

The new spiral model has been found to be useful when coding software when the capabilities of the technology are not known before commencing the task. This process requires functions to be used whereby the functionality can be added in a modular or consecutive way. Limitations of this process are decisions which are made early on with a function can require rework to make it fit with desired behaviour later. Applying such a flexible approach to the task of investigating the potential of a technology has demonstrated the possibility for such a method to demonstrate promise for a technology without the full capabilities of said technology being known. However it is recognised that such a flexible approach would not be the most suitable for many other situations, and cannot be used as an excuse for haphazard system development. It is far from optimal for developing a complex program using established technology in the field. As the lack of structure of the whole does not guarantee that the final solution will achieve the overall intended intent to solve the given problem. The application would also most likely not be the most optimal possible use of the technology. Whereas in this application there is no single intended outcome for the final solution to have other than an investigation of the technology. If the knowledge that has been gained from this POC was known at the start of the software task the decisions on implementing the same capabilities would be different.

5.2.8 Capabilities of The Proof of Concept

From the start this POC was only intended to demonstrate the possibility of using this NLP technology to assist in the collection and comparison of information necessary for meaningful model and simulation integration. Table 5.1 is a simple Red, Amber, and Green method of assessing if the requirement has been met or not has been used. Red not met at all, Amber some work has been done towards this and it shows promise, and Green fully capable.

Requirement	Pass	Promise	Not at all
The application is to read in text files.	✓		
The application is to identify the word types (Noun, Verb, Connective, etc.) of all words in the document being analysed.	✓		
The application is to identify sentences with Nouns present in text.	✓		
The application is to identify sentences with Verbs present in text.	✓		
The application is to identify any Noun-verb phrases present in text.	✓		
Identified Nouns, Verbs and Noun-verb phrases are to be saved to a file that can be read back into the application.	✓		
The application is to have user defined repository of key words.	✓		
The application is to identify key user defined words or phrases.	✓		
The application is to identify the sentences that the identified keywords are part of.	✓		
The application is to be capable of counting the frequency that types of words are present in the text.	✓		
The application is to compare if two or more documents have the same words in and report the number of times that it does.	✓		
The application is to compare two or more documents to ascertain if they contain the same keywords and the number of times that they occur in each document.	✓		
The application is to identify sentences with Nouns with pre determiners.		✓	
The application is to compare two or more documents to ascertain if they contain the say Noun, pre determiner phrases.		✓	

Table 5.1 Functional requirements RAG test. This figure shows for each of the requirements if it has been fulfilled or not. Green pass, Amber shows promise, and Red not capable at all with no sight of being able to do so.

From this analysis it shows that the majority of the functionality that was initially required has been achieved with this NLP technology. Work towards the last two requirements has been started and the progress was good. With additional time it is believed that this capability could be achieved. To ensure that the constraints that were placed upon the POC have been adhered to a RAG

assessment of the of the functional constraint requirements was conducted the results of which can be seen below in Table 5.2.

REQUIREMENT	PASS	PROMISE	NOT AT ALL
The application is to read in txt files where the text has no formatting other than space, tab, and carriage returns. No bold underlined italicized etc.	✓		
The application must operate on a mid-range engineering laptop with 8GB RAM, 250 GB memory and a 2.9GHz quad core processor.	✓		
The application should not take any longer than one hour to analyse two documents for all of the required algorithms.	✓		
The development environment used must not require any more computational power than that available for execution of the developed application.	✓		
The development environment and application is to operate on a windows 7, 64-bit operating system.	✓		
All uses of any software are to be fully legal for academic research use.	✓		

Table 5.2 RAG analysis of the Functional Constraints of the POC.

The RAG analysis shown in Table 5.2 Figure shows that the POC operates within all of the constraints that were set during the requirements stage.

5.2.9 ALGORITHMS IMPLEMENTED IN THE PROOF OF CONCEPT

To achieve the desired functionality there were a number of logic based algorithms that were developed, implemented and tested. To describe the behaviour of the NLP application, set theory is used in conjunction with formal logic. Within the formal logic literature there are discrepancies as to what the operators represent. For this reason Table 5.3 below details the operators used and a description of what they represent given.

LOGIC SYMBOL	DESCRIPTION
{ }	Set. Items within a set are between the two parentheses
X_y	Set. Capital denotes a type of set with the subscript being a label to denote the origin of the elements within the set.
∈	Is an element of. q ∈ X means q is an element of the set X.
=	Equals. Contains the same values.
|	Containing. {X | P(X)}, is used to denote the set containing all objects for which the condition P has.
∩	Intersect. X ∩ Y means the set that contains all those elements that Y and X have in common
⊆	Subset. {X ∈ Q : P(X)} denotes the set of all x that are already members of Q such that the condition P holds for X.
∧	Logical And. The statement A∧B is true if both A and B are true; if both (or one) is false, the statement is false.
∨	Logical OR. The statement A ∨ B is true if A or B (or both) are true; if both are false, the statement is false.

Table 5.3 : Definition of the logical symbols used in mathematical descriptions.

Each term is defined before use and all will be in the type face as shown below.

$$D = \{The\ entire\ content\ of\ a\ textual\ document\} \qquad \textit{Equation 5-1}$$

$$W = Words \qquad \textit{Equation 5-2}$$

$$P = Punctuation \qquad \textit{Equation 5-3}$$

All documents are formed of words and punctuation. Words include natural words as well as numbers. Punctuation is considered to include everything else that is not captured by the words term in a document. The text documents are only considered to be comprised of ASCII text.

$$D = \{W, P\} \qquad \textit{Equation 5-4}$$

For the description of the following algorithms consider that two documents are being analysed to ascertain if there is a possibility that the models or simulations that they represent could be successfully integrated. To distinguish the two sets the subscript A and B superscript are used throughout the logical representations.

Therefore

$$D^A = \{W^A, P^A\}$$ *Equation 5-5*

And

$$D^B = \{W^B, P^A\}$$ *Equation 5-6*

There are means by which a document can be broken down into lines. A line is a complete sentence or part of list that has been separated by the author by a carriage return. When D_A and D_B are converted to lines each line is a set and all of the line sets are part of a larger document lines set.

$$L = Lines$$ *Equation 5-7*

$$L = D = \{W, P\}$$ *Equation 5-8*

Considering the two documents

$$L^A = D^A = \{W^A, P^A\}$$ *Equation 5-9*

$$L^B = D^B = \{W^B, P^B\}$$ *Equation 5-10*

Methods exist for assessing and categorising each word or punctuation point in a document. This method has been used with each element of the lines sets being categorised and assigned a relevant tag.

$$T_L = \{T | T = T_{WL} \vee T_{PL}\}$$ *Equation 5-11*

The subscript for the two documents is added to the T_L, T_L^A, T_L^B, Respectively.

With the tags identified for all of the elements in the documents, the tags can be further decomposed. The tagged words contain the tags which are displayed in Table 5.4 below. The content of this table has been taken from the NLTK help function.

Type	Tag	Example words
noun, common, singular or mass	'NN'	common-carrier, cabbage, shed, thermostat
noun, proper, singular	'NNP'	Ranzer, Conchita, Christos, Cougar, Yvette, Ervin, Liverpool
noun, proper, plural	'NNPS'	Americans, Amusements, Animals, Antiques, Apache
noun, common, plural	'NNS'	Undergraduates, products, bodyguards, coasts
verb, base form	'VB'	Ask, assemble, bake, balkanize, bank, benefit, boil, build
verb, past tense	'VBD'	Dipped, pleaded, swiped, tidied, registered, cushioned, aimed
verb, present participle or gerund	'VBG'	Stirring, focusing, judging, stalling, lactating, veering
verb, past participle	'VBN'	Dilapidated, aerosolized, chaired languished, experimented
verb, present tense, not 3rd person singular	'VBP'	Predominate, wrap, resort, twist, spill, cure, lengthen, brush
verb, present tense, 3rd person singular	'VBZ'	Bases, reconstructs, marks, mixes, displeases, seals, carps
conjunction, coordinating	'CC'	And, both, but, either, for, less, minus, neither, nor, or, plus, so
numeral, cardinal	'CD'	Seven, 1987, twenty, 78-degrees, IX ,'60s, 271, dozen
Determiner	'DT'	All, an, another, any, both, each either, half, many, neither, no
existential there	'EX'	There
foreign word	'FW'	Gemeinschaft, hund, ich, jeux, habeas, Haementeria, Herr
preposition or conjunction, subordinating	'IN'	Astride, among, upon, whether, out, inside, pro, despite, on, by
adjective or numeral, ordinal	'JJ'	Third, ill-mannered, pre-war, regrettable, oiled, calamitous
adjective, comparative	'JJR'	Bleaker, braver, breezier, briefer, calmer, cheaper, choosier
adjective, superlative	'JJS'	Cheapest, classiest, cleanest, darkest, deadliest, dearest
list item marker	'LS'	A A. B B. C C., G H I J K, SP-44001 SP-44002 SP-44005
modal auxiliary	'MD'	Can, cannot, could, couldn't, dare, may, might, ought, shall
pre-determiner	'PDT'	All, both, half, many, quite, such, sure, this
genitive marker	'POS'	', 's

pronoun, personal	'PRP'	Hers, herself, him, himself, it, itself, me, myself, one, oneself
pronoun, possessive	'PRP$'	Her, his, mine, my, our, ours, their, thy, your
Adverb	'RB'	Occasionally, unabatingly, maddeningly, adventurously
adverb, comparative	'RBR'	Further, gloomier, grander, harder, harsher, leaner, lengthier
adverb, superlative	'RBS'	Best, biggest, bluntest, earliest, farthest, hardest, largest, least
Particle	'RP'	Aboard, about, across, along, at, Crop, down, ever, fast, for, forth
'to' as preposition or infinitive marker	'TO'	To
Interjection	'UH'	Goodbye, Gosh, Wow, Jeepers, Oops, huh, dammit, shucks
WH-determiner	'WDT'	That, what, whatever, which, whichever
WH-pronoun	'WP'	That, what, whatever, whatsoever, which, who, whom
WH-pronoun, possessive	'WP$'	Whose
Wh-adverb	'WRB'	How, however, whence, whenever, where, whereby
sentence terminator	'.'	. , ! , ?
colon or ellipsis	':'	: , ; , ...
opening quotation mark	'``'	` , ``

Table 5.4 This table shows all the possible tags that can be assigned by the NLTK POS tagger. The content of this table was compiled from the NLTK help files. [116]

Identification of, Noun, Verb, Noun-verb Phrases and Comparison between Documents

Within systems engineering there is the concept of noun-verb phrases. The idea is that an object is defined by the noun and the action that it does or has done to it is captured by a verb. Using the tagged text sentence noun-verb sentences can be identified.

The words that are identified as nouns are captured by the following tags; noun, common, singular or mass; noun, proper, singular; noun, proper, plural; noun, common, plural.

$$Identified\ nouns = \ NT = \{'NN','NNP','NNPS','NNS'\} \qquad \textit{Equation 5-12}$$

The nouns are a sub-set of the words in a document.

$$Nouns\ in\ a\ document \ = NT \subseteq W \qquad \textit{Equation 5-13}$$

The nouns in the two documents are denoted by.

$$N^A = NT \subseteq W^A$$ *Equation 5-14*

$$N^B = NT \subseteq W^B$$ *Equation 5-15*

Nouns that are common in both document A and Documents B can be defined as

$$C_{NANB} = N^A \cap N^B$$ *Equation 5-16*

All of the verbs present in the text that are of interest to analyses of capabilities of models and simulations are captured in the tags: verb, base form; verb, past tense; verb, present participle or gerund; verb, past participle; verb, present tense, not 3rd person singular; verb, present tense, 3rd person singular.

$$Identified\ Verbs = VT$$
$$VT = \{'VB','VBD','VBG','VBN','VBP','VBZ'\}$$ *Equation 5-17*

The Verbs are a subset of the whole text such that

$$Verbs\ in\ Document = VT \subseteq W$$ *Equation 5-18*

This is applicable to both documents hence

$$V^A = VT \subseteq W^A$$ *Equation 5-19*

$$V^B = VT \subseteq W^B$$ *Equation 5-20*

The verbs that appear in both document A and document B

$$C_{VAVB} = V^A \cap V^B$$ *Equation 5-21*

The first algorithm that was implemented was the implementation of equations Equation 5-19, Equation 5-20, and Equation 5-21. This is the simplest comparison that can be made between two documents. By extracting the nouns, the objects involved are located by examining the verbs, limited indication as to the actions involved can be gleaned. This first algorithm proved that the nouns and verbs can be identified by the POC. The application can be made to be of more use if noun-verb phrases are identified and compared.

With Noun-verb phrases if the same nouns and verbs are in sentences in two documents there is the strong possibility that they are referring to the same or similar things. Hence an algorithm for finding such similar sentences was developed.

A line that has at least one noun in is denoted by

$$Line\ that\ contains\ at\ least\ one\ noun = L_N \quad \text{Equation 5-22}$$

A line that has at least one verb is denoted by

$$Line\ that\ contains\ at\ least\ one\ verb = L_V \quad \text{Equation 5-23}$$

To implement the noun-verb search, lines that have at least one noun and one verb need to be captured. Hence lines that have at least one noun and at least one verb are subsets of lines. A line that has both a noun and a verb in will be a subset of the lines that have at least one noun and are also a member of lines that have at least one verb.

$$Line\ with\ at\ least\ one\ noun\ and\ one\ verb = L_{NV} \quad \text{Equation 5-24}$$

$$L_{NV} = L_{NV} \in L = L_{NV} \in L_N \wedge L_V \quad \text{Equation 5-25}$$

For the purpose of implementation the consideration as to if there are multiple noun-verb phrases present has not been addressed as long as there is at least one noun and one verb, it is selected.

For the two documents the subscripts are added see below.

$$Document\ A\ Line\ with\ at\ least\ one\ noun\ and\ one\ verb = L_{NV}^{A} \quad \text{Equation 5-26}$$

$$L_{NV}^{A} = L_{NV}^{A} \in L^{A} = L_{NV}^{A} \in L_N^A \wedge L_V^A \quad \text{Equation 5-27}$$

$$Document\ B\ Line\ with\ at\ least\ one\ noun\ and\ one\ verb = L_{NV}^{B} \quad \text{Equation 5-28}$$

$$L_{NV}^{B} = L_{NV}^{B} \in L^{B} = L_{NV}^{B} \in L_N^A \wedge L_V^A \quad \text{Equation 5-29}$$

The sentences in both documents that have the same nouns and verbs in are expressed by the following equations.

$$NVC = L_{NV}^{A} \cap L_{NV}^{B} \quad \text{Equation 5-30}$$

The equation above may not always have elements in. NVC is allowed to be empty.

Direct and percentage comparisons between documents

When comparing two documents using tags the direct numerical comparison of the number of times a particular tag is present may not be the most

representative metric of the similarity. If two documents are of differing lengths then a percentage comparison of the number of times a tag appears in the document is not affected by the difference in document length.

The equation used for such analysis is shown below.

$$percentage\ of\ tag = \frac{Number\ of\ times\ a\ tag\ has\ been\ assigned\ in\ the\ document}{Total\ number\ of\ all\ tags\ assigned\ in\ the\ document} \times 100 \qquad \textit{Equation 5-31}$$

This percentage calculation was used to produce a tag percentage set for the two documents. The tag percentages were then displayed comparatively on a bar chart.

The percentage of times a tag is used for different words can also be used as a metric to compare documents as it is a means of assessing how many different words the tag is being used for.

$$Percentage\ of\ tagged\ word = \frac{Number\ of\ times\ tag\ has\ been\ assigned}{Number\ of\ diffrent\ words} \times 100 \qquad \textit{Equation 5-32}$$

The percentage of tagged words gives an interesting metric and it has substantially different values for each tag name that that of the percentage of tag.

Finding Company words and Phrases

To locate the words and phrases that have been identified as being important, sets of data were used so that the words within the document could be searched. The words and phrases that are of interest were sorted into sets of lists.

Category of words	Examples of words in category	Symbol used in description
Development Environments	'LabVIEW', 'Matlab', 'SIMPACK'	De
Modelling Terms	'input', 'output' ,'interface'.	Mt
Project Terms	'Kick Off', 'validated', 'verified'	Pt
Programing Languages	'C', 'C#', 'Java'	Pl
File Types	'mdl', 'txt', 'jpg'	Ft

Table 5.5 Categories of words with example words and their respective symbols used in logical descriptions.

Searches are conducted to find if any of the following sets have elements

$$W_{De} = W \cap De \qquad \textit{Equation 5-33}$$

$$W_{Mt} = W \cap Mt \qquad \textit{Equation 5-34}$$

$$W_{Pt} = W \cap Pt \qquad \textit{Equation 5-35}$$

$$W_{Pl} = W \cap Pl$$ *Equation 5-36*

$$W_{Ft} = W \cap Ft$$ *Equation 5-37*

A file is then produced and stored which contains all identified company words and phrases.

$$Documents\ Company\ Words\ and\ Phrases\ = Idwp$$
$$Idwp = \{W_{De}, W_{Mt}, W_{Pt}, W_{Pl}, W_{Ft}\}$$ *Equation 5-38*

To use the company words and phrases as a means to compare two sets of documents (or integration tables) a comparison needs to be made and the resultant compare list found. The set description of this can be seen below.

$$The\ Idwp\ for\ document\ A = Idwp^A$$ *Equation 5-39*

$$The\ Idwp\ for\ document\ B = Idwp^B$$ *Equation 5-40*

The final list of company specific words and phrases that are in both documents is shown below.

$$Company\ words\ and\ phrases\ common\ in\ both\ documents = Idwp_{AB}$$ *Equation 5-41*

$$Idwp^{AB} = Idwp^A \cap Idwp^B$$ *Equation 5-42*

This is calculated by implementing

$$Idwp^{AB} = \{W_{De}^A \cap W_{De}^B, W_{Mt}^A \cap W_{Mt}^B, W_{Pt}^A \cap W_{Pt}^B, W_{Pl}^A \cap W_{Pl}^B, W_{Ft}^A \cap W_{Ft}^B\}$$ *Equation 5-43*

These algorithms were implemented for a number of functions. To achieve some of the steps other supporting functions had to be produced.

5.2.10 STRUCTURE OF THE PROOF OF CONCEPT

The structure of the final iteration of the POC was as initially intended by the architecture shown in Figure 5.5. This architecture was found to work well with the successive iterations of increased factuality, however it is to be noted that this structure would not scale well for a large project. If this POC was to be taken further a redesign would be recommended, however it served its purpose as a test base. The final structure of the program can be seen in the Figure 5.6 below.

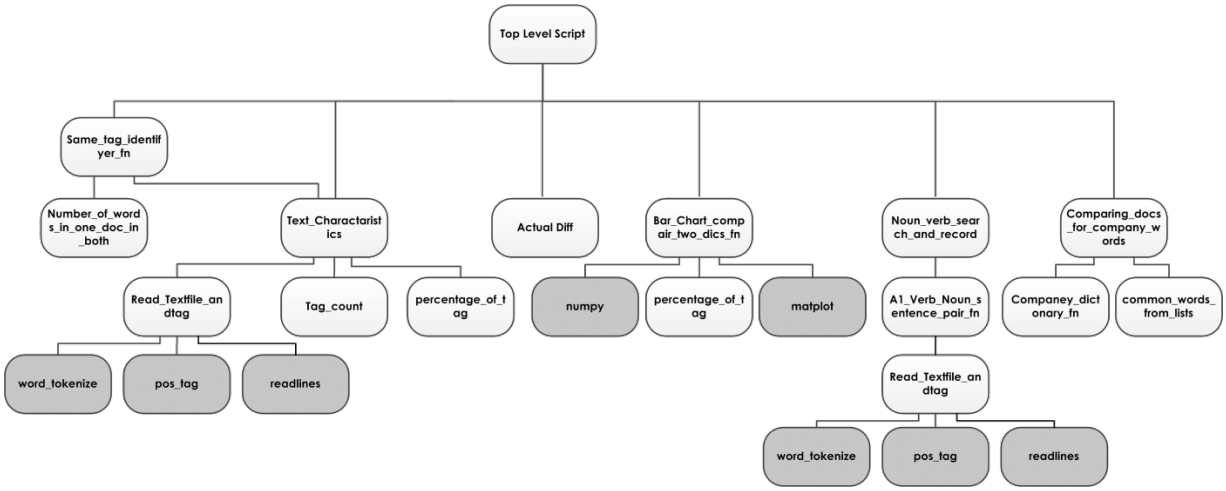

Figure 5.7 Final architecture of the POC. White boxes denote functions that were created for this project. Boxes shaded in grey are functions from existing libraries.

5.2.11 FUNCTIONALITY OF PROOF OF CONCEPT FUNCTIONS

The functions shown in Figure 5.6 implement the algorithms presented earlier in section 5.2.9 each function will be described and the functionality discussed.

Top Level Script

This is the section of code that calls the functions in a defined order. Each successive function can not only call on the raw text files but also the values from previous functions. This architecture has allowed for each additional function to tack on to its predecessors. This means however that removing any single function may cause other to cease to work as intended.

Same_tag_identifyer_fn

This function identifies words that are identical in two documents and gives the frequency of use of that word from both documents.

This function conducts the following steps:

1. Reads in two text files
2. Calls the Text_characaristics_fn which gives tagged files
3. The two sets of tagged words are searched to see if there are any words that are the same in both documents
4. If any words are found to be present in both documents the frequency of their occurrence is found in both documents using Number_of_words_in_one_doc_in_both.

This function allows for a very simple comparison of the text found in two documents.

Number_of_words_in_one_doc_in_both

To enable the number of times that a word appears in two documents this function was created. This function outputs the words that appear in both documents and the frequency that it occurs in both documents.

Text_charactaristics
There are characteristics that all textual documents have. This function returns a list of all of the words tagged, the number of times each tag has been assigned, the number of lines the document has, the number of words that the document has, the total number of tags that have been assigned, the percentage of the total that each tag has been assigned and the percentage of the whole that each word has been used. To accomplish this Tag_count_Read_textfile_andtag, percentage_of_tag, and standard Python functions were used.

Read_Textfile_andtag
Within NLP technologies there is the technique of tagging words (more on this can be found in section 5.2.9). This function uses the NLTK pos tag to tag all of the content of a text file. This function returns a list of words and their assigned tags, as well as a list of the content of the text file.

Tag_count
The frequency of the different possible tags is calculated by this function. The output of this function is a list of the words and tags, a list of each tag and how many times it is present in the list of words and tags.

percentage_of_tag
From a given list of tagged words this function calculates from the tags present the percentage that a designated tag appears in the list as well as if a word type tag is called the percentage of word tags. This function requires; the number of times a tag is present in a list, the total number of tags in the list, the key of the tag in question and the number of words in the text.

Actual_mod_Dif
A simple function that calculates the actual (modula) difference between two numbers, the numbers can be any numerical data type. The output is a floating point number.

Bar_chart_compair_two_dics
This function produces a bar chart of the percentage distribution of each possible tag present in two documents, on the same axis. This function uses numpy a Python tool for more advanced numerical analyses, part of which is graphing tools. The output of this function is a bar chart on a GUI that can be manipulated for closer visual inspection. For the program to continue this GUI has to be closed.

Noun_verb_search_and_record
This function calls two text files and identifies any sentences that have the same noun-verb pair in. The function used A1_Verb_Noun_sentence_pair_fn to locate the noun-verb sentences and compares the identified sentences to ascertain if they have or have not got the same noun-verb sentence pairs in. The function returns the pair of sentences that have the same noun and verb pair in, as a numerical value as to the number of identified noun-verb sentence pairs. Any

identified noun-verb sentence pairs are saved in a text file that can be interrogated at a later date.

A1_Verb_Noun_sentence_pair_fn
A function that extracts any noun-verb sentences from a tagged document. The outputs of this function are pairs of identified sentence which have the same noun and verb in as well as a list of just the pair of nouns and verbs. The only input required is the file path of the text file to be analysed. This is so that this function can operate without running the text_charactaristics function.

Comparing_docs_for_company_words
This function orchestrates the searching for specific words from two documents. The function searches for words that are captured in Company_dictionary which include development environments, common modelling terms, common project terms, common project language, and common file types. The function then compares two documents to ascertain if they contain any of the same identified company words.

Companey_dict
This function holds the key words that are relevant to the environment for which the documents are emanating from. The categories that the words are stored in are; development environments, common modelling terms, common project terms, common project language, and common file types. From a given file path this function tags the file using read_textfile_and_tag and searches for the company words. The outputs of this function are lists of the identified words for the text file.

common_words_from_lists
The comparison of the two lists is conducted in this function. The two lists of identified words (which were present in the documents under test) are compared. The output is a list of any words that appear in both lists.

5.2.12 WHERE NATURAL LANGUAGE PROCESSING FITS INTO THE PROPOSED PROCESSES

Within the proposed methods there are a number of the process elements that have been identified as possibly being automated through use of NLP technologies. These process elements were highlighted in section 3 Figure 3.9. Although all of the identified processes elements involve natural language there are subtle differences between them.

Complete integration tables
This task involves the identification of types of words and their capture. There are specific pieces of information that have been identified as being critical for meaningful model and simulation integration all of which are captured in the integration tables(see section3.7.3). The information that is required for the tables can often be found in the model documentation. The perceived application of NLP technology in this case is for an application to pass over the model

documentation and extract the relevant information. At present the manual completion of integration tables takes a considerable time.

Compare integration tables

Once the integration tables have been completed for the component simulation elements, comparisons are made between them to ascertain if the models are both semantically and syntactically compatible. This task involves inspection of the relevant integration tables and ascertaining if integration is meaningful and even possible. If there is more than a handful of integration tables this task can take a considerable amount of time and effort. The content of the integration tables is almost wholly natural language.

There is the potential to have many possible models that are stored in a repository which all have their integration information captures in integration tables. If such a repository existed then being able to automate the process to ascertain which stored models are capable of being integrated has clear commercial benefit. Such a capability would allow for non-domain experts to select models either compatible with the models from their domain, or for those investigating the system form a high level of abstraction, to quickly assemble candidate simulation element for all areas of a potential design.

Does the model documentation match a section of simulation requirements?

Assessing if a potential model complies with a set of requirements would be of clear potential benefit. If the model and requirements are clearly documented using natural language there is the potential for NLP to aid in the comparison. There are examples of off-the-shelf tools that can search for semantically similar requirements within a requirements set. This capability could also then be used to asses for similarities between a requirements set and the documentation of a simulation. If there were to be a repository of models with supporting documentation it would be possible with such an NLP application to pass over the library and recommend potential existing models as solutions to the design being investigated.

TESTING THE PROOF OF CONCEPT

The NLP POC is to be tested using the work that was conducted for the case studies (See section 4 for more information regarding the case studies). The POC is to be compared to the manual processes that were conducted. These tests form the bases of a verification of the POC and hence a reference point is required (as presented in section 3.5.2). In this case the reference point is to be against the manual work that has been conducted in the case studies. To remove as much bias as possible within the confines of this project, a number of steps have been taken, as indicated below.

- The code for the POC was written before any of the manual work was conducted.
- All parameters and dictionaries that form part of the application were set up with the simulation domain in mind but they were not tailored specifically for any singular document. The subject of the case studies was not finalised until after the POC was finished.
- Once the manual work was completed it remained unchanged regardless of the results of the NLP POC.
- The textual documentation that was used for the manual process and POC are exactly the same including words layout and formatting.
- The results of the POC are reported as they are outputted from the application in their raw form.

Due to this only being a POC it is not expected that the output will be comparable to the manual operations. However it is expected that an indication of suitability will be found. The output of these tests will then be used to make recommendations regarding the use of NLP technologies within this problem domain.

The raw results of all of the testing can be found in the section 9.5 as the program is a Python application that outputs to the command line, produces graphs, and writes to file, all of which have been copied and placed in this document.

5.3 PROOF OF CONCEPT TESTING AND CASE STUDY ONE

The squash court case study that was used as part of the case study chapter section 4.2 is a fully worked example and has all of the information that is required by not only the proposed processes but also for the NLP POC Testing. Each of the models has a '.txt' file that has all of the relevant information regarding many aspects of the component models. There is also a '.txt' file that contains all of the textual requirements for the intended simulation.

There are two parts of the proposed NLP POC case study one test. The first stage is the comparison between the model documentation and the simulation requirements. This is to ascertain if NLP could be used to select potential components from a repository. The second stage of the test is to compare the model documentation of each of the models with the documentation of every other selected model. This is to assess if NLP could detect if models and simulations are semantically similar enough for meaningful integration to be possible.

From the investigation into NLP technology and through the development of the NLP POC the direct comparison of two texts looking for specific words that appear in both texts, which have been written in a set format such as the integration tables, could easily be achieved and hence it has not been investigated further by this research.

5.3.1 FINDINGS FROM TEST ONE

This test is the comparison between the requirements document and the various potential model documentation. This test takes the form of three test cases. Input A is the requirements document and input B is the specific potential model documentation for that test case. The structure of the testing inputs is captured in Table 5.7 below.

Analysis NB	INPUT A	INPUT B
1	Requirements_For_Case_Study_One_Squash_Ball_Moving_Around_A_Court.txt	Documentation_of_a_Particle_Moving_in_Free_Space.txt
2	Requirements_For_Case_Study_One_Squash_Ball_Moving_Around_A_Court.txt	Documentation_of_Energy_transfer_model.txt
3	Requirements_For_Case_Study_One_Squash_Ball_Moving_Around_A_Court.txt	Documentation_of_Squash_Court_in_or_Out_Model.txt

Table 5.6: The inputs for case study one analysis test cases one to three.

Across the three test cases there were similarities regarding the percentages of tags and their distribution. Across the first three analyses *Noun Common* was the highest percentage tag, followed by determiner and preposition. Interestingly the common noun tag is a higher percentage in the in the model documentation than the requirements document despite this being the most common tag in both the requirements documents and model documentation, all other tags vary with either the model documentation taking a higher percentage. It is possible that the percentages of the tags in documentation

may be a way of identifying the general style, hence suitability for analysis for potential model integration. However the tag percentage alone is not a meaningful way of assessing the potential for meaningful model integration or even requirements compliance.

The company specific modelling term section of the application did not extract a significant amount of usable data. This may be due to the dictionary used not matching the vocabulary used in the documentation. This potential for using this as a method for capturing the information necessary for the integration tables would have to be altered to not look at the common company specific words but rather just the company specific words for that document.

The words that appear in both documents have proved to be an interesting part of the experiment. It was originally thought that the company specific word search would generate a significant amount of useful information, whereas a search of words that appear in both documents proved to provide more indication as to whether the documents pertain to the same information. However the meaning of the words that appear in both documents still requires an inductive step to ascertain if they are meaningful or not. For example, the *noun common* list in analysis one contains ['space' , 'date' , 'time' , 'file' , 'ball'] to establish if the documents relate to similar information the 'ball', and 'space' could be of potential interest whereas 'date', 'time', and 'file' may be of less interest. This pattern of some identified commonly occurring words being of more use than others is repeated across the analysis.

Across the first three analyses the number of identified noun-verb sentences varies. Each of the identified noun-verb phrase pairs was then manually assessed to verify if the phrase pair exists and is meaningful. An identification that is deemed meaningful is a pair that ether shares the same semantic information or the identified pair would benefit engineers attempting to understand if the integration would be meaningful. It became apparent that although two sentences contain the same noun-verb combination it does not necessarily mean that they relate to the same semantic information. This information was captures in Table 5.8 below.

Analysis NB	NB Noun-verb Sentences	NB Meaningful Noun-verb Pairs	NB Non Meaningful Noun-verb Pairs	Percentage of Meaningful Noun-verb Pairs
1	10	6	4	60%
2	8	4	4	50%
3	9	2	7	22.22%

Table 5.7: Number of identified noun-verb phrases with a breakdown as to if the match is meaningful for analysis one to three.

From the inspection of Table 5.8 it is clear to see that not all of the identified noun-verb phrases were meaningful in some case, such as analysis three; there were in fact more non meaningful noun-verb pairs than meaningful pairs. From

these results it is worth noting that the length of the documents is relatively short. For comparison see Table 5.8 for percentages and Table 5.9 for the length of the documents.

Document Name	NB of words	NB OF LINES
Requirements_For_Case_Study_One_Squash_Ball_Moving_Around_A_Court.txt	217	20
Documentation_of_a_Particle_Moving_in_Free_Space.txt	264	53
Documentation_of_Energy_transfer_model.txt	204	42
Documentation_of_Squash_Court_in_or_Out_Model.txt	409	96

Table 5.8: Documentation basic characteristics including number of words and the number of lines.

It is of interest that not all of the documents are the same length. The first three documents are around a similar number of words and lines whereas the fourth document is significantly longer in both words and number of lines. The tables above show that comparing the requirements document to longer documents does not necessarily mean that the number of sentences with the same noun and verbs in will increase at the same rate.

5.3.2 FINDINGS FROM TEST TWO

The second stage of this test involves comparing the three potential model documentation documents with each other directly. The tests were conducted in the order shown in Table 5.10. With the analysis conducted in this way it insures that the documentation for each model is compared with all other models documentation.

Analysis NB	INPUT A	INPUT B
4	Documentation_of_a_Particle_Moving_in_Free_Space.txt	Documentation_of_Energy_transfer_model.txt
5	Documentation_of_a_Particle_Moving_in_Free_Space.txt	Documentation_of_Squash_Court_in_or_Out_Model.txt
6	Documentation_of_Energy_transfer_model.txt	Documentation_of_Squash_Court_in_or_Out_Model.txt

Table 5.9: Analysis inputs to the NLP POC.

The outputs of the second stage of the testing produced some interesting results regarding the tag percentages when observations are made between the documents. With a change in the types of documents that are being analysed there would presumably be a noticeable difference in the comparison percentages. However the bar charts that are produced follow the similar distribution of percentages to the first three analyses, this may be an indication as to the similarity of the style and overall content of the documents being analysed. However on closer inspection the noun common percentages are far closer together across tests 4,5, and 6 than in 1,2 and 3. This is to be expected as the models are of similar phenomena and all of the documentation name explicit objects and actions rather than in the requirements document where abstract concepts are discussed.

The company specific common words searching function was no more successful in the second set of three tests than in the first three tests. The words 'inputs', 'outputs' and 'LabVIEW' were captured. This is not a useable or meaningful output from the analysis that would not be captured from a cursory glance at the documentation or the files that form the model and its documentation.

The numbers and quality of the identified noun-verb phrases has been assessed. The results of the assessment can be seen in Table 5.11 below.

Analysis NB	NB Noun-verb Sentences	NB Meaningful Noun-verb Pairs	NB Non Meaningful Noun-verb Pairs	Percentage of Meaningful Noun-verb Pairs
4	17	11	5	65%
5	18	10	8	55.56%
6	15	9	6	60%

Table 5.10: Number of identified noun-verb phrases with a breakdown as to if the match is meaningful or not for analysis four to six.

The number of identified similar noun-verb phrases is higher across the second set of analyses when compared to the first, as to is the percentage of meaningful noun-verb pairs. This may be due not only to the documents being semantically similar due to the information held within them but also the writing style being similar as all of the documents being model documentation, rather than comparing documentation to requirements. However there are more meaningful noun-verb pairs that were identified. This alludes to the fact that it may be more of a semantically similar rather than just being similar in style. The identifications that were made in the second test would be of genuine use to an engineer wanting to know if there is a potential for meaningful integration.

5.4 PROOF OF CONCEPT TESTING AND CASE STUDY TWO

For the second NLP case study the automotive example that was used in the case study section 4.3 will be used. Although this automotive case study was not a full example of the end to end proposed process, it does have sufficient documentation for the stage of the proposed methods that would most benefit from the use of NLP technologies, that of searching for models that comply with specified requirements.

The six tests that will form the bases of NLP testing in this case study are comparisons between the simulation requirements and the documentation that accompanied the potential models. The inputs to the POC application can be seen in Table 5.12 below.

Analysis NB	INPUT A	INPUT B
1	Requirements_for_a_Combined_Braking_and_Steering_System.txt	Modeling_an_Anti_Lock_Braking_System_Matlab_Documentation.txt
2	Requirements_for_a_Combined_Braking_and_Steering_System.txt	Vehicle_Body.txt
3	Requirements_for_a_Combined_Braking_and_Steering_System.txt	Power_Assisted_Steering_Mechanism.txt
4	Requirements_for_a_Combined_Braking_and_Steering_System.txt	Simple_2D_kinematic_vehicle_steering_model_and_animation.txt
5	Requirements_for_a_Combined_Braking_and_Steering_System.txt	Tyre_Simple.txt
6	Requirements_for_a_Combined_Braking_and_Steering_System.txt	Tyre_Magic_Formula.txt

Table 5.11: Inputs to the NLP POC for the NLP testing case study two.

The tag percentages graphs that were produced from analyses one to six produced some interesting results. The tag with the highest percentage across all of the documents is *noun common*. However the next two highest tag percentages vary. However the *noun common* and determiner tags are often in the top three. This may be an indicator as to all of the documentation that is produced relating to the model documentation is of a similar style.

The identified company words proved to be largely ineffectual at extracting substantial amounts of meaningful information from the documentation. The one reoccurring word that it was capable of locating was the development environment. Whereas it repeatedly recorded 'C' as a language in use as it was present as a single letter used as part of an alphabetic list to denote the third entry. This highlights one of the issues with using rule-based analysis.

The words that appear in both documents gave an interesting insight into the documents under test. The common nouns that were extracted often gave an indication as to whether the two documents pertained to similar systems. These on their own could not be used to categorically say if the models were compliant to the requirements, however it could be used as part of an assessment.

The number of noun-verb sentence pairs that were identified across the six test cases varies as does the quality and relevance of the sentence pairs captured. The percentage of meaningful noun-verb pairs was calculated to enable meaningful comparisons relating to the quality of the identified noun-verb pairs to be made between the six sets of analyses.

Analysis NB	NB Noun-verb Sentences	NB Meaningful Noun-verb Pairs	NB Non Meaningful Noun-verb Pairs	Percentage of Meaningful Noun-verb Pairs
1	12	8	3	67%
2	5	1	4	20%
3	3	1	2	33.33%
4	9	3	6	33.33%
5	7	2	5	28.57%
6	9	2	7	22.22%

Table 5.12: NLP case study two test results the number of identified noun-verb phrases with a breakdown as to if the match is meaningful or not.

The highest number of noun-verb sentences captured with the highest proportional number of meaningful noun-verb pair sets was analysis one. It is of interest to compare this to the manual assessment of the model documentation against the requirements that was conducted in section 4.3.21.

The 'modeling an Anti-Lock Braking System' model scored as one of the more compliant models on the RAG assessment but not the most however on closer inspection, it is in fact the model that is closest to intended subject of the simulation. Thence the results for analysis one in Table 5.13 shows promises for NLP in this domain. The results of the 'vehicle_body' are of interest as this model scored the lowest in the manual RAG assessment in Table 4.28 with the highest number of Red flags, and it also scored the lowest in the Table 5.13. This is a constructive result for the POC.

A point of interest is the performance of the tyre (magic Formula) model in this test as it was identified in the proposed process as being of potential use. However it was more of a secondary component system rather than one of the identified sub-systems, hence it scored relatively well on the RAG assessment Table 5.13 with only amber and green. However it scored the second lowest percentage of meaningful noun-verb pairs. This may be due to the inductive step that was taken in the manual process by which the engineer recognised a need in the requirements that was not explicitly stated. The NLP POC was only comparing what was in the requirements document with the models' documentation meaning that there would have not been the language as it was implied rather than explicitly stated. This issue of finding useful components that were implied rather than explicitly stated is unlikely to be solved with rule-based NLP technologies.

5.4.1 Results From Proof of Concept Testing

The testing of the NLP POC included two case studies from different domains, regarding different systems, and the documents used for testing were written by different authors. The NLP POC functioned and generated some interesting results.

Looking at the distribution of tags of the documents gave an insight into the differences between requirements documents and model documentation. It also demonstrated the effectiveness of off the shelf taggers. The tagging technology is such that some of the tags can be correctly identified even if the word is unknown to the tagger. It is capable of this by using the position of the word in the sentence and its surrounding tags. The results were that the distribution of the tags gave an insight into the percentages of nouns and verbs that were used. The high use of common nouns throughout the testing indicates the general nature of the documents. However the use of tag percentages is not sufficient to ascertain if the models that the documents refer to are compatible.

The identification of common company specific words and phrases requires significant work to be of any real value. What were thought to be general modelling terms were not general or extensive enough to capture the modelling terms that were used in the model or requirements documents. However the capability of being able to identify specific words and phrases from text has been established. It is proposed that this could still be a viable method if the dictionaries that are used to set the words to be found were tuned further to the application and organisation to which it is operating within. This could be of real use in the comparison of integration tables as it contains a restricted vocabulary and sentence structure which could be of potential issue to tagging technologies.

By comparing common words that appeared in the two documents insights were gained as to how using such a method can indicate if the document pertains to the same or similar information. In the two case studies it was the proper and common nouns that were the primary sources of this information. It also became apparent that not all of the tags were of use at this stage. For instance the parenthesis tag set contained a number of parentheses without any other information; the same is the case for the punctuation that was captured. It is proposed that this would be a useable tool for comparing strict sets of data such as the integration tables.

The noun-verb sentence pair testing produced some interesting results. It revealed that applying the systems engineering concept of noun-verb sentence pair matching is not as effective as it may well first seem. This may be due to the fact that the documents fully comply with standardised methods, but it is also recognised that the majority of existing model and simulation documentation do not comply, making this result more relevant for this problem space. By automating the capture of sentences with noun-verb pairs does not guarantee

that they are referring to nouns and verbs that are of interest or relevant to the task at hand. However the results do indicate that it is possible to at least use it as a means of ascertaining if the documents pertain to the same system, and to some extent how compliant the model is with a set of requirements. A semantic test of the results of the logical test could potentially mitigate this issue to some extent however this would require further research.

When comparing the results of these two case study tests with the initial remit of the application of NLP technology, there are significant indications that this POC demonstrates potential. With further development it could be of real use in the problem space as expressed in section 1.3. It is foreseen that NLP technology with further development could be capable of not only assessing if the documentation of a model could be of use when comparing it to a requirements document, but then also handle the task of mining the data necessary to complete the integration tables. It is also perceived that it could be made to ascertain if there is potential to meaningfully integrate two models based of the completed integration tables. The results of which could be fed back to an engineer who could evaluate if it is worth time to investigate further and complete the necessary integration. Such a tool would go a long way to simplify the integration of existing models and simulations to produce a full system test.

5.5 SUMMARY

In the work that has been conducted on the systems processes it was proposed that NLP could be of potential benefit in reducing the time it takes to complete particular process elements. This chapter documents the investigation and eventual POC test of NLP technology.

The seemingly simple task of getting a computer to understand text has been shown to be far from simple and indeed a well-established research field in its own right. With the current limited capabilities of NLP it has still been successfully implemented across multiple domains. Specific effort was directed at finding instances where it has been used within engineering industries and found to be of benefit.

The underlying methods that the current NLP tools adopt are discussed. The strengths and weaknesses of logical and statistical methods are highlighted. It was found that within the most recent NLP literature advocates of both methods still remain. There are those within the literature who promote the use of formal, prescriptive, or even limited diction languages as a means of reducing ambiguity and simplifying the NLP task. This has its obvious advantages but due to the defined problem space is unfortunately not a viable solution in this instance. From research that has been conducted regarding different domains and their use of the English language it has been proposed that each domain has its own interpretation of words, grammar, and syntax. The effects of these differing interpretations have resulted in situation that makes the NLP task significantly more challenging.

Currently available tools and packages for the purpose of developing NLP tools are identified and their capabilities assessed. The capabilities of the available tools and libraries indicated that there is the real possibility for individuals to develop their own NLP tools and applications.

From the information gathered regarding the current state of NLP a POC was developed to test the possibility and potential benefits of using NLP tools within the identified problem space.

The proposed systems framework (see section 3.2.4) was used and a customised spiral framework approach implemented. Requirements were captured and used as the basis for architecture and a design. This design was verified and implemented. After each new function was created the output behaviour was verified manually using developmental test files. The completed application was tested using the text files that were used as part of the case studies used in section 4.

The current limitations of computer science and the lack of a general purpose artificial intelligence, means that current NLP applications, when tasked with finding similar meaning phrases, do not have the same ability as people. With this

application of NLP technology, the result is that the application may be much quicker than engineers are at finding the majority of models and simulations that comply with a given set of requirements. However, the application is unlikely to capture them all. This then becomes a trade-off that an organisation would have to evaluate as to what will be cheaper in the long run. Engineers spending time looking through existing models and simulations documentation, for the chance that some will be useable; or running a NLP application, in the knowledge that some work that duplicates an existing model or simulation may have to be carried out.

The results of this testing indicate a strong potential for this technology to be of use to reduce the time it takes to conduct the proposed processes in section 3.

6 DISCUSSION

6.1 INTRODUCTION

When a model or simulation is created it contains particular aspects of knowledge from those who create it. The effects of this knowledge encapsulated by models are discussed as well as the challenges it brings to the integration task. A further issue discussed includes data storage of potential models. Ontologies were often given as the technological means by which an information repository could be implemented. Taking into consideration the deeper philosophical issues regarding model and simulation integration the suitability and practicality of using ontologies for the identified problem space is discussed.

With any new technology for engineering, consideration has to be given to the ways in which the current business practice works in that domain. When investigating engineering companies involved in modelling and simulation integration there are in fact deeper reasons why integration is currently being conducted. Some of these key challenges are discussed specifically with how they affect the integration task.

Automation has been attempted in many aspects of human endeavour, so it follows that the possibilities and current technological barriers to the automation of engineering tasks are discussed.

The potential paradigm shift necessary to capitalise on the potential implementation of model and simulation integration has been theorised. This is due to current standard engineering practices may not be in a position to accept virtual full system testing with current processes. The reason for this and possible ways to overcome it are discussed.

6.2 PHILOSOPHICAL ASPECTS

The integration of models and simulations raise many issues, only some of which have been captured in this work (see section 2). An issue so far not discussed, yet important for meaningful integration, is the decision-making of the modellers themselves.

6.2.1 PHILOSOPHICAL ARGUMENT OF HIGH FIDELITY SIMULATION INTEGRATION

Recent increases in computational power as well as improvements in computational methods, led to the possibility of the concept of high fidelity modelling and simulation to test complete complex engineering designs virtually. This has led to some optimistic statements in the literature such as "High-fidelity simulations of real-world systems, are truly transforming computational science into a fully predictive science." [69]. If this is the case then the future of engineering design will consist of an ever greater number of high fidelity models and simulations. In the next ten to fifteen years it is foreseen that high fidelity models with a suitable human interface will allow the virtual testing of a new product with stakeholders before physical prototyping is used. This testing will also be for unknown design characteristics where there are no existing experimental data [69]. This promises even more to engineers, for such a simulator would allow for unknown interactions to become known and analysed, further reducing the need for costly iterations of the physical prototype. This approach in essence allows for much more information to be gained from using simulations when compared to using low fidelity simulations in isolation.

6.2.2 REDUCTIONISM

The premise of reductionist thinking is that any difficult or complex system can be broken down into simpler less complex sub-systems or constituent parts. The behaviour of these sub-systems and elemental parts are considered to be understandable to the extent that they can be modelled. Often the reductionist approach has been explained in terms of divide and conquer, with the assumptions that decomposing a problem down enough times will lead to a point where the behaviour of the smaller parts is known and can be traced [9]. In theory this approach can be iterated in a recursive manner to the extent where the result is a model which only contains elements which cannot be broken down any further (referred to as an atomic model).

If the reductionist approach is correct then it would be possible to integrate the elemental models into sub-system models, and in turn work up through the model-based hierarchy until such a point where a full system model is created. Such a model could potentially be of a high fidelity to a pre-determined accuracy and mimic many levels of understanding.

However the reductionism concept makes many assumptions which include:

- It is possible to break down the complex situation into more simple parts
- The properties of the whole can be explained in terms of the properties of the component parts [117]
- When a system is decomposed into sub-systems none of the behaviours of the system are lost
- Each level can be understood by those involved with the modelling and simulation
- The behaviour at each of the levels can be modelled at a required accuracy and fidelity
- There are methods by which the component models can be meaningfully integrated together into one model or simulation
- A decomposed set of component parts can be composed together to form a representation of the whole.

Some of these assumptions may not be valid in all situations and this poses a problem which has been known for some time. In the early 1990s, papers with bold statements such as "…the whole being different from the sum of all the parts (failure of reductionism)…" [118] can be found. Such comments are not uncommon and continue to appear from time to time. Hence it is probable that some of these assumptions may not be valid in the problem space defined in section 1.3. For instance, the models to be integrated have been composed from a bottom up approach based on understanding of elements rather than decomposed from the desired (or existing) behaviour of the system. The impact of this means that the integration of the existing models will not be as straightforward as the reductionist approach dictates when working down from a complex system. For this reason there is likely to be obstacles as a result of the assumptions that will only become apparent when they are reached by those who are integrating the models.

6.2.3 The Need For Model and Simulation Validation

Models and simulations are created based on conceptual understanding of the phenomena that are being represented. This process often involves the use of the modeller's own mental models of the phenomena, and how it is expected to behave when exposed to specific environmental conditions. As our collective understanding of the way in which the world works around us grows, so to in turn have our abilities to represent it in the forms of models and simulations. In the well-known words of Sir Isaac Newton "*We are standing on the shoulders of giants*". With the information explosion that has happened we now have access to information like never before. One consequence of this is that models and simulations that are currently being used in industry can represent systems with a level of accuracy that have never been achieved before. However despite all of this knowledge and understanding there is still a disparity between the real world phenomena and the models and simulations that are being used to represent it. There is a common consensus across the more philosophical literature that

although we think we know a lot about the way the world works around us, we do at present not know how everything works and interacts.

There is the argument that if a system is modelled with absolute accuracy the model will become the intended system being developed rather than a representation of that system. This argument is somewhat flippantly referred to in some documents, however this makes the assumption that it is impossible for us to model or simulate a phenomena and its environment close enough to actually predict the behaviour with an incontestable accuracy. This is an enticing idea, however it does have the potential to hinder those who may otherwise strive to produce ever more accurate modelling and simulation techniques. In practice however this argument can be considered academic as we just do not have such an intimate understanding of phenomena to allow for this level of modelling to be achieved. A quote that is often given when discussing such matters is one by G.E.P. Box which states "All models are wrong but some are useful" [119]. This is where there is a notable difference between scientific and engineering literature. The scientific literature is focused around furthering knowledge and being ever more correct, accurate, and precise; whereas the engineering literature is far more focused on getting a product out faster and in a more cost effective manner irrespective of how correct the modelling may or may not have been during the development. For this reason many engineers know that the models and simulations are technically wrong, or will not be a complete picture of what will happen in practice, however they are good enough to get the job done.

Due to this lack of complete understanding (whether this is recognised or otherwise) there are common tactics that are employed within engineering to model a potential design before physical prototyping takes place.

Such methods include:

- Assuming that a large number of identifiable factors are considered negligible (idealisation)
- Using abstraction to take complex situations and make them easier to comprehend
- Physical prototypes are built and measurements are taken.

The resultant effect of the uses of any or all of these approaches means that there will be an inevitable disconnect between the model or simulation and the behaviour of the phenomena being represented. Assumptions and implications will be present in the perception of the modeller and hence their work is a manifestation directly affected by their assumptions. This raises the issues of how different from the phenomena can the model or simulation be and still be of use. How to understand the differences between the model and reality caused by the modellers assumptions also becomes a challenge. This is where the validation

of modelling fits into the modelling and simulation process. Validation of the model or simulation which can be described as "process of determining the degree to which a calculation method is an accurate representation of the real world from the perspective of the intended uses of the calculation method" [120]. This often takes the form of a physical prototype for final verification and validation in the system environment.

6.2.4 CHALLENGES WITH KNOWNS AND UNKNOWNS

If the accepted principles of systems engineering are used during a project one of the first stages is to produce a set of requirements. Requirements often contain a spectrum with fully understandable known parts at one end to completely unknown aspects that will only become apparent in the final products behaviour in its intended environment. Within the literature there are those who see requirements engineering as somewhat of an art form, as it is concerned with the discovery of the previously unknown in context of a design, despite abstraction, uncertainty and ambiguity [121]. This shows that there is an acceptance of the complexity of the system being designed and the impact on requirements. The result of which is that complete understanding is a significant challenge and is often considered impossible and indeed in many cases not needed. This can lead to aspects of a design being in the requirements without knowledge of whether it is possible or not.

A system and its components will have attributes, operations and interactions. Many of these are known to the modeller and are quantifiable; these are referred in this work as 'known knowns'. Though this is not the case for all attributes, operations and interactions, as there are occasions when there are unknowns that are not derivable from the model or simulation. In this case they are considered to be 'known unknowns'. There are ways that modellers have to find out 'known unknowns'. They use computation, or many require obtaining the data from elsewhere (physical testing), making an educated estimate of the appropriate value, or essentially bypassing the unknown aspect by developing the model to account for the known uncertainty. In some cases it may also just be preferable to state the unknown as a limitation of the model or simulation.

There can also be a situation where the modeller is unaware of attributes, operations or interactions; however there is information available to quantify it. This is considered to be an 'unknown known'. These can be identified if a rigorous verification process is undertaken, then it becomes a case of restructuring the model to account for the overlooked 'unknown known'.

The most significant challenge to the uncovering of unexpected behaviour of a whole system is that of the 'unknown unknowns'. This is where the modeller is unaware of the attributes, operations, or interactions. This means that the resultant model may not accurately represent the behaviour of the system as a whole [18]. This represents a significant challenge for any modeller. It is a requirement for modellers to become well versed in the domain they are

modelling and understand it to a high extent. This highlights the need for specific domain experts with modelling capabilities.

Attempts have been made as to how to represent this spectrum of knowing. One such attempt uses a graphical representation with axis for the types of knowns and points plotted [122]. This representation also takes into consideration the aspect of the number of actors involved in the process. Each actor's perception is captured in a single frame. These representations may not in themselves be of practical use, but they do illustrate the problem of what is known and by whom.

As systems are becoming ever more complex, it quickly becomes difficult for one person to grasp an understanding of the whole system – this is especially so when the concept is as a result of a vast number of integrated models that are based on the work of many people. With many modellers involved there may be attributes, operations and interactions that are outside the individual model components. If the models are developed in isolation without the intention of ever running in an integrated fashion, the likelihood of there being unknown unknowns in the resultant model is high.

6.2.5 Impacts of Hardware and Human in the Loop Testing

There are those who have incorporated partial physical prototyping with modelling and simulation. By bringing in the physical components and subjecting them to suitable testing allows for sections of the final system to be tested before full system assembly. There are sometimes difficulties with predicting the behaviour of particular interactions from first principles and as such physical prototypes are made to test how they react with their intended environment. By using Hardware In the Loop (HIL) testing these unknown or difficult component behaviours can be integrated into a simulation to ascertain how they will react with the rest of the system. In essence increasing the fidelity of the model and increasing the resultant accuracy of the outputs. Figure 6.1 is a simplification of the general component parts of a HIL test. The figure represents software simulations that produce inputs to the hardware by way of a software-hardware interface. Dependent on the hardware and test in question, signals may be returned by the hardware itself and or by way of sensors to the software components. Often the software part of the HIL test represents more than one single component and may also have a models or simulation integration aspect. The HIL approach allows for portions of the system to be implemented and simulations to bridge the gaps between and around the physical components.

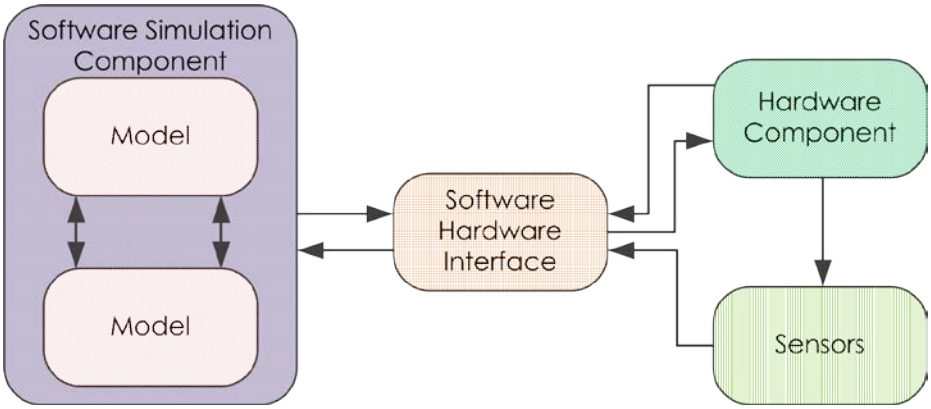

Figure 6.1: Hardware in the Loop Simulation parts

Within the automotive industry HIL testing has developed beyond the point where single components are tested, and now include networks of components that can be tested [123]. By testing the networks of components before they are assembled as part of a full prototype allows for the interactions between them to be tested earlier on in the product life cycle. There are instances within the literature where HIL testing and model-based design have been combined into a single process [30].

Human In The Loop (HITL) is similar to HIL testing but where in HIL there are physical components, HITL also has a person or persons as part of the test. The Figure 6.2 below shows a simplified representation of the component parts of a HITL test. HITL comprises physical components as well as a people. This is shown in the Figure 6.2 below with the interactions between the human-machine interface, the human, and sensors. Human in the loop is where a potential user interacts with a simulation of the system being developed. Often the intent is for the simulation to run at the same rate as the real system it is designed to mimic.

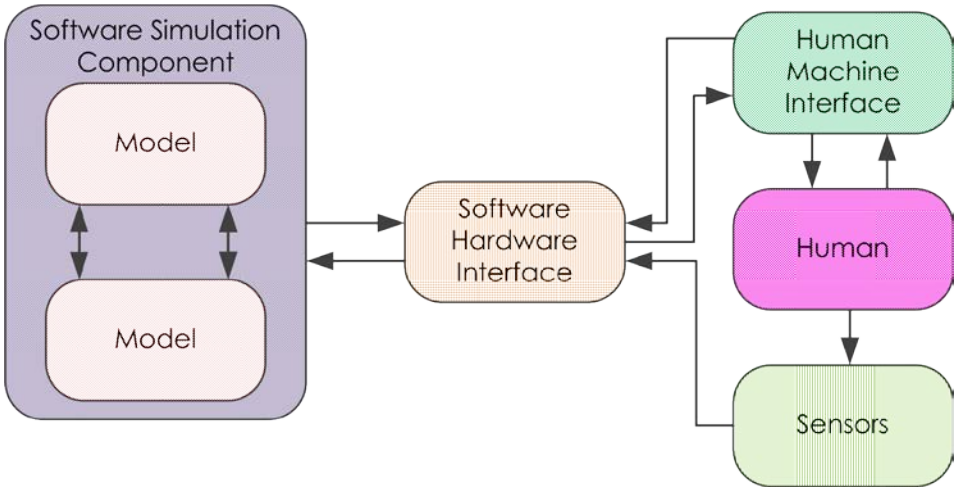

Figure 6.2: Human in the Loop component parts

The purpose of such HITL tests are to ascertain if the human-machine interface is acceptable as well as if the system reacts in the intended way. From a philosophical perspective the involvement of a person in the simulation brings the

simulation environment closer to the systems operational environment. By bringing the environments closer together allows for more unknowns to be incorporated into the testing of the potential design. HITL has been adopted as a method of testing in the field of ergonomics and is considered to be an effect way of asserting the suitability of a design when considering the human interface [124].

HITL is a current hot topic within the engineering academic literature with it making the editorial of IEEE transactions on automation and engineering [125] where HITL is identified as being a key tool to be used for the investigation of automation. This is a good indication that HITL will gain more popularity in the future as a means of testing the design of systems, as well as how people behave under new operating conditions.

HIL and HITL are both attempts at making simulations closer to the system being emulated by bringing actual components of the system into the simulation. This is a method that has been demonstrated to give valuable results and looks to continue to be used in industry.

6.2.6 System Creators, End Users, and System Customers

In an engineering application it is rare for a system to be requested, created, and used by the same group of individuals. In many engineering projects these are three different groups that are involved throughout the system lifecycle. There are also many other categories of people who interact with the life cycle of a system. However for the purpose of discussing the philosophical aspects of the problem space, these three categories will suffice and are shown in Figure 6.3 below. In this context the 'System Creator' refers to those who design and build the system, the 'End Users' refer to those who will be interacting with the system, and 'System Customers' are those who are buying or paying for the system to be created.

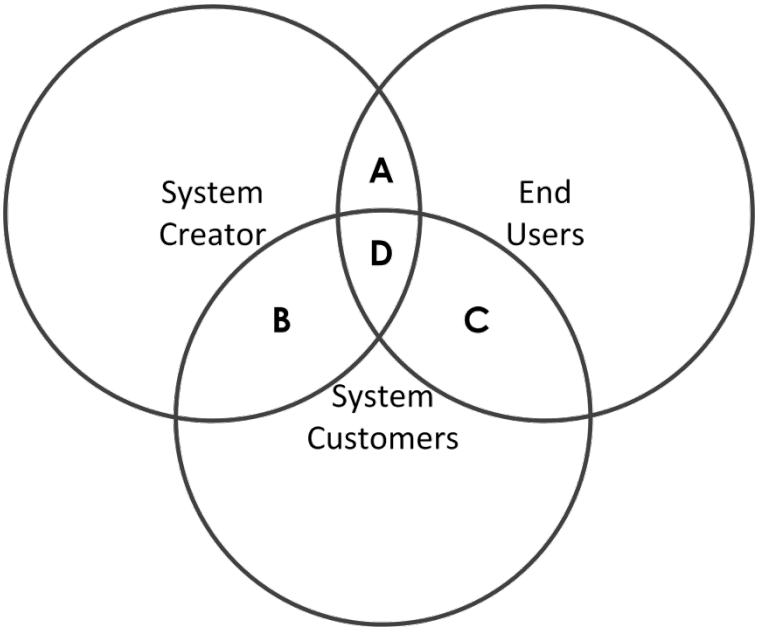

Figure 6.3 The intersections between the three groups involved with the development of a system; customers, creator, and end users.

In Figure 6.3 above the letters A,B,C, and D reflect the overlapping areas of concern between the groups of individuals. General themes that appear across the literature will be discussed using the three categories and the four identified overlapping areas. The most common challenge that is discussed is that of the communication necessary between 'system creators' and 'system customers'. This is represented by the letter B. The HITL testing demonstrates that the overlap A is of concern in many engineering projects. The interaction C represents situations where those commissioning the system may only be a sub-set of those who will be using the system. As long as the 'systems creators' have the opportunity to interact with the 'system users' importance of C is reduced. Ideally the area D is where at least one closely working team in the project would sit ensuring that the concerns of all involved are considered. Which all points towards the underlying issue which does not generally get mentioned in the literature which is that the people building the system are often not the ones who will be using or financing the creation of the system. For all of the parties to be happy with the end design the viewpoints of the three groups also need to align. Many of the systems engineering tools and approaches are concerned with positioning the project (or at least some of the engineers) in the area that is shown in D.

6.2.7 Virtual Simulation Vs Physical Prototype

There are those who are pushing towards simulation and prototyping to be fully virtual. This is a process that would model and simulate the system designed, and the first instantiated instance of the design will be the first product off the production line. This is an interesting idea and one that is obviously enticing for higher level management, as if this became successful it could reduce not only the financial cost of a project but also reduce the development time. However

this idea has some potential issues. If the statement in section 6.2.2, 6.2.3, and 6.2.4 are correct, at present we do not understand the world around us in significant detail to allow for all phenomena to be modelled with suitably high fidelity. Significantly, even using a series of validated models that are integrated may not collectively mimic all of the behaviours that will be present in the final solution. This is due to models only representing what is known. Some of the interactions are unknown until the design is physically implemented. Not all forms for prototyping are appropriate for all situations as it depends on the system being designed. Some forms of prototyping are better at producing the necessary information than others at various stages of the product life cycle [126]. Hence at present virtual prototyping is not at a stage where it can fully replace physical prototypes. However there may come a time when it becomes possible.

6.2.8 The Need and Place of Physical Prototypes

It should not be neglected that physical prototypes are prevalent within industry, be they for sub-systems or the system as a whole. Prototypes are used not only to validate designs as described in section 6.2.5, but also acquire the values that may otherwise be too complex or timely to calculate to the accuracy necessary [127]. In essence a full system prototype has all of the interactions that the full system will have when implemented. Any 'unknown unknown' (as defined in section 6.2.4) become apparent. This is the crux for the need of prototypes. For a physical product, if the first full system to be assembled is the first one off the production line, and if there are any interactions that the modellers and designers were not aware of, the behaviour of the system as a whole may be significantly different from what was expected.

A key area within the automotive industry where physical prototypes are needed is that of safety ratings. Crash testing is difficult to simulate and the preferred way to ascertain how the vehicle will behave is to subject it to the forces that it is likely to experience in a crash. The Euro NCAP is a non-profit, non-governmental organisation which requires physical prototypes to be supplied for its destructive tests [128].

By moving to full unit tests there may be interactions that affect the behaviour which would not be identified without the interconnection of the component models and simulations. However prototypes will still demonstrate behaviour, be that subtle or otherwise, that do not appear in models due to the lack of understanding of the system by the modeller. The move towards full unit simulation test will not completely remove the need for physical prototypes. However it should reduce the number of iterations of the prototypes having to be made.

6.2.9 Time Spent on a Project

There is a question as to where best to use engineers' time when conducting a project. A somewhat simplified version of how systems engineering principles are implemented is that a significant amount of time is spent on understanding the

problem space, ensuring that effective means of communication and decision making is established, while ensuring traceability throughout the process. The intent is that spending time on these tasks will reduce the overall time that the project takes by ensuring that the project goes right first time, thus reducing rework. This approach has been adopted across many industries and has shown to be of benefit.

There is a trade-off between the time that it takes to conduct the additional systems tasks and the time that rework is needed using prototypes. This concept of the trade off in time and effort can be seen in Figure 6.4 below.

Figure 6.4: The trade-off between the systems engineering approach and the traditional use of prototypes.

In the example in Figure 6.4 more time effort and other resources has been invested in the systems engineering than in the prototypes. However it is recognised that this may not always be the most effective use of resources for all projects, or even across all areas of the same project. There may be instances when systems engineering methods may not be of significant use to engineers who are better of spending time working on short term goals using traditional engineering approaches. This decision is finely balanced and difficult to optimise.

6.2.10 Impact of Increased Computational Power on Modelling and Simulation

The ever increasing amount of computational power that is becoming available to engineers at cost effective rates has the potential to change many aspects of existing engineering processes, as well as making new processes possible. Computational power continues to increase, which leads to questions as how to best use this resource. Not only does this mean that ever more complex algorithms can be executed, but also previously disregarded methods due to computational overhead or sub-optimal use of processors can be considered for use again. Some of the previously disregarded methods are brute force in nature meaning that they work through every possible combination of problem, or are methods that are quick to set up but take a comparatively long time to execute.

The latter option is where HPC are perceived to make a big difference. The time is fast approaching where the engineers' time is worth more than the computational time that the computers take to execute a simulation.

6.3 SIMULATION AND MODEL INTEGRATION ISSUES

There are identified issues that surround the integration between the implementation of candidate models and simulations. Some of the issues are related to the way in which the modelling and simulation types are implemented and others are due to the viewpoint of the modeller who made the model or simulation in the first place. Such philosophical issues are discussed below.

6.3.1 IMPLEMENTATION PLATFORMS

To implement a model or simulation a platform to support it is used. This could be, software, hardware, or even pen and paper. Hence the means by which the concepts for the model or simulation have been captured play a significant role during the integrating task. It may be desirable for simplifying the integration task that models and simulations all use the same platform. However this is a challenge in its own right as the changes must retain all of the same information but capture it in a different way.

There is a difference which needs to be identified. The term platform is not only used to refer to developmental tools used for simulation [63] but can also be used to describe the engineering solution that is being implemented if it uses a platform approach [75]. The term 'platform approach' in solutions often refers to a single general base solution that can be modified for a specific solution when conducting model and simulation integration.

The difficulty of integrating multiple simulations from different implementation platforms is by no mean a trivial task and has been the subject of many research projects [63]. This is the task of taking a model or simulation produced on one platform and either getting it to run on another or enabling communication between platforms. To combat the different simulation platforms there have been attempts to make umbrella solutions that allow for the integration between multiple platforms [63]. However they often have significant limitations as to what they can communicate between. Standards have also been used in an attempt to harmonise the communication between simulation platforms. The FMI standard recognises that different platforms exist and are used for simulation. This is one of the key reasons for the creation of FMI [22] as previously discussed in section 2.3.3.

6.3.2 ABSTRACTION

Many engineering projects have too many individual components and are too complex for a single person to have a complete detailed understanding of the complete system. Therefore abstraction of the system to reduce detail to gain an understanding of the whole is often used. Abstraction within modelling and simulations domains is discussed as being a means of coping with the complexity as well as being a potential source of contention within integration tasks.

The importance of the concept of abstraction is by no means overstated as in the literature such statements are made "Abstractions are essential in handling computational and modelling requirements of complex systems." [129]. This is the means by which behaviour can be conceived and judgements made without the need to fully understand all of the exact workings of the system. The higher the level of abstraction, the more of the overall system it is possible to understand with the trade-off of not expressing the detailed finite working of the system in question. This should not be confused with fidelity. The trade-off between system-wide understanding and detailed finite understanding is a constant balance that the modeller has to work with. Too much abstraction it is possible for important details to be lost whereas if too little abstraction is used too many detailed workings will be expressed leading to confusion.

Due to the way in which we perceive the world around us, as soon as we create a model it will inherently have a level of abstraction built within it. This means that when considering abstraction it is not as simple as there being only two levels (a low level and high level) but rather abstraction is a shifting scale with models at many different levels within it.

The level of abstraction is based on the assumptions and understanding of the modeller as well as what is seen as relevant to include within the model. Due to the very nature of the concept of abstraction it could be considered to be the encapsulation of the specific viewpoint and hence the catch all of the differences between models. Within the literature there are concerns raised regarding model re-use and the inherent contradictions that arise in identifying the appropriate level of abstraction to implement [14]. Therefore the brining together of models of differing levels of abstraction poses many challenges, which need to be overcome for meaningful integration to occur.

Abstraction is not purely held by the modeller alone but it is also inherent in the tools that are used as most tools allow for different abstractions to be captured [20]. With this in mind it is not unreasonable to see that there can be conflicts between the abstractions of the modeller and the modelling tool. Therefore two models made in the same environment cannot necessarily be considered to be at the same level of abstraction. Within the literature such challenges of brining models of differing abstraction together are recognised and even identified as a problem with model re-use [129]. Whereas other sources go further and state that "the major technical pitfall might lie with the abstraction challenge" [14]. This then makes the need to find a means of harmonising multiple levels to allow for meaningful integration more pressing [130].

6.3.3 Fidelity

As with abstraction, fidelity is not a term that has a unified definition throughout the literature and in many instances a definition is not supplied. The standard ISO 15288 states "The necessary level of fidelity is a factor in determining the appropriate level of rigour." [67] It is stated without real definition to explain what

is meant by the term fidelity. Often fidelity is linked with accuracy as in [61] where the accuracy is referred to as how close the results of the simulation matches that of the physical phenomena being simulated. This is the definition that is going to be used in this work.

This raises the issue of why would engineers use anything other than the highest level of fidelity possible in any given situation. The concept of modelling at differing levels of fidelity allows for engineers to walk the line between the accuracy of the model and resources needed. The higher the fidelity, generally the more effort is needed to produce the model as well as any simulations based on it. The use of low fidelity models and simulations still have their place in engineering and there are examples where low fidelity modelling has been of benefit [14].

Issues arise when models at different fidelities are brought together. If the outputs of a low fidelity model were fed in as input parameters into a high fidelity model (rather than measured or estimated parameters) there is the logical argument that the accuracy of the outputs of the high fidelity model is compromised. Due to the sensitive nature of some high fidelity models, the starting values have a significant impact on the final results. This means that the integration of two models, one of high fidelity and one of low fidelity, can be of less use than using two low fidelity models. The converse of this gives a very different outcome. With a high fidelity model feeds into a low fidelity model it does not affect the accuracy of the outcome of the low fidelity model. However there is still little benefit in this configuration over the integration of two low fidelity models. So this raises the overall question of how different the fidelity of the models can be before the results of integration become meaningless.

As with abstraction, some tools have different capabilities of modelling or simulating different phenomena at different fidelities. There are concerns that using a single vendor, with the current tools on the market, that for a complex system a unified level of fidelity could not be possible as no one tool can adequately model or simulate the whole system at a high fidelity [123]. This indicates the potential value of integration methods between different tool vendors.

6.3.4 TIME

When bringing models and simulations which were created in isolation together, it is unlikely that they will use the exact same time base. This becomes even less likely when models and simulations are created using different fidelities and scales.

When it comes to combining time bases "conventional approaches to sampling signals or images follow Shannon's celebrated theorem: the sampling rate must be at least twice the maximum frequency present in the signal (the so-called Nyquist rate)." [131].An example: if a model is based on time T and the second

model ran at 0.0001T then the second model would be calling for data that were not there. A common means by which this situation is handled is by using interpolation. This works by using a straight line approximation between the points of the first model. Such a method can cause significant issues when simulations are executed in parallel.

There are issues surrounding co-simulation and the time that the various parts execute. If the mathematical solvers do not start at exactly the same time then there can be a difference in the clock times between the models. Any differences between the solvers can lead to irregularities. These irregularities can result in inaccuracies in the results. As a consequence some solvers allow their clock rates to be shared outside their execution, or even set externally.

If the simulations are executed across a network, there are also considerations with communications and model execution. The latency in networks is a common topic in computer science. The literature that is concerned with federated simulations often discuss issues and express that simulation time and parallelism poses a considerable challenge [48]. Methods have been developed to mitigate some of these issues, however all add to the time it takes for the simulations to run.

6.3.5 Local and Distributed Integration

The challenges to integration increase when models and simulations are integrated across a network when compared to the same integration on a standalone machine.

When running integrated simulations on a single machine the issues of resources and their allocation become paramount. It is possible to produce a simulation that exceeds the capabilities of a single commercially available machine. For this reason there have been efforts to distribute the processing across a network. This is to negate the need to use a single high performance computer to conduct computationally intensive simulations. However there are other challenges which result from distributing the processing across a network.

By taking the processing and splitting it up there are issues of orchestrating what is to happen and when. This orchestration is mentioned in literature such as in the FMI standard [26]. However this orchestration is not often discussed in detail as to how to accurately implement it. Often tools such as Isight [8] state they have orchestration capabilities which is used as a competitive advantage and as such the way that it is implemented is not published in available literature. Within the orchestration there is the potential to compute sections in parallel as well as sequentially, which makes the access to shared values an even greater challenge. The access to memory and how values are stored or distributed across all of the component parts is another challenge. Some methods such as Data Distribution Service (DDS) broadcast all of the calculated information to all other components after each stimulation step. This is acceptable as long as all of

the components iterate at a similar rate, and as such this too is still an area of research. Keeping track of where all of the intermediate files are stored as well as the component parts can be an issue when a system is distributed across a network. There are manual ways of keeping track of such information however this quickly scales up to the extent where this method is not sustainable. Hence there have been many research projects [129] which consider resource management of a distributed simulation.

Delay or latency across a network can be an issue for many distributed systems. This is also the case with distributed simulation [132]. Due to the way in which many integrated simulations function, network latency does not just mean that the simulation will take longer to complete the desired test, but also may affect the integrity of the results. One such issue caused by network latency can be expressed if a producer and consumer architecture is considered (such as that discussed in section 2.4.2). If such a distributed simulation is implemented where each component uses the most recent data value available on the network, without consideration to simulation time stamps, the integrity of the results can be questionable. This is due to there being no guarantee that the consumers are using data from the same simulation time step as the producer, due to differences in latency across the network.

Maintaining the distributed system which utilises services is a research field in its own right and there are many researchers who specifically focuses on how to formalise the switching in and out of various services based on their performance [133]. As a task this has been shown to be non-trivial. This indicates that distributed computing is still developing and there are challenges that would need to be addressed by any engineering company looking to utilise distributed simulation as part of their design process.

6.4 Current Methods of Storing and Interrogating Models From a Repository

The storage of models and simulations and the data that they produce is not the focus of this research, however it needs to be addressed. It is recognised that processing power and data storage are not infinite and have costs associated with them. This brief overview of data storage, cloud computing, and big data covers the key areas when contemplating a large scale integration project.

Throughout the literature there are many technologies and methodologies that demonstrate having a centrally stored repository of models and simulations within a organisation. However few discuss these methods or means of implementing such a repository. An overarching issue that many comment on is the seemingly simple issue of storing data. To put the data storage issue in perspective "…2007 marked the cross over year in which more digital data was created than there is data storage to host it." [134]. This does not mean that engineering projects (that this study is considering) are going to produce more data than could possibly be stored in the entire world, however it does put our current data producing capabilities in perspective compared with our ability to store the data we are producing. Some have gone as far as to say that the management, organisation, access, and preservation of data is a grand challenge of the information age [134].

There is a general consensus that the cost of storage of digital data is going down both in a local and service sense. However there are those [134] that have noticed this trend as well as the cost of maintaining data centres (where project data are stored) for the required amount of time, is taking an increasing percentage out of project data budgets. To have the data stored alone is not enough, it also has to be secure. Indeed the loss of large amounts of data or otherwise still makes national news from time to time [135]. The cynical approach to such news is these are just the cases that are publicised; the likelihood is that far more data are lost and not reported. All domains that are currently holding large amounts of data now have to consider stewardship plans for the data, how it is to be stored, and still be accessible when needed.

One potential solution to the vulnerability of storing all of the data in one physical location is to take advantage of the growing field of cloud computing. At first glance this appears to fit well with model and simulation integration. It is a logical step to ascertain if it is possible to take advantage of this technology. An enticing aspect of cloud computing servers is that on the most part provide a computation or storage service while shielding the customers from the hardware and software infrastructure that is required to provide the service [136]. This could a significant advantage for some engineering companies as they could focus on the engineering of the solution rather than the infrastructure that is needed to implement the modelling and simulations that they require. Due to the way in

which many of the cloud computing services are set up "using 1,000 servers for one hour costs no more than using one server for 1,000 hours." [135], allowing for many more architectures of solutions to be utilised without some of the issues of hardware resource availability found in Local Area Network (LAN).

There are issues with cloud computing which are often found within the literature such as security and the proprietary APIs that are currently used [135]. It is surprising to some however that organisations object to cloud computing due to the data not staying local to them, whereas they use 'secure' email servers that are provided by an external organisation. For some cases it may be more appropriate to use a desktop grid (harnessing the unused power of organisation computers not running at full capacity) rather than a public cloud. Research investigated the cost benefit analysis of cloud computing and desktop grids [136]. They found that depending on the number of flops that are needed and the number of available nodes, both approaches can be financially viable. It is not as simple as one is cheaper to implement than the other.

The use of could computing may not be as applicable to as many situations as proposed, as Larry Ellison the CEO of Oracle indicates,

> *The interesting thing about cloud computing is that we've redefined cloud computing to include everything that we already do…. I don't understand what we would do differently in the light of cloud computing other than change the wording of some of our ads."* [135].

As the number of component models and simulations that are integrated together in a single simulation increases, so too does the total size of all the data that are created. If large scale high fidelity simulations of full systems are to become commonplace consideration as to how the resultant data sets are to be integrated has to be considered. Many of the popular data base protocols such as eSQL when it comes to integrating data sets rapidly start increasing in run time as the set grows past one million lines of data, and so the realisation is that it is easier to get the data into a database than out [137]. This is where the research field of big data comes into play and the new experimental methods of handling large data sets become directly applicable to current data mining problems.

6.5 ONTOLOGIES AND THEIR USES FOR INTEGRATION

Ontologies are a means by which information can be stored in a structured repository in a form that allows for not only queries to be made about the stored entities but also regarding the relationships between them. There is evidence that the application of ontology technology to the problem of model integration has been investigated. Key areas have been identifed where the storage of models and or simulation or even the component parts there of could aid organisaitons in the development of complex systems [138]. Having the component models in a form that can be integrated and the relationships between the possible variations captured has obvious potential to reduce the effort needed to find usable components in future projects. There are many other research teams that have also recognised the potential and have been working to exploit this potential.

It has been shown that ontologies can help to alleviate specific aspects of the integration problem [129], [130], [139]–[142]. It is to be noted however that none of the identified literature is concerned with identical problem space to this work though they do share some similarities. One of the common themes identified throughout the ontological literature is that this technology gives greater value to an existing repository of data when compared to a simple indexed repository. This increase value is due to the relationships between entities being recorded and in a way which can be integrated. Hence ontologies can be used within modelling and simulation to construct code libraries that contain revisable knowledge bases of structure domain specific information that does not need to be captured for each application but is held at a more general level [138]. One of the most significant issues with creating a usefull onltology is the capturing and reprentation of the identified relations that proves to be one of the greatest challenges [138]. Algorithams can be developed to significantly improve the semantic search of heterogenous information sources, meaning that the intial structure and form of the ontology is less of an issue compared to the traditional search process [143]. This work shows promise for being capable of handling a variety of data sources and types. This is something that would need to handled, as in the model and simulaiton domain there are many different domains that all work in unision, and any data storage means would have to handle the variety of domains.

There are those who have been working with ontologies for the purpose of model and simulation integration [144]. A common theme across the ontological model simulation literature is the concept that a practical application would be to have a repository of models and simulations that could be queried to ascertain which model could be integrated as well as how to go about connecting them.

6.5.1 Ontologies Applied to Simulation Integration

In the literature there have been many attempts to use ontologies for the integration of models and simulations with varying levels of success. A common area of the integration problem research are attempts to automate the mediation, organisation, and exchange of semantic information between the component models or simulations [130]. Different ontologies use different means of describing the entities and the relations between them. These descriptions can be customised to the project such as "Discrete-event Modelling Ontology (DeMO)" [143] and "Framework for Adaptive Modelling and Ontology-driven Simulation (FAMOS)." [129]. While others have experimented with using existing well established descriptions such as "Web Ontology Language (OWL), OWL Lite, OWL DL (Description Logic) and OWL Full" [143]. There are both strengths and weaknesses in using custom and well established ontologies. Having a custom ontology allows for greater flexibility to capture all the relevant information but makes integrating ontologies together a significant issue and often requires custom tools for their creation. Whereas using a standard ontology makes the integration between ontologies less of a challenge as well as there being off-the-shelf tools for their creation, however not all information may fit the schemas.

The goal of using ontologies for modelling is to answer the question of "How do the components of different types of models relate?" [143]. Identified taxonomies exist in other fields in computing and the proposition is that a taxonomy must exist and hence it must be possible to be captured in an ontology. The major challenge in capturing the information is that of 'inter-process mismatches.' [130]. This term describes where the models and simulations are created based on different concepts and are executed in different environments.

A way in which semantic mismatches have been limited throughout the literature is by using a standardised schema such as XML [28], [63], [66], [142], [145]–[147], however this does not eliminate the issue completely. Incidentally, it is for the same reason that there is a common method used in SOA applications see section 2.6. When storing information in ontologies the concept of depth and breadth has to be taken into consideration.

> *"If an ontology is too broad and deep, it will be very hard to develop and maintain as well as having implications for automation."* [144].

If the ontology is too narrow or shallow it will not be capable of capturing the diverse range of information. There is not a great deal of practical guidance on depth and breadth but rather there are statements such as

> *"One may argue that ultimately a good ontology should seamlessly capture all (or most) of the naturally existing hierarchies within itself."*[143].

For practical engineering terms this is not helpful and may point toward a subjective measure rather than an empirical boundary. For the application in this

problem space there are serious problems with this as the models are disparate and the environments are heterogeneous, meaning that a single schema would be a significant challenge in itself. Even an existing flexible standard, such as ISO 10303-239 (defining information exchange throughout product life cycle) has been found wanting in problem spaces similar to the one in this study. Once a schema has been decided upon it is then possible to build a database to implement the ontology. Data entry is another issue and in most of the literature this is not considered. For a solution such as this to be used in an organisation the entry of all of their existing models and simulations may take many thousands of work hours just to capture the necessary information defined by the schema. This work would need to be carried out by skilled workers with a detailed understanding of the models in question and how to use the schema to capture it. This time resource factor cannot be ignored as a potential factor for the adoption of this technology.

6.5.2 How Ontologies Can Aid in Integration

A method that has demonstrated real promise is the use of a number of ontologies which are domain specific (using domain specific language) with an overarching meta ontology that consists of Neutral Simulation Language (NSL). This method of linking ontologies together is demonstrated in the linking of the DeMO ontology to the SUMO and SUO ontologies. [145]

Mapping of concepts between the domain specific ontologies and the meta ontologies can be used to integrate models together as the mapping is present both in the ontologies and the models themselves allowing for model translation. "The idea of ontology-driven translation uses ontologies as the foundation for translating information from one simulation application to another." [145]. The method of using three ontologies to translate models is shown in Figure 6.5 below.

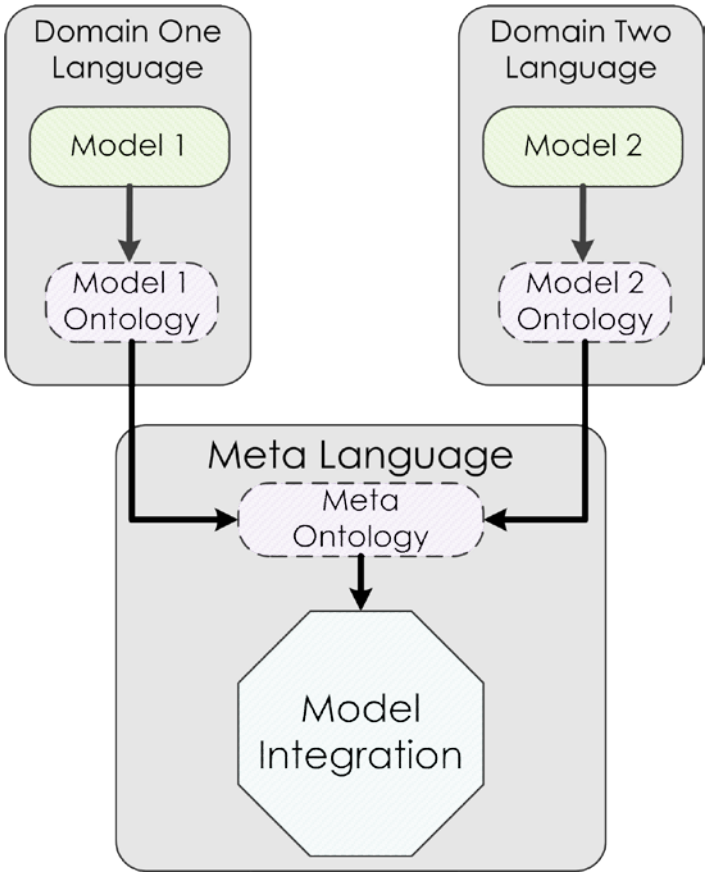

Figure 6.5 Two models from different domains using ontologies to capture the semantics of the models. A meta language is used to produce a meta ontology. The information from the meta ontology is used to perform the syntactic part of the model integration.

This method of translating between ontologies does however mean that all of the component ontologies would need to be produced for each model type, environment, and the mappings captured. This capturing of the information in an ontology would be a considerable increase in resources and investment for any organisation designing a system. This also raises issues with scaling and the effects of many hundreds of thousands of relationships having to be mapped to give a high fidelity, full, system simulation.

6.5.3 Evaluation of Ontologies For This Problem Space

The literature suggests that the prime use of ontologies for model and simulation integration is for the capture of the relations between components that have been identified. It is only relatively recently that this approach has been shown to be at an implementable possibility.

Despite the promise however there are issues with scalability and the resultant computational burden. Many researchers make statements such as “The modelling and simulation community has not taken advantage of the benefits of ontology management technology.” [130]. This technology is not currently able to conduct the complete integration process form end to end. Therefore ontology-based technology would be a component part of the integration process. There is the potential to combine ontological descriptions of the

compoent parts of the simulaiton with service orented architectures [31]. This combination takes one technology that is syntactically strong and another that is semantically strong, combining them to produce a potential solution to the integration problem. This technology hence shows promise if the scalability, and time constraint issues are overcome in the future.

The ontologies are there, the algorithms to integrate them are available, the computational power issues (if trends continue) will become less of an issue as technology progresses, however the issue of getting the data into the ontologies is still a significant issue for any organisation looking to implement them.

6.6 BUSINESS CHALLENGES WITH MODEL AND SIMULATION INTEGRATION

The operation of modelling and simulation within the production of a product or service will only be conducted if the organisation deems that is the most cost effective means of conducting the necessary analysis. This is the one factor that drives all of this modelling and simulation, and as such is to be considered above all else when the organisation is operating within a competitive market environment.

6.6.1 ADOPTION OF NEW TECHNOLOGIES

In the current economic climate it is important for organisations to get new products and services into the market in the shortest possible time. Hence anything that takes time in a project must have intrinsic value to the project. This is a consideration that needs to be made not only relating to any new technologies but also any potential organisational disruptions implementing it may cause. The concept of Integration Readiness Levels (IRL) has been introduced [148] which is an attempt to quantify how ready a sub-system is to be integrated into the wider system. At present such assessment methods are still subjective and often require the judgement of a domain expert. This method of using the scale of IRL is not specifically set up to handle the integration of models and simulations, however the concepts that it encapsulates may be. At present there are limited examples of such metrics being used specifically for models and simulations. It is recognised that the adoption of new technologies is a wide business research area and is not the focus of this research. However the recognition of these issues is necessary for any potential solution to be useable by any organisation.

6.6.2 VENDOR LOCK IN AND RISKS OF ONE SUPPLIER

There are many software companies that can supply a whole suite of tools for many different aspects of the modelling and simulation task. In some instances this can commence from requirements capture all the way through the design, V&V, and even on to deployment. For some organisations this gives advantages, as all of the tools at all the various stages work together. However it does come at considerable risk. If an organisation was to fully buy into a single tool vendor's ecosystem, they face the possibility of: software vendors increasing the cost of their tool above the line of inflation, problems if the software company ceases to exist, and functionality of tools changes between versions. Therefore it is not desirable for any organisation to use tools from only one company. For a discussion of using a range of tools from a multitude of simulation tasks see Section 6.6.4.

For the identified reasons it is common for large organisations to use whole suites of tools from a host of suppliers. This too comes with issues, as there is then no

guarantee that any of the tools will work with each other if any integration or co-simulation is required.

6.6.3 The Possibility To Do More With The Same

From a business standpoint, having invested resources into models it is desirable to re-use them continually. In the literature it is identified that having some means of reusing the models and simulations for future projects has the prospect of reducing the amount of resources that would have to be invested in the new project [14]. However sources which are concerned with the use of such an approach within an organisation rather than a novel approach to model re-use are often cautious with statements such as "… re-use of a sub-system model could be more costly than developing it from scratch." [14] This shows that there are identified issues with model re-use when it comes to using it within an organisation. It is recognised that re-use is as much a cultural concern as it is a technological one [14]. For the re-use of models and simulations to become a reality a decision from the organisation leaders would need to not only move the culture of the organisation but also invest sufficient resources to support such a change.

6.6.4 Commercial-Off-The Shelf

The concept of Commercial-Off-The-Shelf (COTS) was popularised in the 1980s [149]. The basic premise is for product manufacturers to buy parts from commercial vendors. This would allow companies to access skills that they may not have in house or they require a part that uses specialist skills for only a specific part that it would not be financially viable to employ full time engineers to develop. There is also an element of risk that is passed from the customer to the supplier of the COTS component. The effects of using COTS within an organisation are far reaching and largely outside the consideration of this research. However there are aspects of using COTS that directly affect the potential success of model and simulation integration which are discussed. A generalised procedure for a COTS customer to a supplier can be seen below in Figure 6.6.

When utilising COTS as part of a system design the parts that are purchased need to be specified in such a way as to ensure that it behaves as exactly as is intended by the system designers. Once the vendors have produced the specified sub-system many customers then use rigorous testing to ensure that the delivered sub-system operates as is required. If the COTS product is physical, the use of HIL testing is often used in this activity. If the product passes all of the desired tests it is then integrated into the rest of the solution.

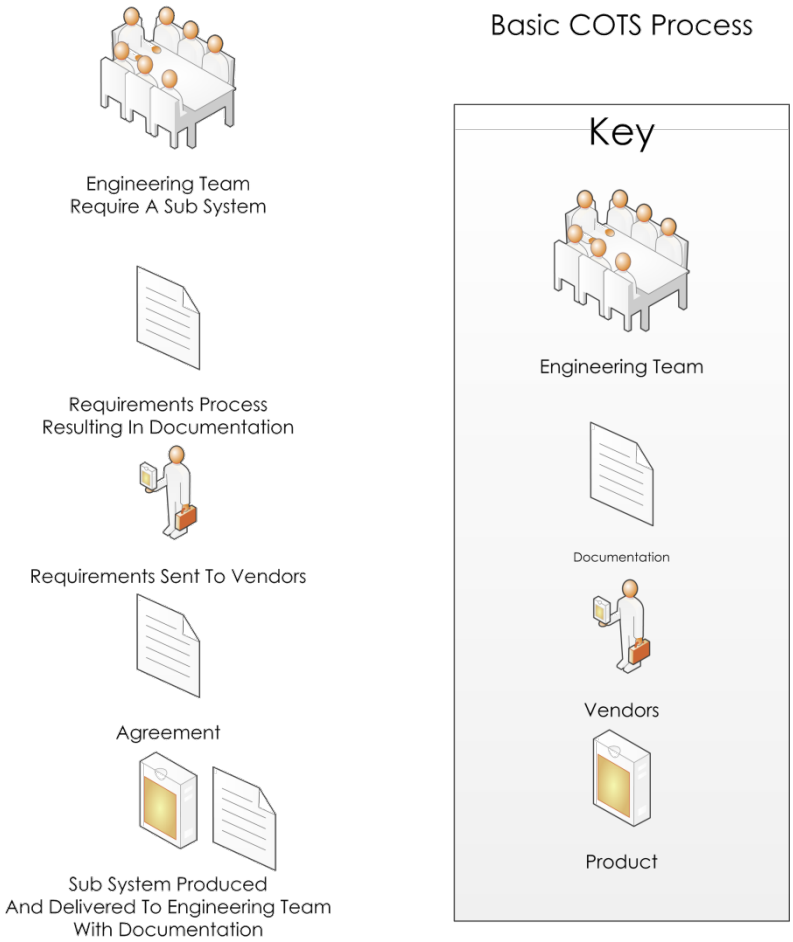

Figure 6.6: A representation of the COTS process time represented top down.

The issue of how to go about obtaining COTS products is a research area in itself. The key steps are; Define, Search, Filter, Evaluate, Analyse, Customise and Implement [150]. They explored many variations on these themes for means of effective COTS selection. It is recognised that in most of the literature COTS is only considered from the perspective of using the products for the first time, not having the components as a result of a previous project. This means that a similar selection process could be applied to the components that an organisation has available internally. This approach shows promise of being applicable to a wider selection processes than just for COTS components.

To enable a COTS sub-system to be usable in the system being designed there requires a detailed understanding of exactly what the COTS sub-system is to be and how it is to operate. This shared understanding is often achieved by using detailed requirements documents combined with dialogue between involved parties. This requirements capture, analysis and inevitable clarification dialogue produces a considerable amount of paperwork.

> *"the degree of subtlety involved in even describing the COTS simulation package interoperability problem can lead to long, lengthy discussions where the parties involved typically finish with no definitive understanding of the problems that must be solved"*[151].

This quote illustrates the extent to which the communication issues between parties can be a timely exercise.

Buying in components in such a manner does mean the full white box understanding of the components and hence the system as a whole may not be possible. This has been identified as a potential integration issue, see section 6.2.9. To mitigate the lack of white box understanding, requirements documents and contracts may state that details as to how the sub-system works in practice have to be supplied. There has been effort invested into finding way to mitigate some of the integration issues between COTS simulations such [151] where they propose the use of Interoperability reference models which rely on the integration problem to fit one of the proposed patterns. To implement such an approach, organisations would need to formulate their own reference models and stick to patterns.

With the need to accurately specify a sub-system and have enough documentation to use it can result in a situation where depending on the organisation the documentation for a COTS sub-system may be far more detailed than components that are made in house.

When using COTS the time and other resources of the organisation buying the items are diverted away from the development of the sub-systems in house. It is intended that only a fraction of what would have been spent in house on development and manufacture is spent on negotiating and purchasing the COTS sub-system. In the literature this potential issue with using COTS within an engineering organisation is not widely discussed.

The re-use of model and simulation COTS products is a topic that has been considered predominately in the defence domain due to lengthy development and life-time of the products. Software obsolescence is less well understood and considerably less mitigated against than hardware obsolescence and is starting to have a real impact on projects [75]. There are a multitude of reasons why software can become obsolete and there is the business decision as when to update to a newer or competitive vendor. The issue of when it is an appropriate time to upgrade software has been discussed [149]. This work is conducted from an IT perspective for tools such as word processors, however it is worth noting that the same sort of issues will be apparent between general office tools and specialist engineering tools. The output of this research revolves around only updating when the tool either cannot, or does not, fulfil the objectives that the tool is being used for, which includes aspects such as vendor support. The benefit of looking at sources such as this is they are combatting similar issues in more

mature fields though in different domains. Obsolescence is often cast aside by many model and simulation researchers as the issues of general model and simulation re-use is still considered to be a considerable challenge without the potential issues that the use of COTS products bring. However due to the prevalence of COTS in industry it cannot be cast aside in this research. The general issues to the re-use of models and simulations are covered in sections 2, and 6.2.9, however there are specific issues to meaningful integration that have to be overcome before obsolescence becomes an issue.

Due to the very nature of COTS the organisation purchasing the models and simulations does not necessarily have a complete white box understanding of how it works. This issue is further complicated if there is no long-term support for the model or simulation after completion of the initial contract. It is also common for any models or simulations to be distributed in a protected format to maintain the intellectual property of the COTS vendor. As shown in section 3.7 information is critical for meaningful integration. If COTS products are to be re-used within integrated modelling and simulation there will need to be guarantees of long term support or greater transparency of solutions.

Applied Dynamics International is a company that formulated a method for using simulation and model specified requirements to build a system up from parts made by various manufacturers [30]. This method involves simulated sub-systems where the individual sub-system simulations could be replaced for hardware with sensors mounted. From Applied Dynamics Internationals marketing material they promote working with large defence and aerospace manufacturers including: Northrop Grumman, BAE Systems, Raytheon, and General Dynamics. These companies are all concerned with the design of highly complex systems. This type of approach has the potential to revolutionise how projects use COTS and also large projects that have many teams working on complex highly electronic-centric projects. Such an approach requires considerable effort in architecture of the solution before the simulation commences. However it is a means by which COTS has been shown to be a viable resource for the development of complex systems.

The use of COTS products in the automotive industry is widespread. It has become so widespread that there have been efforts, such as Autosar [152], to set standards so that automotive and COTS manufacturers have a common means of communicating with OEMS and each other. The core partners involved in this venture include; Bayeischye Motorn Werke AG (BMW Group), Robert Bosch GMbh, Ford Motor Company, Toyota Motor Corporation, General Motors Holding LLC, Continentals AG, Daimler AG, Peugot Citroen Automobiles S.A. and Volkseagen AG. Some consider that these are some of the largest players in the automotive business. As such market leaders are investing in such a joint venture, it demonstrates the extent to which the COTS challenge is causing issues within their organisations. By projecting a unified want the member organisations have a higher bargaining power with the COTS producers. It is worth noting that

modelling and simulation venders are also members of the Autosar organisation. These vendors include but are not limited to: Dassault Systemes, dSPACE GmbH, and MathWorks. This indicates that the simulation tool vendors also recognise that for their products to be used by the automotive companies involved in Autosar, their products need to be compatible.

It is worth commenting that the engineering modelling and simulation tools can in themselves be considered as COTS products. There are companies that will produce model or simulation tools for the study of a specific phenomenon. The process that is used for the purchase of such a tool is much the same as that which has been discussed for the acquisition of a component to a system. The larger tool vendors that sell generic modelling or simulation tools such as Math works and Dassault Systemes can still be considered COTS suppliers, however there products may not be adapted to individual customer's wants or needs.

6.7 AUTOMATION OF ENGINEERING TASKS

With the ever increasing computational power there is the potential to automate some of the engineering tasks that are conducted as part of any engineering project. The technologies that are being explored include: machine learning, Natural Language Processing, automated code generation, and automated documentation. The tasks that have been shown to be possible to automate are deductive in nature. They do not require any imagination or additional information to make the required decisions. Only deductive tasks have been automated due to a limitation of the computational systems that are currently used.

It has been observed that an area of engineering that could be automated is that of requirements handling. There are COTS tools available that can aid in the capture and management of requirements, but as of yet little attention has been given specifically to model and simulation requirements analysis. The technology is available and the problem space is understood significantly well to allow for such a tool to be developed, the potential for which was demonstrated in section 5.

There are current limitations such as computational power and company structure that restrict the full capability of current methods to allow for the level of potential automation. There is also the ethical question of whether we should look to automate the engineering process at all or whether we should keep humans in the loop to ensure that there is accountability for actions. This ethical question also extends further to the issues surrounding 100% virtual prototyping discussed in section 6.2.7.

6.8 Potential Paradigm Shift Brought About by Model Integration

With the introduction of a way to identify meaningful integration from a sea of potentially erroneous, meaningless, integration possibilities comes a strategy that could change the very way in which engineers design systems. By having a reliable means of assuring integration, is the possibility to not only model one sub-system or part thereof but rather simulate the system as a whole allowing for previously unconceived interactions to be identified. Such interactions and behaviours could be harnessed and used to better achieve the desired behaviour of the system being designed. This approach has the potential to change engineering tests from being verification that a design has the potential to work, to an investigation into the phenomena that engineers are harnessing to produce a desired output. This is essentially using the integrated simulation as a research tool as well as a design tool. If such observations were then validated through the model validation processes proposed in section 3.5.3 the organisation conducting the test could accrue more information and indeed further their own understanding about the domain that they are working within, potentially increasing the companies' competitive advantage.

For such a change in the paradigm there would need to be consideration as to not only how the knowledge that is gained through the testing is to be captured and stored, but also how the findings feed back into the design process. If an organisation distributes work in teams based on sub-systems then the knowledge gained about how these sub-systems interact could be lost or ignored. It would require a process and organisation with a holistic nature to be capable of making use of the system wide observations, which is far from a trivial task. The additional information that can be gained from integrating sub-systems can be seen in Figure 6.7 below.

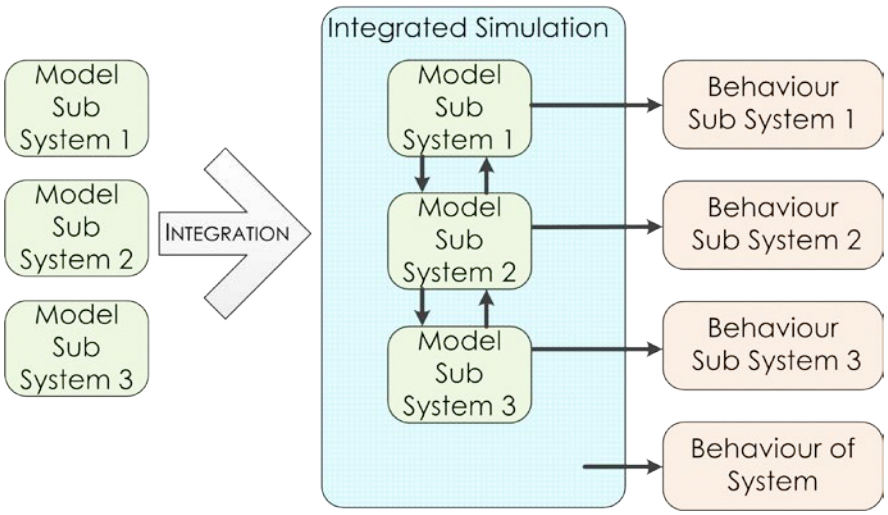

Figure 6.7: Outputs of an integrated simulation are more than the output of the individual components as the behaviour of the system as whole is also captured.

The issue of how to best use an integrated simulation is further complicated when variation of design is considered. If a design is formulated, models are generated for the sub-systems; the sub-system models are integrated for a full system test, and the output of the integrated test requires a change in design. If the change in design is then to be evaluated using some of the existing simulation components there can be issues with ensuring that the semantics of the models remain suitably similar for the outputs to remain meaningful. Hence the process surrounding the use of integrated simulations requires as much thought and planning before the project starts as the way in which the integration of the models is to be conducted.

The impact of full system testing using many integrated mathematical models is still a new concept that has the potential to change the way in which large engineering project are currently conducted.

6.9 SUMMARY

The deeper philosophical challenges to model and simulation integration have been discussed in this chapter. The challenges to integration have been found to stem from the very way in which the engineer's traditional reductionist approach to engineering problems can cause issues. Even when the models and simulations have been created, if they have not been validated against the phenomenon that they are mimicking, there is a question as to how useful the model or simulation can be. In an effort to move the final design closer to the simulation, HIL and HITL are used. The effects and unique challenges of using such methods in modelling and simulation are discussed.

When a system is created there are situations where those who are creating and financing its creation are different. These differences can result in the focus and hence the product not being that of those who are going to be using it. This is a deeper reason for models and simulations to focus on specific issues resulting in testing that may not be appropriate for what is needed for a successful product.

With modelling and simulation improving to the extent that virtual prototyping can now be of real use to an engineering team, questions are being asked as to whether we will soon be at a stage of 100% virtual testing, meaning no use of prototypes. There are fundamental issues with the differences between a model or simulation and the real world phenomena. These issues are discussed and found to bring into question the possibility of 100% virtual testing for physical products being a real prospect. This means there may well be the need for prototypes for testing before the product is put into production for many years to come.

There are questions as to where time is spent on a project as all time comes at a cost. The impact of spending time on systems engineering approaches rather than traditional engineering methods has been discussed. The result from this questioning has resulted in the opinion that it is dependent on the problem that they are being applied to. If the problem situation and solution are simple then it may be cheaper to use traditional methods, however in the inverse situation where the situation and solution is complex the converse is advised and systems engineering principles used.

The effects of computational technology have a real impact on the types of modelling techniques that can be reasonably implemented. The increase in computational power has resulted in previously discarded methods now being used. The computational technology is not limited to processing power on one machine. With the current research into computer networks and cloud computing there is the real prospect of having more computational power available for engineering project than ever before. How to utilise this computational power is an ongoing research question. With this computational power also come more data than ever before. How to best store this information

was discussed and ontologies have been repeatedly identified in the literature as being the best way of capturing such information. The suitability and feasibility of this technology use in the identified problem space was seen as being a real challenge without significant investment.

When implementing model and simulation integration there are business challenges that have been identified that need to be overcome for it to make a positive impact in practice. To reduce costs, automation is often used. However within engineering; there are aspects that require inductive rather than deductive reasoning, a capability currently outside the computational science discipline. It has been identified that to make best use of model and simulation integration as part of the engineering processes it may require a fundamental paradigm shift within engineering organisations.

7 CONCLUSION

7.1 CONCLUSION

The conclusions drawn from the work reported are far from what was envisaged at the start of this research project. The intended direction of the work was to develop new systems methods that incorporated modelling and simulation and apply these to real world automotive case studies where existing methods were found wanting. However due to the real world nature of the work the lack of available documented models and simulations led to the choice of two very different case studies. The first a materials behavioural product design (training squash ball), and the second was an open source automotive example that emulated issues with commercially sensitive material. However it is proposed that the case studies are still suitable to test the proposed methods, serve as a reference point to demonstrate how the processes can aid in the identification of issues before the execution of the final integration, are suitably complex, and satisfy the study aim and objectives (section 4).

The aim was formulated from the understanding of the problem and was verified (section 1.5). In order to provide evidence in support of the general aim, a set of testable objectives were formed. By satisfying the objectives the aim was also satisfied, hence to review this research the objectives are to be used as a reference point for verification of this work as a whole.

Objective A: Determine the potential of automatic identification of model and simulation dependencies, and the respective assumptions encapsulated.

> Before the identification of dependencies and assumptions can be automated the information that is required has to be known. Section 3.7 highlighted the information needed for integration of two or more models or simulations. This information covers their dependencies and assumptions. A means of capturing this information was proposed in the form of machine readable tables. The automation of mining and comparison of the identified required data (in section 3.7) was investigated using NLP in section 5. It was found that the NLP technology shows real potential for aiding the automation of the integration of existing model and simulation components.

Objective B: Discern the means by which levels of abstraction can be identified and the impact on the task of integration established.

> The effects on the integration task of having candidate models and simulations with differing levels of abstraction were documented in the literature as representing a significant challenge (section 2.4). There are currently many different methods for modelling and simulation which operate at different levels of abstraction. These methods each bring variation to the integration task (section 2.5).

A novel method was developed to capture the levels of abstraction (section 3.8). The method uses the term *abstraction* as a concept by which two or more models can be compared against each other and an assessment of level of abstraction made. It was found that this method could be of use when models or simulations are representing the same phenomena. However the question of comparing abstraction across models and simulations of differing phenomena remains an open research question.

Whenever a model or simulation is implemented a level of abstraction is used consciously or otherwise by the modeller. Conceptually, this affects the integration task by rendering it either meaningful or meaningless (section 6.3.2). This goes deeper as model abstraction is the result of what is known about the subject of the model or simulation. Knowns and unknowns have been shown to directly affect the validation of any integration (section 6.2.4).

Objective C: Develop a means by which the generation and creation of middleware can be automated.

The use of middleware is established for integrating existing models and simulations, however there are only limited examples of where automation is used - largely due to the complexity of the problem (section 2.3.1). Due to the flexibility of middleware and its ability to overcome the challenges of data exchange and conversion, it was deemed to be of potential use in this problem space (section 2.3.2). The wide range of middleware was discussed to allow meaningful discussion (section 2.4.3). The concept of middleware has been extended to situations where models and simulations can be distributed and run across a network. A method that has been shown to work in such situations is that of federated systems, which takes middleware and incorporates orchestration functionality with web services to enable distributed computation (section 2.4.4).

To automate a process it first needs to be assessed to see if it can be developed as a solution for the task. A process for assisting in the integration of models and simulations was proposed in section 3.5. The generation of middleware was directly expressed as part of the SEIS process section in the SESEMS and *defining the gaps* sub process sections 3.5.5 through 3.5.8.

Part of the proposed process entails the capture of the minimum required information needed for an informed decision regarding whether a specific integration is meaningful or not (section 3.7.3). Consideration was given to machine readability of the captured information, allowing for the possibility of automation. However it may not be possible to automate some processes due to the nature of the decision making task (section

5.1.2). This was found to be the case for the proposed method for defining levels of abstraction, as it required inductive thinking section 3.8. The potential to automate the selection of existing models using requirements for a desired simulation as well as the mining of data for the proposed integration tables was investigated using a POC NLP application. It was found that there is potential for this technology to be of assistance in this problem space (section 5).

Objective D: Validate the designed process with a case study using real world models.

The proposed methods were validated using two case studies. The first case study took the form of a full end-to-end example of the proposed methods. This example used the development of a squash ball as a platform (section 4.2) and demonstrated the feasibility of using the proposed methods and illustrated the verification process. The second case study takes the form of an open source version of an automotive domain application (section 4.3). This case study mimicked that which was found with industrial examples. Using the proposed methods a non-domain expert is capable of finding inconsistences that would have rendered the integration meaningless.

As has been shown, each of the objectives has evidence to support the conclusions drawn and hence the aim has also been met.

From the work that has been conducted throughout this project it has been demonstrated that the methods of systems engineering are as applicable to the formulation of virtual testing of a system as they are to the creation of a systems being designed.

If virtual experimentation exceeds the size and complexity that a small team can manage, then application of the systems engineering principles are of significant benefit. Crucially, with the investment in time that systems engineering methods require, they give a solid reference point that allows for meaningful verification and later validation to take place. This also allows for models and simulation to be compared in a meaningful way at a later date.

With the current modelling and simulation practices such as they are, the design of a virtual test of a designed product requires considerable work. It has been argued that the resultant simulation can be a complicated or even complex system in its own right. The recognition that the simulations used are systems in their own right also then leads onto the fact the simulations themselves can also require designing, hence similar systems engineering methods are applicable.

The integration of models and simulations is far easier and the results potentially more valid if the component models are constructed with the specific test and integration from their initial specification. Attempting to integrate existing models

and simulations into a single test may be more time consuming and the results less valid than creating the whole experiment in a bespoke manner.

The documentation of models and simulations is critical if the models are to be used at a later date. If there is no documentation to accompany a model then it becomes next to worthless once those who were involved in in its creation are no longer available. Without documentation much of the automated integration that could otherwise be conducted is made far more difficult or even impossible. The information required for meaningful integration at a later date is captured in section 3.7. It is hence recommended that even if the tables are not used that the documentation includes as a minimum the information identified in these tables.

It has been shown that for any integration to be meaningful the semantics of the component models must be aligned not only with each other but also the experiment that is to be conducted. It is not sufficient to just handle the communication between the component parts for an integration to be meaningful.

To capitalise on the value intrinsically held by existing models and simulations they need to be in a location where engineers can readily access them. It is hence recommended that existing models and simulations are stored in a repository alongside their documentation. With the repository being such that engineers form across the organisation can at least know of the existence of the available models and simulations even if there access is limited.

7.2 CONTRIBUTION TO KNOWLEDGE

The application and refinement of systems engineering principles to the integration of existing models and simulations has resulted in contributions having been made in both process and methodology.

A new methodological representation of systems thinking has been created to depict the way in which it can be implemented in practice within an organisation. This representation can be found in section 3.2 and the application of it in section 3.5. This representation is innovative as it attempts to map the linear processes found in industry with concurrent parallelism that is often found within systems thinking. This representation, unlike many others, prioritises the understanding of the problem before attempts are made to specify requirements. It also clearly depicts how each stage builds on the work before. It is, in effect, a way of applying systems thinking without over-specifying each aspect of the project, allowing for the most appropriate method to be applied as and when needed. This is an attempt to work with the existing industrial methods rather than replacing them.

The information that is required for meaningful integration of existing models and simulations has been developed (see section 3.7). The specified information allows for individuals that were not involved with the creation of specific models or simulations to still make a valid judgment as to whether two or more potential components can be meaningfully integrated. This builds on work conducted with standards, however critically it also focuses around the semantics rather than just the inputs and outputs.

From the literature it was identified that levels of abstraction were repeatedly used as reason for not having a meaningful model or simulation integration (section 2). However there are few examples of where a means of specifying levels of abstraction has been found and used. The method proposed in section 3.8 used a means of looking at what is conceptually represented by the variables and parameters used in the mathematical representations.

The issue of verifying if the integration of two or more models or simulations and assessing if it is meaningful or not is one that has little coverage in the literature. The factors that cause the issue as to verifying whether a specific model and simulation integration is meaningful or not is what to use as a reference point. It is proposed in section 3.2 that the verification is not just a monolithic task to be conducted at the end once all integration work has been conducted, but rather can be broken down into smaller tasks, with different references dependent on the specific situation. A means of validating the integration is presented (section 3.5) and its use discussed (section 6.2.3).

It was identified that there is currently a dislocation between the virtual modelling simulation and testing with the rest of the classical engineering process (section 6.8). A process in section 3.5 proposed a means of aligning this incongruence.

However it is acknowledged that more work needs to be conducted to find methods to investigate this difference further.

The identification of requirements for a model and simulation being different from the requirements of the system being emulated is a crucial one. Not only do the requirements act as a reference point for verification but also the operational environment is different and hence requires altered attributes. This resulted in a model or simulation requirements writing guide, it details factors that have to be taken into consideration when implementing a virtual test of a potential design (section 3.6).

Using the contributions in methodology a complete end-to-end process is proposed in section 3.5 for virtual simulation and testing of a potential design. The process encapsulates: the linearised systems engineering approach, the simulation requirements guide, integration tables, and NLP, to produce an end-to-end guide for engineers who are tasked with the virtual simulation and test of a potential design using existing components. The process was verified in section 4 with a product development study (section 4.2) as well as an emulated industrial automotive example (section 4.3). The process was found to be of use and especially so for non-domain experts tasked with the integration of existing models and simulations. A key finding from this NLP based work is that simple noun-verb phrases are not sufficient to identify requirements or the description of capabilities that fulfil said requirements.

7.3 FUTURE WORK

This work has brought to light many opportunities for future investigation and study. The case studies (section 4) demonstrated that the proposed processes work and show promise. However it would be beneficial to widen the usage and develop the processes further with examples from other domains. The processes, once demonstrated in multiple instances, could then have a phased in adoption by an organisation and a cultural review of its acceptance studied. Such work would strengthen the processes and potentially reduce the time to market of complex engineering projects.

The processes proposed in section 3 could be developed into a piece of software in its own right, guiding the user through the integration process and operating as a backbone that other forms of automation could be bound to. This has the potential to bridge the gaps that were identified in the literature (section 2) and provide an end-to-end service for virtual testing.

The requirements writing guide for modelling and simulation could have a greater impact if it was developed into a standard. Converting the guide into a standard would enable undisputable analysis as to whether a model or simulation abides by the concepts set forth in the guide. This would then allow for more meaningful discussion to be had by groups of individuals using the standard regarding how to proceed with virtual testing. Converting this guide to a standard would also allow for the semantic side of integration to be captured without significant additional effort being invested into the initial modelling phase of virtual testing.

Once a significant number of successful validated integrations have been conducted using the proposed methods it would be an ideal opportunity to investigate technologies to support a repository that could be interrogated. Use of ontologies would be useful here (section 6.5). Such a repository of existing models and simulations has the potential to reduce the time that it takes for similar integrations at a later date to be set up and tested. With such a network of known integrations, analysis regarding their relationships could be drawn out and potential patterns identified, further reducing the time taken to conduct meaningful model and simulation integration and thus time to market.

A novel method of defining abstraction has been proposed and tested in section 3.8. The findings of the testing were promising, showing that such a concept can be implemented in a meaningfully way. It is therefore a logical progression to postulate if other similar methods could be investigated to capture terms such as model or simulation '*fidelity*'. Such methods could potentially be used to capture further information regarding the semantic similarity between existing models and simulations.

The work that was conducted regarding the use of NLP for process support of the selection of existing models to produce an integrated simulation is just the tip of

the iceberg (section 5). The POC demonstrated the potential for this technology to be of use in this domain and it warrants further development, and the formulation into a commercially usable industrial standard tool. The development of this, it is perceived, would entail the capture of multiple domain specific words and phrases that could be used to better identify the information needed for the integration tables (section 3.7), as well as the development of statistical style NLP techniques to ascertain the critical assumptions that are used during the creation of models and simulations. There is also no reason why these methods could not be expanded to the analysis of model based systems engineering tools such as UML and SysML as the requirements reference point, allowing for such a tool to be applicable across a greater variety of engineering projects.

The systems lifecycle discussed in section 3.5.1 and how to make use of the ever expanding capabilities of computation modelling and simulation is an ever changing research question. The possibility of virtual prototyping is now real, though how to best use these capabilities is still a matter of contention. There are still questions of how to best implement modelling and simulation as part of the overall systems lifecycle. Such questions are; when to start, what to model, what to simulate, and the extent to which to virtually prototype, are all issues that are not widely discussed in the literature. Hence it is proposed that it would be worthwhile specifically investigating such questions using real world applications to find more effective ways of using this new technology as part of the product development lifecycle.

There is the issue of how to structure an organisation, such that when a complete complex design is simulated, there can be ownership of the behaviours that were not apparent in the individual sub-systems testing. As such, oversight would need to exhibit influence over all sub-systems behaviour, as well as demonstrating a working understanding all interactions between the components. This gives rise to the concern that our abilities for simulation are verging on exceeding our ability to fully make use of them within an organisational setting.

8 REFERENCES

[1] J. Mossinger, “Software in automotive systems,” *IEEE Software*, vol. 27, no. 2, pp. 92–94, 2010.

[2] P. Checkland, “Soft systems methodology: a thirty year retrospective,” *Systems Research and Behavioral Science*, vol. 17, no. S1, pp. S11–S58, 2000.

[3] S. Beer, *The viable system model: interpretations and applications of Stafford Beer’s VSM*. Chichester: John Wiley & Sons, 1989.

[4] I. Lorscheid, B.-O. Heine, and M. Meyer, “Opening the ‘Black Box’ of simulations: increased transparency and effective communication through the systematic design of experiments,” *Computational and Mathematical Organization Theory*, vol. 18, no. 1, pp. 22–62, 2012.

[5] L. Ilzarbe, M. J. Alvarez, E. Viles, and M. Tanco, “Practical applications of design of experiments in the field of engineering: a bibliographical review,” *Quality and Reliability Engineering International*, vol. 24, no. 19th Febuary, pp. 417–428, 2008.

[6] J. Asprion, O. Chinellato, and L. Guzzella, “A fast and accurate physics-based model for the NOx emissions of diesel engines,” *Applied Energy*, vol. 103, pp. 221–233, Mar. 2013.

[7] A. Mouzakitis, D. Copp, R. Parker, and K. Burnham, “Hardware-in-the-loop system for testing automotive ECU diagnostic software,” *Measurement and control*, vol. 42, no. 8, pp. 238–245, 2009.

[8] D. Di Ruscio, “Isight automate design exploration and optimization.” Dassault Systemems, pp. 1–6, 2014.

[9] A. C. Ahn, M. Tewari, C.-S. Poon, and R. S. Phillips, “The limits of reductionism in medicine: could systems biology offer an alternative?,” *PLoS medicine*, vol. 3, no. 6, pp. 709–713, 2006.

[10] F. el Khaldi, C. Ahouangonou, M. Niess, and O. David, “Cloud based HPC for innovative virtual prototyping methodology: automotive applications,” *Transportation Research Procedia*, vol. 14, no. Transport Research Arena TRA2016, pp. 993–1002, 2016.

[11] “ANSYS,” *ANSYS website*, 2016. [Online]. Available: http://www.ansys.com/en-GB/?gclid=Cj0KEQjwwry8BRDjsbjMpPSDvagBEiQA5oW0nlWgvGZfXJB0TAG9xgj9nJfljtr5QWbnckxGVBLpPFIaAvd28P8HAQ. [Accessed: 15-Mar-2017].

[12] B. Hutchinson, “Pump simulation advances with ANSYS 17.0,” *ANSYS Blog*, 2016. [Online]. Available: http://www.ansys-blog.com/pump-simulation-advances-ansys-17-0/. [Accessed: 15-Mar-2017].

[13] Flowmaster Ltd, “Flowmaster case study automotive,” *Mentor Graphics publication material*. Flowmaster Ltd, pp. 1–5, Apr-2007.

[14] S. Robinson, R. E. Nance, R. J. Paul, M. Pidd, and S. J. E. Taylor, “Simulation model reuse: definitions, benefits and obstacles,” *Simulation Modelling Practice and Theory*, vol. 12, pp. 479–494, 2004.

[15] R. G. Bartholet, D. C. Brogan, and P. F. Reynolds, "The computational complexity of component selection in simulation reuse," *2005 Winter Simulation Conference*, Buena Vista, USA, 2005, pp. 2472–2481.

[16] D. McKenzie, S. M. O'Neill, N. K. Larkin, and R. A. Norheim, "Integrating models to predict regional haze from wildland fire," *Ecological Modelling*, vol. 199, no. 3, pp. 278–288, Dec. 2006.

[17] H. A. H. Handley, "Incorporating the NATO human view in the DoDAF 2 . 0 meta model," *Systems Engineering*, vol. 15, no. 1, pp. 108–117, 2012.

[18] S. B. Engineering, "Enginsoft newsletter simulation based engineering & sciences," *Enginsoft Newsletter*. Enginsoft, pp. 1–72, Jan-2014.

[19] M. U. Awais, P. Palensky, W. Mueller, E. Widl, and A. Elsheikh, "Distributed hybrid simulation using the HLA and the functional mock-up interface," in *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, Vienna, Austria, 2013, pp. 7564–7569.

[20] A. Ledeczi, J. Davis, S. Neema, and A. Agrawal, "Modeling methodology for integrated simulation of embedded systems," *ACM Transactions on Modeling and Computer Simulation*, vol. 13, no. 1, pp. 82–103, Jan. 2003.

[21] "The NIST validator," *NIST Website*, 2015. [Online]. Available: http://validator.omg.org/se-interop/changelog?hunchentoot-session=9%3AC224B632FA49F33EA800D6B342B99A54. [Accessed: 15-Mar-2017].

[22] Modelica Association Project "FMI," "Functional mock-up interface for model exchange and co-simulation." Modelica Association Project "FMI" Functional, pp. 1–120, 2014.

[23] "What is an S-function," *MathWorks help file*, 2016. [Online]. Available: http://uk.mathworks.com/help/simulink/sfg/what-is-an-s-function.html?s_tid=gn_loc_drop. [Accessed: 15-Mar-2017].

[24] "National Instruments TestStand," *National Instruments*, 2016. [Online]. Available: http://www.ni.com/teststand/. [Accessed: 15-Mar-2017].

[25] ISO, "ISO standards," *ISO website*, 2016. [Online]. Available: http://www.iso.org/iso/home/standards.htm. [Accessed: 15-Mar-2017].

[26] "FMI support in tools," *FMI Standard organisation*, 2016. [Online]. Available: https://www.fmi-standard.org/tools. [Accessed: 15-Mar-2016].

[27] I. Standard, "Systems and software engineering - life cycle processes - requirments engineering." ISO/IEC/IEE, pp. 1–83, 2011.

[28] G. A. Lewis, E. Morris, S. Simanta, and L. Wrage, "Why standards are not enough to guarantee end-to-end interoperability," in *7th International Conference on Composition-Based Software Systems*, Madrid, Spain, 2008, pp. 164–173.

[29] P. Welch, "Programming Sucks," *Still Drinking (Web blog)*, 2014. [Online]. Available: http://www.stilldrinking.org/programming-sucks. [Accessed: 15-

Mar-2017].

[30] A. Dynamics, “Understanding advanced system integration labs: simulation-centric system integration.” Applied Synamics International, pp. 1–12, 2007.

[31] J. Touzi, F. Benaben, H. Pingaud, and J. P. Lorré, “A model-driven approach for collaborative service-oriented architecture design,” *International Journal of Production Economics*, vol. 121, no. 1, pp. 5–20, Sep. 2009.

[32] M. a. Jaeger, H. Parzyjegla, G. Mühl, and K. Herrmann, “Self-organizing broker topologies for publish/subscribe systems,” in *Proceedings of the 2007 ACM symposium on Applied computing - SAC New York, New York, USA*, 2007, pp. 543–550.

[33] J. M. Lopez-Vega, J. Povedano-Molina, G. Pardo-Castellote, and J. M. Lopez-Soler, “A content-aware bridging service for publish/subscribe environments,” *Journal of Systems and Software*, vol. 86, no. 1, pp. 108–124, Jan. 2013.

[34] L. Jalali, S. Mehrotra, and N. Venkatasubramanian, “Formal Modeling: Actors; Open Systems, Biological Systems,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, G. Agha, O. Danvy, and J. Meseguer, Eds. Berlin: Springer Berlin Heidelberg, 2011, pp. 352–367.

[35] W. Kang, K. Kapitanova, and S. H. Son, “RDDS: A real-time data distribution service for cyber-physical systems,” *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 393–405, 2012.

[36] H. -a. Jacobsen, “Tutorial: OMG data distribution service,” in *23rd International Conference on Distributed Computing Systems Workshops, Québec, Canada, 2003. Proceedings.*, 2003, pp. 198–198.

[37] A. Hakiri, P. Berthou, and T. Gayraud, “Addressing the challenge of distributed interactive simulation with data distribution service,” *CoRR*, vol. abs/1008.3, pp. 1–9, 2010.

[38] F. Safi Esfahani, M. A. Azmi Murad, M. N. B. Sulaiman, and N. I. Udzir, “Adaptable decentralized service oriented architecture,” *Journal of Systems and Software*, vol. 84, no. 10, pp. 1591–1617, Oct. 2011.

[39] M. Huhns and M. P. Singh, “Service-oriented computing: Key concepts and principles,” *IEEE Internet Computing*, vol. 9, no. 1, pp. 75–81, 2005.

[40] D. E. Martin, P. A. Wilsey, R. J. Hoekstra, E. R. Keiter, S. A. Hutchinson, T. V Russo, and L. J. Waters, “Integrating multiple parallel simulation engines for mixed-technology parallel simulation,” in *Simulation Symposium, 35th Annual*, San Deigo, USA, 2002, no. 7888, pp. 45–52.

[41] Y. Huang, R. McMurran, G. Dhadyalla, R. P. Jones, and A. Mouzakitis, “Model-based testing of a vehicle instrument cluster for design validation using machine vision,” *Measurement Science and Technology*, vol. 20, no. 6, pp. 1–11, Jun. 2009.

[42] M. López-Sanz, C. J. Acuña, C. E. Cuesta, and E. Marcos, “Modelling of service-oriented architectures with UML,” *Electronic Notes in Theoretical Computer Science*, vol. 194, no. 4, pp. 23–37, Apr. 2008.

[43] M. W. Beall, “An object-oriented framework for the reliable automated solution of problems in mathematical physics,” Rensselaer Polytechnic Institute, 1999.

[44] J. Mihm, C. Loch, and A. Huchzermeier, “Problem-solving oscillations in complex engineering projects,” *Management Science*, vol. 49, no. 6, pp. 733–750, 2003.

[45] A. Bharambe, S. Rao, and S. Seshan, “Mercury: a scalable publish-subscribe system for internet games,” in *1st Workshop on Network and Systems Support for Games (NetGames ’02)*, 2002, pp. 3–9.

[46] D. Gregorczyk, “WS-Eventing SOAP-over-UDP multicast extension,” in *2011 IEEE International Conference on Web Services*, Washington, USA, 2011, pp. 660–665.

[47] R. Kewley, J. Cook, N. Goerger, D. Henderson, and E. Teague, “Federated simulations for systems of systems integration,” in *2008 Winter Simulation Conference*, Miami, USA, 2008, pp. 1121–1129.

[48] W. Cai, Z. Yuan, M. Y. H. Low, and S. J. Turner, “Federate migration in HLA-based simulation,” *Future Generation Computer Systems*, vol. 21, no. 1, pp. 87–95, Jan. 2005.

[49] J. M. Schlesselman, G. Pardo-Castellote, and B. Farabaugh, “OMG Data-Distribution Service (DDS): architectural update,” in *IEEE MILCOM 2004. Military Communications Conference, Orlando, USA,* 2004, vol. 2, pp. 961–967.

[50] A. Hakiri, P. Berthou, A. Gokhale, D. C. Schmidt, and T. Gayraud, “Supporting end-to-end quality of service properties in OMG data distribution service publish/subscribe middleware over wide area networks,” *Journal of Systems and Software*, vol. 86, no. 10, pp. 2574–2593, Oct. 2013.

[51] R. Joshi and G. Castellote, “A comparison and mapping of data distribution service and high-level architecture,” *RTI Self-publication*. Real-Time Innovations, pp. 1–9, 2006.

[52] G. Pardo-Castellote, “OMG data-distribution service: architectural overview,” in *23rd International Conference on Distributed Computing Systems Workshops, Rhode Island, USA,* 2003, pp. 200–206.

[53] F. Campolongo, J. Cariboni, and A. Saltelli, “An effective screening design for sensitivity analysis of large models,” *Environmental Modelling and Software*, vol. 22, no. 10, pp. 1509–1518, 2007.

[54] P. Bauer and P. J. Van Duijsen, “Challenges and advances in simulation,” in *Power Electronics Specialists Conference, Recife, Brazil, 2005. PESC ’05. IEEE 36th*, 2005, pp. 1030–1036.

[55] A. H. Nayfeh, M. I. Younis, and E. M. Abdel-Rahman, “Reduced-order

models for MEMS applications," *Nonlinear Dynamics*, vol. 41, no. 1–3, pp. 211–236, 2005.

[56] R. E. Shannon, R. Mayer, and H. H. Adelsberger, "Expert systems and simulation," *Simulation*, vol. 44:6, pp. 275–284, 1985.

[57] J. He, D. a Crolla, M. C. Levesley, and W. J. Manning, "Coordination of active steering, driveline, and braking for integrated vehicle dynamics control," *Proceedings of the Institution of Mechanical Engineers Part D Journal of Automobile Engineering*, vol. 220, no. 10, pp. 1401–1420, 2006.

[58] J.-F. Castet and J. H. Saleh, "Satellite reliability: statistical data analysis and modeling," *Journal of Spacecraft and Rockets*, vol. 46, no. 5, pp. 1065–1076, 2009.

[59] J. Eason and S. Cremaschi, "Adaptive sequential sampling for surrogate model generation with artificial neural networks," *Computers and Chemical Engineering*, vol. 68, pp. 220–232, 2014.

[60] B. Andersson, R. Andersson, L. Håkansson, M. Mortensen, R. Sudiyo, and B. van Wachem, *Computational fluid dynamics for engineers*. Cambridge, New York: Cambridge University Press, 2012.

[61] P. Sutton, "The application of multi-attribute optimisation as a systems engineering tool in an automotive CAE environment," Unversity of Bath, 2012.

[62] F. Liu, M. P. F. Sutcliffe, and W. R. Graham, "Prediction of tread block forces for a free-rolling tyre in contact with a rough road," *Wear*, vol. 282–283, pp. 1–11, Apr. 2012.

[63] H. Neema, J. Gohl, Z. Lattmann, J. Sztipanovits, G. Karsai, S. Neema, T. Bapty, J. Batteh, H. Tummescheit, and C. Sureshkumar, "Model-based integration platform for FMI co-simulation and heterogeneous simulations of cyber-physical systems," in *Proceedings of the 10th International Modelica Conference, Lund, Sweden,* 2014, pp. 235–245.

[64] "Type conversions and type safety (modern C++)," *Microsoft Developer Network*, 2016. [Online]. Available: https://msdn.microsoft.com/en-us/library/hh279667.aspx. [Accessed: 07-Jul-2016].

[65] B. Falkenhainer and K. D. Forbus, "Cmpositional modeling: finding the right model for the job," *Artificial Intelligence*, vol. 51, no. 1–3, pp. 95–143, 1991.

[66] O. El-Gayar and A. Deokar, "A semantic service-oriented architecture for distributed model management systems," *Decision Support Systems*, vol. 55, no. 1, pp. 374–384, Apr. 2013.

[67] "ISO 15288 System life cycle processes rev 2013 CD." ISO/IEC/IEE, 2013.

[68] International Standard, "International standard ISO 15926-2 insustrial automation systems and integration - integration of life-cycle data for process plants including oil and gas production facilities," vol. 2003. ISO, pp. 1–241, 2003.

[69] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, E. Constantinescu, D. Estep, K. Evans, C. Farhat, A. Hakim, G. Hammond, G. Hansen, J. Hill, T. Isaac, X. Jiao, K. Jordan, D. Kaushik, E. Kaxiras, A. Koniges, K. Lee, A. Lott, Q. Lu, J. Magerlein, R. Maxwell, M. McCourt, M. Mehl, R. Pawlowski, a. P. Randles, D. Reynolds, B. Riviere, U. Rude, T. Scheibe, J. Shadid, B. Sheehan, M. Shephard, A. Siegel, B. Smith, X. Tang, C. Wilson, and B. Wohlmuth, "Multiphysics simulations: challenges and opportunities," *International Journal of High Performance Computing Applications*, vol. 27, no. 1, pp. 4–83, Feb. 2013.

[70] M. Horstemeyer, "Multiscale Modeling : A Review," in *Practical Aspects of Computational Chemistry*, First., J. Leszczynski and M. K. Shukla, Eds. Springer Netherlands, 2009, pp. 87–135.

[71] J. W. Fenner, B. Brook, G. Clapworthy, P. V Coveney, V. Feipel, H. Gregersen, D. R. Hose, P. Kohl, P. Lawford, K. M. McCormack, D. Pinney, S. R. Thomas, S. Van Sint Jan, S. Waters, and M. Viceconti, "The EuroPhysiome, STEP and a roadmap for the virtual physiological human.," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 366, no. 1878, pp. 2979–99, Sep. 2008.

[72] E. Weinan, *Principles of Multiscale Modeling*. Cambridge University Press, 2011.

[73] A. Pennycook, "English language and linguistics language and linguistics : disinventing standard English disinventing standard English," *English Language and Linguistics*, vol. 4, no. 1, pp. 115–124, 2014.

[74] P. de Haan, "Review article: The Cambridge grammar of the English language," *English Studies*, vol. 86, no. 4, pp. 335–341, May 2005.

[75] L. Merola, "The COTS software obsolescence threat," *Proceedings - Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems*, Orlando, USA, 2006, pp. 127–133, 2006.

[76] INCOSE, "INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Acticities," in *Systems engineering handbook a guide for system life cycle processes and acticities*, 4th Ed., D. D. Walden, G. J. Roedler, K. J. Forsberg, D. R. Hamelin, and T. M. Shortell, Eds. Hoboken: John Wiley & Sons, 2015, pp. 33–36.

[77] M. W. Maier and E. Rechtin, "The Art Of Systems Architecting," in *The Art of Systems Architecting*, 2nd Ed., CRC Press LLC, Ed. Florida: CRC Press LLC, 2000, pp. 21–24.

[78] J. A. Jacquez, "Design of Experiments," *The Franklin Institute*, vol. 335B, no. 2, pp. 259–279, 1998.

[79] R. A. Fisher, "The Influence of Rainfall on the Yield of Wheat at Rothamsted," *Philosophical Transactions of the Royal Society of London. Series B, Containing Papers of a Biological Character*, vol. 213, pp. 89–142, 1925.

[80] M. R. Galankashi, E. Fallahiarezoudar, A. Moazzami, N. M. Yusof, and S. A. Helmi, "Performance evaluation of a petrol station queuing system: a

simulation-based design of experiments study," *Advances in Engineering Software*, vol. 92, pp. 15–26, 2016.

[81] A. Azadeh and S. Tarverdian, "Integration of genetic algorithm, computer simulation and design of experiments for forecasting electrical energy consumption," *Energy Policy*, vol. 35, no. 10, pp. 5229–5241, 2007.

[82] V. Prasad, A. M. Karim, Z. Ulissi, M. Zagrobelny, and D. G. Vlachos, "High throughput multiscale modeling for design of experiments, catalysts, and reactors: application to hydrogen production from ammonia," *Chemical Engineering Science*, vol. 65, no. 1, pp. 240–246, 2010.

[83] T. Gul, R. Bischoff, and H. P. Permentier, "Optimization of reaction parameters for the electrochemical oxidation of lidocaine with a design of experiments approach," *Electrochimica Acta*, vol. 171, pp. 23–28, 2015.

[84] I. Skrjanc, "Evolving fuzzy-model-based design of experiments with supervised hierarchical clustering," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 4, pp. 861–871, 2015.

[85] E. Jack Chen and M. Li, "Design of experiments for interpolation-based metamodels," *Simulation Modelling Practice and Theory*, vol. 44, pp. 14–25, 2014.

[86] M. Hegarty and M. A. Just, "Constructing mental models of machines from text and diagrams," *Journal of Memory and Language*, vol. 32, no. 6, pp. 717–742, 1993.

[87] World Squash Federation, "Specifications For Squash Balls." World Squash Federation, p. 2, 2015.

[88] World Squash Federation, "Specifications For Squash Courts." World Squash Federation, pp. 1–21, 2013.

[89] C. Dickerson and D. Battersby, "PSi Theme one analysis of the vehicle as a complex system," *Jaguar Land Rover*, 2016. [Online]. Available: http://www.psiprog.net/theme-1/. [Accessed: 29-Sep-2016].

[90] S. T. Saini, P. Lokhande, R. Deshmukh, and D. Baviskar, "Natural language processing on ambiguous sentence using NLP tools: core NLP, apertium and PRAAT," *International Journal of Innovative Research & Development*, vol. 5, no. 7, pp. 158–164, 2016.

[91] K. D. Bimson and R. D. Hull, "Unnatural Language Processing: Characterizing The Challenges In Translating Natural Language Semantics Into Ontology Semantics," in *Semantic Web*, M. Workman, Ed. London: Springer International Publishing, 2016, pp. 119–135.

[92] M. Schreiber, B. Kraft, and A. Zündorf, "Cost-efficient Quality Assurance of Natural Language Processing Tools Through Continuous Monitoring With Continuous Integration," in *Proceedings of the 3rd International Workshop on Software Engineering Research and Industrial Practice*, Austin, USA, 2016, pp. 46–52.

[93] "Stanford core NLP – a suite of core NLP tools," *Stanford University website*,

2016. [Online]. Available: http://stanfordnlp.github.io/CoreNLP/. [Accessed: 01-Jan-2017].

[94] E. Agirre, D. Cer, M. Diab, and A. Gonzalez-Agirre, "SemEval-2012 task 6: A pilot on semantic textual similarity," in *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012), in conjunction with the First Joint Conference on Lexical and Computational Semantics*, Montréal, Canada, 2012, pp. 385–393.

[95] S. . MacDonell, K. Min, and A. M. Connor, "Autonomous requirements specification processing using natural language processing," in *Proceedings of the ISCA 14th International Conference on Intelligent and Adaptive Systems and Software Engineering*, Toronto, Canada, 2005, pp. 266–270.

[96] M. A. K. Halliday and C. M. I. M. Matthiessen, *Halliday's introduction to functional grammar*, Fourth. New York: Routledge, 2014.

[97] OMG, "Semantics of Business Vocabulary and Business Rules," 2013. [Online]. Available: http://www.omg.org/spec/SBVR/1.0/PDF. [Accessed: 16-Mar-2017].

[98] G. Fliedl, C. Kop, H. C. Mayr, A. Salbrechter, J. Vöhringer, G. Weber, and C. Winkler, "Deriving static and dynamic concepts from software requirements using sophisticated tagging," *Data and Knowledge Engineering*, vol. 61, no. 3, pp. 433–448, 2006.

[99] D. Falessi, I. C. Society, and G. Cantone, "Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques," *IEEE Transactions on Software Engineering*, vol. 39, no. 1, pp. 18–44, 2013.

[100] C. D. Manning, J. Bauer, J. Finkel, S. J. Bethard, M. Surdeanu, and D. McClosky, "The stanford core NLP natural language processing toolkit," in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Baltimore, USA, 2014, pp. 55–60.

[101] "Natural language toolkit," *NLTK 3.0 documentation*, 2016. [Online]. Available: http://www.nltk.org/. [Accessed: 16-Mar-2017].

[102] "NLTK FAQ," *NLTK GitHub*, 2016. [Online]. Available: https://github.com/nltk/nltk/wiki/FAQ. [Accessed: 26-Nov-2016].

[103] S. Bird, E. Klein, and E. Loper, "Natural Language Processing with Python - Analyzing Text With The Natural Language Toolkit," 2009. [Online]. Available: http://www.nltk.org/book/. [Accessed: 16-Mar-2017].

[104] A. Wilton, A. De Houwer, J. Cenoz, S. M. Gass, D. Reed, A. Knapp, K. Kohn, L. Wei, J. Nortier, B. Seidlhofer, M. H. Verspoor, K. de Bot, and E. van Rein, *English in Europe Today*, vol. 8. Amsterdam: John Benjamins B.V, 2011.

[105] "About the OED," *Oxford English Dictionary*, 2016. [Online]. Available: http://public.oed.com/about/. [Accessed: 16-Mar-2017].

[106] F. J. Chantree, A. De Roeck, B. Nuseibeh, and A. Willis, "Identifying nocuous

ambiguity in natural language requirements," in *14th IEEE International Requirements Engineering Conference*, Minneapolis, USA, 2006, p. 203.

[107] M. K. Mathai, R. Venugopal, and J. T. Abraham, "Hybrid model for software development processes," *International Journal of Research in Engineering and Technology*, vol. 5, no. 1, pp. 198–202, 2016.

[108] S. Das, G. Singh, and B. Kumar, "Windows based graphical objective C IDE," *International Journal of Advancements in Technology*, vol. 7, no. 1, pp. 1–5, 2016.

[109] MathWorks, "The Language of Technical Computing," 2016. [Online]. Available: http://uk.mathworks.com/products/matlab/. [Accessed: 17-Mar-2017].

[110] National Instruments, "Benefits of programming graphically in NI LabVIEW," *National Instruments*. National Instruments, pp. 1–7, 2013.

[111] Nick Coghlan, "Should I use Python 2 or Python 3 for my development activity?," 2015. [Online]. Available: https://wiki.python.org/moin/Python2orPython3. [Accessed: 11-Apr-2016].

[112] H. F. Cervone, "Using Pugh matrix analysis in complex decision-making situations," *OCLC Systems & Services*, vol. 25, no. 2, pp. 76–81, 2009.

[113] C. Analytics, "Download anaconda now!," 2016. [Online]. Available: https://www.continuum.io/downloads. [Accessed: 16-Mar-2017].

[114] "Anaconda 4.0.0 package list," *Anaconda Documentaiton*, 2016. [Online]. Available: https://docs.continuum.io/anaconda/pkg-docs. [Accessed: 16-Mar-2017].

[115] "Installing NLTK data," *NLTK 3.0 documentation*, 2016. [Online]. Available: http://www.nltk.org/data.html. [Accessed: 16-Mar-2017].

[116] "NLTK 3.0 docmentation," *NLTK.org*, 2016. [Online]. Available: http://www.nltk.org/_modules/nltk/help.html. [Accessed: 16-Mar-2017].

[117] S. Sarkar, "Models of reduction and categories of reductionism," *Synthese*, vol. 91, no. 1949, pp. 167–194, 1992.

[118] T. Gordon and D. Greenspan, "The management of chaotic systems," *Technological Forecasting and Social Change*, vol. 47, no. 1, pp. 49–62, Sep. 1994.

[119] G. E. P. Box, "Science and statistics," *Journal of the American Statistical Association*, vol. 71, no. 356, pp. 791–799, 1976.

[120] E. Ronchi, E. D. Kuligowski, P. A. Reneke, and D. Nilsson, "NIST Technical Note 1822 the process of verification and validation of building fire evacuation models." pp. 1–85, 1822.

[121] R. J. Barnes, D. C. Gause, and E. C. Way, "Teaching the unknown and the unknowable in requirements engineering education," *2008 Requirements Engineering Education and Training, REET'08*, pp. 30–37, 2008.

[122] F. Boschetti, “A graphical representation of uncertainty in complex decision making,” *Emergence: Complexity & Organization*, vol. 13, pp. 146–166, 2011.

[123] C. Brückner and B. Swynnerton, “A new architecture for automotive hardware-in-the-loop test,” *ATZelektronik worldwide*, vol. 9, no. 3, pp. 40–43, 2014.

[124] E. Suhir, “Human-in-the-loop: probabilistic predictive modelling, its role, attributes, challenges and applications,” *Theoretical Issues in Ergonomics Science*, vol. 16, no. 2, pp. 99–123, 2014.

[125] M. M. K. Oishi, D. Tilbury, and C. J. Tomlin, “Guest editorial: special section on human-centered automation,” *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 997–998, 2016.

[126] G. Canuto da Silva and P. C. Kaminski, “Selection of virtual and physical prototypes in the product development process,” *International Journal of Advanced Manufacturing Technology*, vol. 84, pp. 1513–1530, 2016.

[127] A. Mouzakitis, J. Wei, and J. Wang, “Vehicle windscreen wiper mathematical model development and optimisation for model based hardware-in-the-loop simulation and control,” in *The 17th International Conference on Automation and Computing*, Huddersfield, UK, 2011, pp. 207–212.

[128] M. van Ratingen and A. Williams, “An update on the euro NCAP safety ratings program,” *Airbag*. Euro NCAP, Brussels, Belgium, pp. 1–39, 2014.

[129] P. Benjamin and M. Graul, “A Framework for adaptive modeling and ontology driven simulation (FAMOS),” vol. 6227, pp. 622705-622705–11, May 2006.

[130] A. Verma, K. Akella, and B. Perakath, “Using Ontologies For Simulation Integration,” in *2007 Winter Simulation Conference*, Washington, USA, 2007, pp. 1081–1089.

[131] E. J. Candes and M. Wakin, “An introduction to compressive sampling,” *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[132] D. C. Schmidt, “Model-driven engineering,” *IEEE Computer*, vol. 39, no. 2, pp. 25–31, 2006.

[133] V. Andrikopoulos, S. Benbernou, and M. P. Papazoglou, “On the Evolution of Services,” *IEEE Transactions on Software Engineering*, vol. 38, no. 3, pp. 609–628, 2012.

[134] F. Berman, “Got data?: a guide to data preservation in the information age,” *Communications of the ACM*, vol. 51, no. 12, pp. 50–56, 2008.

[135] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin, “A view of cloud computing,” *Communications of the ACM*, vol. 53, no. 4, p. 50, 2010.

[136] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. P. Anderson, “Cost-

benefit analysis of cloud codlputing versus desktop grids," in *lPDPS 2009 - Proceedings of the 2009 lEEE International Parallel and Distributed Processing Symposium*, Rome, Italy, 2009, pp. 1–12.

[137] A. Jacobs, "The pathologies of big data," *Communications of the ACM*, vol. 7, no. 6, p. 10, 2009.

[138] P. Benjamin, P. Mukul, and R. Mayer, "Using Ontologies For Simulation Modeling," in *Proceedings of the 2006 Winter Simulation Conference*, Monterey, USA, 2006, pp. 1151–1159.

[139] E. Honour and R. Valerdi, "Advancing an Ontology for Systems Engineering to Allow Consistent Measurement," in *Conference on Systems Engineering Research*, Los Angeles, USA, 2006, pp. 1–12.

[140] I. Niles and A. Pease, "Towards a standard upper ontology," *Proceedings of the international conference on Formal Ontology in Information Systems - FOIS', New York, USA,* 2001, pp. 2–9.

[141] J. S. Liang, "Generation of automotive troubleshooting configuration system using an ontology-based approach," *Computers in Industry*, vol. 63, no. 5, pp. 405–422, Jun. 2012.

[142] T. Karhela, A. Villberg, and H. Niemistö, "Open ontology-based integration platform for modeling and simulation in engineering," *International Journal of Modeling, Simulation, and Scientific Computing (IJMSSC)*, vol. 3, no. 2, pp. 1–36, Jun. 2012.

[143] J. Miller and G. Baramidze, "Ontologies for modeling and simulation: An initial framework," *ACM Journal*, vol. V, pp. 1–47, 2004.

[144] J. A. Miller and P. A. Fishwick, "Ontologies for modeling and simulation: issues and approaches," in *Proceedings of the 2004 Winter Simulaiton Conference*, Washington, USA, 2004, p. 264.

[145] P. Benjamin and K. Akella, "Towards ontology-driven interoperability for simulation-based applications," in *Proceedings of the 2009 Winter Simulation Conference Austin*, USA, 2009, pp. 1375–1386.

[146] M. Jamshidi, "System of systems engineering - new challenges for the 21st century," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 23, no. 5, pp. 4–19, 2008.

[147] J. Westbrook, N. Ito, H. Nakamura, K. Henrick, and H. M. Berman, "PDBML: the representation of archival macromolecular structure data in XML.," *Bioinformatics (Oxford, England)*, vol. 21, no. 7, pp. 988–92, Apr. 2005.

[148] B. Sauser, D. Verma, J. Ramirez-Marquez, and R. Gove, "From TRL to SRL: The concept of systems readiness levels," in *Conference on Systems Engineering Research*, Los Angeles, USA, 2006, pp. 1–10.

[149] J. S. Seibel, T. A. Mazzuchi, and S. Sarkani, "Same vendor, version-to-version upgrage decision support model for commercial off-the-shelf productivity applications," *Systems Engineering*, vol. 9, no. 4, pp. 296–312, 2006.

[150] A. Mohamed, G. Ruhe, and A. Eberlein, "COTS selection: past, present, and future," in *Proceedings of the International Symposium and Workshop on Engineering of Computer Based Systems*, Tucson, USA, 2007, pp. 103–112.

[151] S. J. E. Taylor and S. J. Turner, "Guidelines for commercial off-the-shelf simulation package interoperability," in *2008 Winter Simulation Conference*, Miami, USA, 2008, pp. 193–204.

[152] "Autosar," 2046. [Online]. Available: http://www.autosar.org/. [Accessed: 16-Mar-2017].

9 APPENDIX

9.1 APPENDIX PROCESS ELEMENTS

For each of the proposed process and sub processes a description of the elements has been express in the following tables.

9.1.1 SYSTEM CREATION LIFECYCLE

PROCESS ELEMENT	DESCRIPTION
1) Customer Want	For a system to be created there has to be a want or a need for it. In the manufacture of goods and services this manifests as a customer want. The organisation that is looking to create something to fulfil the customer wants first has to recognise that there is a want and capture an understanding of it.
2) Requirements	The capturing of system requirements is the task of understanding the customer want, distilling what is necessary to fulfil the wants as well as the needs of the customer, synthesising the information, and recording the statements. There are many methods and approaches that can be used to elicit requirements and ensure that they are; correct, concise and clear. It is considered outside the bounds of this research to specify how the requirements for the system are to be elicited and verified. However this process requires there to be a full set of requirements for the system being designed.
3) Architecture	From the requirements architecture for potential solutions can be formulated. Having an architecture specified can be of benefit to the simulation and modelling tasks later on in the process. The means by which the architecture is formed and represented is outside of the bounds of this research, however it is strongly recommended for architecture to be in place before simulation work is started is. Having a complete architecture of a potential solution is a prerequisite for other proposed methods.
4) Design	The design uses the requirements and architecture to produce a potential design that satisfies what the customers' wants and needs are. The design of a potential solution is what is tested using virtual simulation and testing. The means by which the engineers produce there potential designs is outside of the bounds of this research. The fact that designs are fully formulated and reflect the requirements which have been captured is what is needed for the processes that have been developed. A fully specified potential design is required for the following process to operate.

5 & 6) Virtual Simulation and Testing	To verify if a potential design is worth taking to the physical prototype stage virtual simulation and testing can be used. The concept is that the virtual testing will be quicker and cheaper than building physical prototypes and testing them. For testing to start a clear and complete design is needed along with architecture and a full requirement set. The potential design is tested in a virtual environment. This is a non-trivial task and is the primary focus of this body of research. The process by which the simulations are selected integrated and verified is the main focus of this body of work see the aims and objectives found in section 1.5. The SEIS process and its sub processes are detailed below. The models or simulations if functioning correctly will produce a data set that can be used to inform design decisions.
7) Do the Results of the Tests Support the Design	The results of the models and simulations will either serve to support the case for a specific design or will show flaws that were not investigated before testing took place. This decision is to be tempered by the confidence that engineers have in the results of the virtual testing. If the results of the virtual testing support the design physical prototyping is the next stage. If the results show the design to be found wanting, a redesign may be needed.
8) Build Prototype	The design that has been specified is to be realised with a physical prototype. Physical prototypes are often expensive due to the tooling and highly skilled labour that is needed. It is advised that this stage should only be entered into if the virtual testing shows promise or there is no known way to simulate it.

9) Test Prototype with Same Tests as Conducted Virtually	By conducting the same physical tests as what were conducted virtually previously ensures validation of the models, simulations and the integration of them. Using the same tests that are conducted physically as were conducted virtually gives a solid reference point. This reference point can not only be used for validation of the models and simulations but also for any future work concerning this platform. It is recognised that not all virtual tests can be reproduced in the real world as this can be the reason for testing some systems virtually in the first instance. Time is a critical factor in any product development and there may not be the time to conduct all of the virtual tests physically. However if the reuse of models and simulation is going to be more of a focus of organisations the validation of models is going to have to become more of a priority. For model verification to be conducted physical reference points are needed. In this instance reference points can only be captured by conducting physical testing. This issue is one that the organisations need to make on a strategic organisational level.
10) Does the Behaviour of Prototype Match that shown in Simulations?	By comparing the outputs of the physical testing with the output of the virtual testing the validation of the virtual models and simulations can take place. If within the organisation value is attributed to the virtual models and simulations being correct for use later the results feed back into the SEIS process and alterations can be made.
11) Does the Prototype fulfil Customer and stakeholders Requirements?	A comparison is to be conducted between the behaviour of the physical prototype and the requirements that were captured earlier in the process. An engineering decision is to be made at this point. If enough of the requirements are met by the physical prototype then the manufacture of the product can start. If it is decided that the physical prototype does not meet enough of the requirements then it is back to the design to see if changes can be made so that the potential design comply with the requirements.
12) Enter Manufacture	The design can be taken to the manufacturing phase. This involves taking the potential design and what was learnt and producing a means by which the product can be manufactured. The means by which this is done is outside of the bound of this research.

Table 9.1: Textual description of Systems creation Lifecycle stages.

9.1.2 VERIFICATION REPRESENTATION

PROCESS ELEMENT	DESCRIPTION
1) Item/s Being Verified	The section of work that is to be tested.
2) Requirement (Reference Point)	To ascertain if the item under test is satisfactory or not a known correct reference point is required. This can be requirements but could very easily be golden unit that is known to be correct.
3) Form Verification Test	An experiment is to be created for the purpose of which is to ascertain if the item under test has the attributes or operates in a way that is within tolerances of the reference point.
4) Using Formed Test Does Item Being Verified Meet Requirement Reference Point?	Conduct the test that was specified in stage 3. The outputs of the test are then to be compared to the reference point fixed in stage 2. The outputs of the test may need to be processed and put through various analytical tools before a meaningful comparison can be made.
5) Passed Verification Test	The item being testing is acceptable and falls within tolerances specified in the stage two.
6) Failed Verification Test	The item being tested has been defined as being outside of the boundaries specified in the verification test.

Table 9.2: Textual descriptions of the stages of the verification testing process.

9.1.3 VALIDATION REPRESENTATION

PROCESS ELEMENT	DESCRIPTION
1) Requirement Set (Reference Points)	To ascertain if the item under test is satisfactory or not a known correct reference point is required. For Validation this can be the requirements set or the opinions of the stakeholders of the system.
2) Defined Testing Procedure	An experiment is to be created for the purpose of which is to ascertain if the item under test has the attributes or operates in a way that is within tolerances of the reference point.
3) System Being Validated	The complete functional system that is to be tested.
4) Conduct Tests	The formulated test is to be conducted and the results stored.
5) Analyse Results	The results of the testing are to be analysed using the appropriate means for the data that is produced.
6) Does Item Being Validated Meet Reference Points?	Use appropriate methods to ascertain if the result of the analysis indicates whether the system being tested operates within the tolerances of the attributes or operations specified in the reference point.
7) Passed Validation Test	The system falls within the specified tolerances.
8) Failed Validation Test	The system falls outside of the specified tolerances.

Table 9.3: Textual descriptions of the stages of the Validation testing process.

9.1.4 SYSTEMS ENGINEERING IN INTEGRATION OF SIMULATIONS

PROCESS ELEMENT	DESCRIPTION
1) Understanding the Problem Space	The understanding of a problem space is the foundation for the information needed for decisions that are made throughout the rest of this process. If the understanding of the problem space is not present then the decisions that are made may be incongruent with the problem space. This understanding can be gained in many ways the specifics of which are outside of the bounds of this research however many methods exist in literature.
2) Requirements Specification of System	A detailed set of requirements are necessary for the rest of this process. Requirements are used as a reference point for not only what is going to be created and hence needs to be mimicked but also how the simulation representing the system being designed is to be tested. There are many ways to capture requirements and analyse them to get them to such a state whereby they can be relied upon to be correct, complete and concise. This process does not dictate the method by which the system requirements are to be elicited, refined or organised. The requirements are to be written in such a way as they can be tested against for the purpose of verification and validation.
3) Architecture of System Being Designed	To produce a simulation it requires an understanding of the nature of the system being simulated. Having the architecture of the system being designed gives a reference point as to what is to be mimicked as well as an indication of how the architecture of the simulation being designed may function. The architecture will show what the constituent sub-systems are and how they are intended to communicate. This division of the whole system into constituent parts may be of use when defining which available existing models could be selected.
4) System Design of System being Designed	A full candidate system design is needed as the exact nature of what needs to be simulated needs to be known by the modeller. The design does not need to be final and ideally it is to be in a state that allows for engineers to make changes depending on the results of the virtual tests. This complete design is needed as this is the basis of the underpinning ideas as to why simulations are used in the engineering design process. Without a concrete design at this stage decisions will be made by individual modellers which may course variations across the simulation potentially coursing issues with accuracy.

5) Design and Simulation Boundary	This is the point where the focus of work changes from the system being designed and moves to the simulation being designed. All of the knowledge regarding the simulation to be modelled is known and in a form that can be used later on in the process. All external factors to the design of the simulation are considered to be held constant from this point onwards. The work at this point does not need to be fully virtual in nature as physical test data; hardware in the loop and human in the loop testing can be used within this process.
6) Desire to Test Potential Design	There needs to be the desire in the first instance to learn more about the suitability of the potential design. The purpose of simulation needs to be understood and known by all those involved with the integration task. To maximise the value of the testing procedure the engineering project needs to be structured in such a way as to harness the results of the simulation and use them to have a positive effect on the thinking regarding the system design.
7) Design of Experiment	To ensure that the experiment that is being conducted with the simulation task actually tests what the designers need it to test the experiment itself needs to be designed. It is not simply enough to have the desire to test a system and crack on and start coding a simulation. A complex system with grate variability in its construction will require careful testing to ensure that all of the aspects of the system are tested in an appropriate way to ensure that time and effort are not wasted testing parts of the system in an incorrect or none representative way. The grater the variation present in a system, the grater the effort needs to be to exhaustively test all of the possible interactions that can occur during operation. The design of the experiment is to be conducted to ensure that only what is needed to be tested is tested as experimentation can be expensive and time consuming. The system designers need to have formulated clear ideas as to what parts or behaviours of the system being designed need testing, and what they are to be tested for. Ensure that all parameters are classified as being either independent or coupled with other parameters. Parameters which are coupled need to be known as this is a source of potential issue when sharing values between models and simulations during execution. Any biases are to be identified and reduced to a minimum wherever possible within the constraints of the project. If it is impossible to reduce the identified biases (to an extent that is acceptable) of the test then they are to be identified and taken into consideration when later looking at the results of

	the simulation. Potential sources of unintended variation that will be present in testing are to be identified and reduced wherever it is possible. Unintentional variation is often the source of unexplained variation that is present in results of tests. The way in which the test is conducted is to reflect the factors that will be present in the operational environment of the system being designed. The inclusion or emission of erroneous external factors may result in the behaviour of the models being very different to that of the final system in its operational environment. The identification of what factors in the experiment will be held constant and which will be changed at what point and in what order. The behaviour of the system may be dependent on the actions that have happened previously, if this is the case then it has to be taken into consideration in the testing that is conducted. If factors are to change during run time be that individual factors or a combination there of, consideration needs to be given as to what is going to be the trigger for the change and when it is going to happen. The simulation may not run at the same rate as it will do in its operational environment and as such trigger events may need to be expressed in other ways than in seconds of operation. At the end of this stage a clear understanding as to what the experiment is looking to achieve is to have been formed and the factors that can affect the results being taken into consideration as well as a clear testing plan.

8) Define Assumptions of Experiment	The assumptions that an experiment is based upon not only affect the results of the experiment but also what existing models can be selected for use as part of it. If there is an assumption mismatch between component models then the integration between them can be meaningless. By explicitly stating the assumptions that are being made it makes it possible to share the understanding between engineers working on the project, as well as functioning as a reference point as to compare existing models and simulations to during the selection process. If the assumptions of the individual component models and simulations align with each other the complete integrated simulation is closer to being semantically coherent. This stage should be relatively straightforward taking the work conducted regarding the DOE in the previous step.
9) Define Boundaries of Experiment	Each system has its bounds and an experiment is no different. By defining the bounds of the experiment it ensures that the effort of the engineers is focused on work that is concerned with achieving the intended experiment. Defining the bounds of the experiment and hence the content of the simulations can also help with reducing feature creep. By reducing feature creep considerable rework is reduced and the confidence in the simulation can be kept at a high level. By explicitly stating the boundaries of the experiment then the inputs and outputs will also be defined. The definition as to what the inputs and outputs of the experiment are to be also acts as a check as to if the DOE is being adhered to. For management this stage is useful as it will explicitly show what is and what is not being considered to be part of the experiment.
10) Simulation Requirements	Requirements act as a reference point for the creation of a system, a means of defining the capabilities of what is to be created and how the system is to be tested once it is created. A simulation can be considered a system in its own right and so is no different in these regards. As the simulation requirements will be used throughout the rest of this process considerable effort has been invested into how simulation requirements are to be written. For more information and specific guidance on this see the simulation requirements writing guide in section 3.6.

11) Set Standards If They Are To Be Used	Standards as shown in the section 2.3.3 have proven to be of use in particular situations and it is at this stage that it is worth investing time to ascertain if a standard would be of use and if so which one/s. There may be more than one standard that could be implemented at any one time, this is due to the there being standards for almost all potential engineering tasks. If standards are to be implemented across the simulation this decision needs to be conceded before models and simulations are selected for use.
12) Verification of Simulation Requirements and Standard Selection	Once a set of Simulation requirements have been elicited it is necessary to verify them against the design of experiment statements. If it has been decided to implement a standard it also requires verification against the DOE. If there are any discrepancies it will be necessary to rework the requirement set or chose a new standard. If the requirements set is changed at any point the standard choice will need to be reassessed.
13) Preliminary Simulation Architecture	To aid in the selection of existing models process a preliminary architectural design is required. The architecture at this point is to server as a means of structure as to what is expected to be built in the simulation being designed. This architecture is not intended to be the final one but rather serve as a stop gap to give an idea as to the parts of the system being designed that need to be simulated. The architecture will also dictate the way in which the constituent parts may communicate and how the simulation system will respond to specific stimulus. This architecture itself will only be loosely defined as it will change as more information about the available models and simulations becomes apparent. To aid in the design of the architecture the available infrastructure for combined simulation execution is to be assessed, as this will aid in the decisions as to which simulations are suitable for integration. Integration infrastructures have limitations be that; amount of data that can be transfer, types of data that can be transferred, number of models, type of model that can be executed, etc. This loosely defined architecture will follow that of the system being designed be that the components of the system or the way that it will respond to specific stimulus.

14) Verification of Preliminary Architecture	By verifying that the architecture matches the simulation requirements further work can be conducted in confidence reducing the likelihood for rework. The loosely defined nature of the preliminary architecture should not be used as an excuse for deviation from requirements. If the architecture does not comply with the simulation requirements it will require rectification.
15) Preliminary Simulation Design	The preliminary architecture can be taken further at this point and a preliminary simulation design can be formulated. This design will take the architecture and embellish the component parts and describe the intended behaviour. This preliminary design can be used as a reference point for later decisions as to weather a model or a simulation is suitable for integration selection.
16) Verification of Preliminary Simulation Design	With the preliminary design being built upon the preliminary architecture verification against the requirements can highlight deviations from initial intent of the experiment.
17) Are There Any Existing Simulations and Models?	Investigating as to if there are any existing models or simulations that could be of use for the experiment that could vastly reduce the cost and time it takes to implement the final simulation. The full question is 'Are there any existing models and simulations within or external to the organisation that the simulation designers have access to?'. Having existing models and simulations can decrease the production time of the overall simulation however it does come at a cost. The search for existing models and simulations may not be as straightforward as it first seams. The engineers undertaking this task have to have an overview of various modelling methods, the organisation that they are operating within, and the system being designed. This requires a very particular set of skills that only comes with experience of operating within an organisation on multiple projects. This is as much a human issue as it is a technical one. If an organisation has a central repository of all their models and simulations which could be interrogated it makes this step significantly quicker than if individuals had to manually ask departments or teams if they have any models or simulations they can uses.
18) Systems Engineering of Selection of Existing Models Process	This stage of the process aids in the selection of existing models for the purpose of integration into a larger simulation. This is a considerable task in itself and hence is a process defined in section 3.5.6.

19) Set Firm Architecture	The setting of the firm architecture is the decision of which models are going to be used, the way they are going to be organised and the means by which they are to communicate. With the knowledge and understanding that has been gained through the SESEM process an informed choice can be made as to which models and simulations to use in the integrated solution. With this newfound understanding the potential existing component parts in combination with the preliminary simulation architecture, preliminary design, and simulation requirements can now be used to formulate the firm architecture. If it has been decided that a standard or standards are to be used this decision is to be reassessed. If the standard is still deemed acceptable then it is to be complied with. If the decided upon standard is not found to be usable in this situation it may be worth readdressing the choice. If a new standard is to be used then it requires verification against the simulation requirements.
20) Assess Computational Requirements	All simulations have a computational overhead and a simulation made from existing models and or simulations is not different. Now that the models and simulations that are going to be used have been decided upon the overall computational burden can be calculated. Each component simulation is to be characterised to ascertain how much memory, processing power, storage, and time that it used to give a set amount of output be that a single iteration or full simulation run. The culmination of the computational overhead of all of the models is the result of the addition of the piece parts. Once the overhead is calculated the computer system that the simulation is intended to execute on can be specified (if it has not been specified in the requirements). The computers that an integrated solution executions on if more than one is needed then the way in which they communicate is as much a part of the integration problem as the mathematical functions in software. Therefore consideration as to the structure of the computational hardware is to be researched and constructed.
21) Verification of Set Architecture	Using the simulation requirements the Set architecture is to be verified. This verification is to ensure that the DOE is adhered to. Verification at this stage also critically covers the way in which the constituent parts will communicate. If there are issues with either the selected models or intended means by which they are to communicate the architecture is to be reworked.

22) Define Communication	With the architecture set and the component models selected the means by which they are to communicate can be explicitly defined. This may be using one specific development environment or using a specific communication infrastructure.
23) Detailed Simulation Design	Using the set architecture the detailed simulation design can be created. This involves looking at the individual models and simulations and assessing how they specifically are going to fit into the simulation being designed. With the knowledge of what is currently available and the means by which the parts are intended to communicate, engineers can specify the complete simulation being designed. This design also details and works to make the existing models fit with the defined architecture.
24) Define Gaps	By using existing simulations and models there will inevitably be gaps between the existing components which require filling to achieve the desired behaviour of the simulation being designed. For more information on how to conduct this task another process diagram has been developed see section 3.5.7 Figure 3.10. Depending on the size of the gap it may be necessary to conduct a full requirement set for the missing capabilities. In other cases the gap may be small and the solution simple.
25) Fill Gaps	Using the understanding and the definitions from the previous step work is to be conducted to create the new necessary simulation content.
26) Complete Integration Tables	Find and record the information required by the integration tables. Further information regarding these tables can be found in section 3.7.
27) Verification of Detailed Design	The full detailed design is to be verified against the simulation requirements. This ensures that the full detailed design complies with the simulation requirements and hence test the hypothesis within the decided upon constraints. This verification includes the new content from filling the gaps stage. This is the last time the individual piece parts are tested independently. Each of the inputs and outputs of the integrations is to be checked to ascertain if they are suitable. The assumptions of the components are to be checked to ensure that they are suitably similar.

28) Integrate Simulations	Up until this point the work has focused around ensuring that there is a suitable understanding of the purpose of the simulation, and that the component parts are suitable for the imitation of the phenomenon that is the focus of the test. It is at this stage that the simulations are integrated using whatever means has been decided to be used (integration environment etc). This process does not dictate what specific integration method is to be used, that decision is to be made by those who are undertaking the specific integration. This is due to there currently not be any one single solution to integrating all types of models and simulations.
29) Verification of Integrations	To verify as to weather a model or simulation integration is both successful and meaningful is a non-trivial exercise. This task is made difficult due to not necessarily having a reference point to compare the result to. To overcome this problem this process involves using a verification experiment for each of the models or simulations. The verification experiment is combined with the values that are held within the integration tables which define the bounds of the model and simulation operating range. Using the Verification experiment involves passing known values into one or more of the models or simulations and comparing known values out of the integrated models or simulations against the excepted values. This process is daisy chained across all of the component parts. The interfaces and communications between the component parts are compared to the set architecture. With the use of the verification experiment and the information in the integration tables reference points have been created. This allows for a meaningful verification to be conducted. If the integration does not comply with the simulation requirements then the integration step requires rework.

30) Verification of Simulation	Once the individual integrations have been verified similar principles can be applied to the whole simulation being designed. Using the method of verification experiments known values can be tracked through the integrated simulations ensuring that the exchanges between the simulations and models are accurate. This requires the verification experiments to align. To ensure that the verification experiments align it may be necessary to use the simulation design as a reference and produce verification experiments tailored for the simulation verification task. Each verification experiment would have to be created before the integration has been made. If the integration experiments of the individual component part experiments do not align with each other, an overall verification experiment is to be formulated which stays within the operational boundaries of the models is to be formulated.
31 – 35)Verification Chain when a Stage Fails	By using the stages of verification rather than relying on a single monolithic verification task, it breaks the task down into smaller more manageable tasks. By braking it down into smaller tasks allows for smaller more specialist teams of verification engineers to work on the sections that they are most knowledgeable about. One of the key technological advantages in using this process is that it has the potential to catch any exceptions earlier on in the simulation design process, while allowing for multiple engineers to work on the same project at the same time.
36) Conduct Experiment	With confidence in the integration of the models to an extent where it can pass the verification procedure the experiments can take place. The design of experiments will dictate nature of the simulations that are to be run and how they are to be run. Any results that are to be used are to be recorded for later analysis.
37) Feedback into Design Process	Once the results have been analysed and the answers to the questions that were posed in the design of experiments have been answered it is time to feed this knowledge back into the design process as shown in section 3.5.1. The feedback mechanism is not explicitly stated in this work however if the information is not fed back in a useable way to the larger project then the entire point of the process of simulation is null and void.

Table 9.4 Textual description of Systems Engineering in Integration of Simulations process elements.

9.1.5 SYSTEMS ENGINEERING OF SELECTION OF EXISTING MODELS AND SIMULATIONS

PROCESS ELEMENT	DESCRIPTION
1) Preliminary Simulation Design	For this process to successfully operate the simulation engineer needs to know the following information about the larger simulation being designed; what is the overall goal, the intended architecture, and critically the simulation requirements. This stage is not strictly part of the SESEM process but rather prerequisite. The simulation must be at the preliminary design phase otherwise decisions that will be made in this process may lead to the simulation being designed away from the intended DOE.
2) Boundary of the Existing Model Selection Process	This point in the diagram denotes where the boundary of this process lies. All of that which has come before this point are outside of this process (SESM) and the boundary where by information leaves this process is detailed later. The required inputs to this process are the preliminary system design, a full simulation requirements set, and the awareness that there may be existing candidate models and simulations which could be used as part of the simulation being designed.
3) Asses the Model and Simulation Landscape	To ensure that those who are undertaking the selection task are in a position to not only find the models and simulations but also know the value of what they are looking at. They must also be well versed in the current model and simulation landscape both internal and externally to their organisation. It is therefore well worth those who are conducting this stage to take some time to understand what modelling and simulation practices are used within the domains that the simulation being designed is to operate within. The assessment of the landscape also includes investigating what is available within the organisation that is conducting the simulation being designed. If this is not the first time that this modelling domain has been utilised by an organisation there is also likely to be a wealth of tacit knowledge held by individuals within said organisation. This knowledge may have the potential to vastly reduce the amount of work that needs to be undertaken, increase the accuracy of the resultant simulation, or hold key information about the models that could be used. The information and experience of individuals in an organisation is not to be ignored or underestimated.

4) Are Potential Models Available?	From the assessment of the simulation landscape there may or may not be models available which results in a straightforward question.
5) End of Process Feed Back to Main Process	As there are no potential models available this process is terminated. The feedback to the main process is that there are no usable existing models or simulations.
6) Is This a New Product / Platform?	Within organisations it is common for iterations of products to be made based on a common platform. This single platform with multiple variants is a concept that has been shown to reduce time to market of a range of different products. This decision is to focus where candidate models and simulations can be located. If the project that the simulation being designed for is a part of or is an incremental improvement of an existing product platform then existing models and simulation potentially contain usable content, that will require minimal modification may exist. If this is a completely new product or platform for the organisation then the engineers will have to look elsewhere for existing candidate models and simulations.
7) Locate Previous Product Models and Simulations	As it has been decided that this project is iterating on an existing work the models and simulations from the previous project is an ideal place to start. Finding the models, simulations, data sets and critically there documentation that was used in the development of the previous product as this has the potential to give a solid base to start working from. As these models are of a similar system they may vastly reduce the necessary work as the existing model or components may be used with little to no modification to model the new system. This is due to potentially very similar overall behaviour of the product.
8) Evaluate Candidate Models or Simulation Functions	Evaluate what the capabilities of the located models are and what they calculate. A basic understanding of what the model are looking to achieve is to be gained.

9) Is Model or Simulation Documentation Available?	The documentation of a model is often the only insight into the modellers' viewpoint when they created the model. The viewpoint of the modeller is critical when models are integrated. The viewpoint captures all of the assumptions, experience and reasoning that went into creating a model or simulation. If the modeller themselves is not available to give this information in person for whatever reason the only insight is going to be the model itself and any accompanying supporting documentation. In some instance it may not be possible to even interrogate the model directly, so the documentation may be the only source of information as to what the model does and how it does it. The information that is required for successful meaningful integration can be found in section 3.7 Information Needs for Model Integration.
10) Can Someone in the Team Understand the Models or Simulations Explicitly	As there is not sufficient documentation the necessary information must be found in an alternative way. As this model or simulation is one that has been utilised in a previous project it is possible that tacit knowledge is held by an member of the organisation. If this person or persons is still within the organisation the necessary information may be captured by asking them directly about the model or simulation. The other possibility is that the model or simulation is relatively simple and an individual in the integration team may be able to fully understand the model or simulation and capture the required information. Whichever method is used if there are one or more persons who have the necessary knowledge the integration documentation can be created. All sections of the integration tables do not need to be completed at this stage however the model information and structure do see section 3.7 regarding integration tables.
11) Evaluate Candidate Models Functions with new understanding	Using the documentation or understanding of what the model tests and how it goes about doing so. The capabilities are to be evaluated.

12) Do Models and Simulations Match Section of the Simulation Requirements exactly or in part?	Using the documentation gathered in the necessary integration information tables it is now possible to ascertain if the current simulation fulfils the requirements of the simulation being designed. As the simulation being designed may have many sub models or simulations it is possible that an existing simulation fully or partially fulfils one or more requirements. If the model does not fulfil any of the new simulation requirements then the selected model is not useful for this new simulation being designed and is to be discarded. As this simulation is based on an existing platform or product it is likely that the model or simulation is only partly applicable to the new requirements of the simulation being designed. If this is the case then model or simulation will require rework or modification to make it fully compliant to the new simulation requirements. There is the possibility that a model fulfils a section of the requirements without exception. In this case it is a possible candidate for selection.
13) Assess if Individual Models or Simulations can be Modified	From the previous step it has been found that the models or simulations require modification or rework to make it comply with the simulation being designed requirements. Some models and simulations can be modified others cannot. An example where a simulation may not be modified is if it was purchased using COTS and the model or simulation is a compiled piece of code that cannot be deconstructed. If a model or simulation does not comply with requirements and cannot be modified then it is unusable for this project and is to be discarded.
14) Selected Models and Simulations Unusable	If a model or simulation is not fully understood or does not fit the Simulation requirements, and cannot be made to comply then it is to be discarded. If this is the case it will then be necessary to locate another simulations or models. A record of which models has been assessed and found wanting is recommended to be kept as this can reduce rework of different people assessing the same models or simulations for suitability time and time again.
15) Can Requirements be Met by Changing Parameters?	Often the quickest and easiest change to make a model or simulation to make it comply with the simulation being designed requirements is to change its parameters and variable values. This can bring a general simulation into line with a specific solution space. For a candidate model or simulation to comply by having parameters changed it must operate across the solution space in the first place for a simple change in parameters to align it with the specific requirements of the simulation being designed.

16) Locate Candidate Models and Simulations	Most engineering models and simulations are based around physical principles and as such existing models and simulations exist in one form or another. These existing models have the potential to reduce the development time of the Simulation being designed. The task is to locate any existing models and simulations that have the potential to mimic the phenomenon that the simulation being developed is focused around.
17) Is Model or Simulation Documentation Available?	The documentation of a model is often the only insight into the modellers' viewpoint when they created the model. The viewpoint of the modeller is critical when models are integrated. The viewpoint captures all of the assumptions, experience and reasoning that the modeller uses when creating a model or simulation. If the modeller is not available to give this information in person for whatever reason the only insight are going to be the model itself and the supporting documentation. In some instance it may not be possible to even interrogate the model directly so the documentation may be the only source of information as to what the model does and how it does it. The information that is required for successful meaningful integration can be found in in section 3.7.
18) Can Someone in the Team Understand the Model Explicitly?	With the lack of documentation complete explicit understanding is necessary for successful meaningful integration. Therefor as these models or simulations has no documentation it may be possible for an individual to capture the complete understanding necessary. This activity is made possible if the model or simulation is formulated from basic physics models then the simulations can be easier to understand. A decision needs to be made as part of this step 'Will it take more effort to understand an existing model or just make a new one?'. This is an engineering judgement that needs to be made by those involved.
19) Do The Models or Simulations Match a Section of the simulation requirements?	Using the understanding gained from the previous step it will dictate the decision as to if the model or simulation is suitable to use, if the simulation needs modification, or if it is of no use to the simulation being designed at all.

20) Asses if located models and simulations can be modified	From the previous step it has been found that the model or simulation requires modification or rework to make it comply with the simulation being designed requirements. Some models and simulations can be modified others cannot. An example where a simulation may not be modified is if it was purchased using COTS and the simulation is a compiled piece of code that cannot be deconstructed. If models have been found outside of the organisation the engineers may have found open source models and simulations. Just because they are open source does not mean that they are free to modify and use freely. If a model or simulation does not comply with requirements and cannot be modified then it is unusable for this project.
21) Is it Possible to Access Variables and Parameters?	As changing variables and parameters is often required to align models and Simulations with what is required for the simulation being designed the question as to if this is possible has to be addressed. Not all models and simulations are created to have the capability for a user to change the variables or parameters. Some models and simulations are hard coded to only mimic one specific situation and cannot be changed without reworking the entre code and mathematical model from the ground up. The parameters also have finite ranges that they are valid over.
22) Change Parameters to Comply with Requirements	Change model or simulation variables or parameters so that the model or simulation complies with the Simulation being designed requirements.
23) Rework of Models and Simulations Needed	It has been established that the models or simulations currently being assessed requires rework. Define what work is needed to bring the model or simulation into line with the Simulation being Designed requirements. This stage does not require the action of making changes just assessing how much work is needed to make the necessary changes and specifying said works.
24) Does an Alternative With Less Rework Exist?	The rework of models and simulations can be a costly resource heavy exercise. Any possibility to reduce the amount of rework is preferable, so if there is an alternative that is believed to require less rework then that is that one which is select for the next stage. It is to be noted that a balance is to be made between looking for alternatives which require less work and conducting the rework of the existing model or simulation. This is an engineering judgement call that has to be made.

25) Rework Model or Simulation	Using the identified changes that need to be made from a previous step and conduct the necessary work to align the existing model or simulation with the simulation being designed requirements.
26) Use other model/s or Simulation/s	Leave the identified models and use the alternative model that requires less rework.
27) Verification	At this point it is intended that the model is at that stage where it is suitable for selection to be used in the simulation being designed. Therefore it is also intended that the model fully complies with the requirements of the simulation being designed. The item under test is the model or simulation that has been reworked and the reference point is the requirements of the simulation being designed. If the model or simulation passes this test it is to be considered for selection in the integrated simulation. Whereas if the model or simulation fails the verification then further rework will be needed.
28) Complete Integration Tables	Fill in the integration tables with the knowledge gained from previous elements of the SESEM process. See 3.7 for more information regarding integration tables.
29) Available for Selection	If a model or simulation passes the final verification it is suitable for use in the simulation being designed.
30) End of Systems Engineering of Selection of Existing Models	This step marks the end boundary of the SESEM process. The outputs of this process feed directly into the Systems Engineering in Integration of Simulations process.

Table 9.5 Textual description of the stages of the Systems Engineering of Selection of Existing Models process.

9.1.6 DEFINING GAPS IN SEIES (A SUB PROCESS)

PROCESS ELEMENT	DESCRIPTION
1) Detailed Design	The detailed design is not strictly a component of this process but it is necessary for the process to operate. For a full explanation of this step see textual description in SEIS process section 9.1.4.
2) Boundary of the Existing Model Selection Process	This boundary defines the start of the defining gaps in the systems engineering in integration of simulations process.
3) Assess Where Selected Models and Simulations Fulfil Simulation Requirements	The differences between the requirements of the simulation being designed and the capabilities of the simulations that have been selected are in essence the gaps which need to be identified. In this step the task is to assess the capabilities of the selected models and it is ascertained which requirements they fulfil. The assessment is to also cover the models communication to ascertain if the component will be able to communicate as specified in the simulation being designed architecture.
4) Are All Requirements Met?	Use the analysis conducted in the previous step with the requirements of the simulation being designed to answer the question 'Are all of the requirements of the simulation being designed met?' If the answer to this question is yes then the gaps between the models and simulations will only be of a communication issue. If not all of the requirements are met then there needs to be work done to meet them.
5) Asses Existing Model Boundaries	All models and simulations have boundaries. The boundaries of the models and simulations are made up of what the modeller intended to capture. The boundaries of the model or simulation can be modified in one respect by the inputs and outputs that are accessible. The inputs and outputs that a model or simulation has reflects an aspect of the phenomenon being mimicked or is related to the operation of the model or simulation itself. The manipulation of the inputs gives control (the extent to which was defined by the original modeller) of the range of situations that the model or simulation can represent. It is therefore necessary for a detailed understanding of what all of the inputs and outputs represent within each model or simulation.

6) Assess Necessary Transformations to Comply With Requirements	It has been defined that the collection of existing models and simulations do not satisfy all of the requirements of the simulation being designed. It is therefore necessary for a transformation to make the selected models and simulations complicit. The task is to assess what transformations are required of the models and simulations to make them comply with the requirements. If the communications are defined (in the systems being designed requirements) then at this stage it will be necessary to ascertain the transformation (if required) of the way the models or simulations communicate.
7) Define Requirements for works to fill the gaps	For all of the identified transformations the work that needs to be done is to be specified. A requirements document for the work that needs to be done is recommended as this facilitates keeping track of and ensures that any modifications only bring the model or simulation into compliance with the requirements of the simulation being designed. Having this step discourages the urge to just get on with it the necessary transformations, while allowing for more than just the individual who is assessing the models and simulations being able to make the necessary changes. This will help to ensure that unnecessary transformational work is not conducted.
8) Assess Data That Requires Transfer	Part of the task of integrating existing models and simulations requires the establishment of the communications between component parts. For use later on in the SESEM process the information that is needed about the data is, the rate that the data is produced, the means by which it is transferred, and how it is to be assessed and recorded.
9) Define Requirements for Implementation of Set Communication	Using the assessment from the previous step define the requirements for the communication between the specific models and Simulations. The communication methods are to be documented so they can be inspected later.
10) End of Define Gaps	This is the boundary of this sub process. The output feeds back into the SEIS. The outputs of this process are requirements documents for the work to be done to fill the gaps between the selected models and simulations.

Table 9.6 Textual description of the Defining Gaps in the Systems Engineering in Integration of Simulations process.

9.1.7 FILL GAPS IN THE SYSTEMS ENGINEERING IN INTEGRATION OF SIMULATIONS

PROCESS ELEMENT	DESCRIPTION
1) Define Gaps	Defining the gaps between the available existing models and simulations is covered in another process see section 3.5.6. The output of the defining gaps process feeds directly into this process.
2) Boundary of the Existing Model Selection Process	This denotes the bounds of this process all before are prerequisite inputs.
3) Gain Understanding of Requirements from Define the Gaps Process	It is important that the gaps that are going to be filled are fully understood. If what is needed to fill the gap is not fully understood any efforts to rectify it will be misguided and unlikely not be what is needed. If there is any uncertainty with the requirements then those who compiled them are to be contacted and any ambiguities resolved. If that is not possible investigate the issues by going back to the Define Gaps process. Do not continue with this process if there is not a detailed understanding of what needs to be produced.
4) Architect the solution/s	With the understanding of the previous step construct architecture for a potential solution. There are many ways in which architecture can be formed however it is not within the remit of this process to denote how to construct the architecture of the solution/s. Having the basis of architecture gives structure for the rest of the potential solution to be based. It will also ensure that suitable consideration has been made that any work will fit into the larger simulation being designed.
5) Design Potential Solutions	The potential solution is to be designed using the basis of the architecture that has been produced in the previous step. The design of the potential solution is to specify exactly what is going to be produced.
6) Verification	Once a potential design has been formulated it is poignant to verify it against the requirements that were specified in the defining the gaps process. This verifications purpose is ensure that before considerable effort is spent implementing the design complies with the requirements. If the design passes the verification then the work can progress onto the building of the solution. If the design does not pass the verification then rework of the potential solution is required. If rework is needed go back to the 'Architect the solution' stage of the process.
7) Build Solution	The task of building the potential solution starts in earnest at this point. The potential solution is to be implemented.

8) Verification	To ensure that what has been produced is going to fit into the simulation being designed, and produced values that are meaningful, it is to be verified against the filling the gaps requirements. To verify the potential solution at this stage could potentiality save significant amount of time when compared to it being a problem and it manifesting as part of a complicated system of simulations in an integrated solution later on.
9) End of Fill Gaps in the SEIS	This is the end of the Fill the gaps process. The output of this process feeds directly into the SEIS process.

Table 9.7 Textual explanation of the stages of the Sill the Gaps in Part of Systems Engineering in Integration of Simulations process.

9.2 APPENDIX IDENTIFIED TOPICS FOR MODEL AND SIMULATION INTEGRATION AND REASONING

For each of the topics of information that has been identified in Figure 3.12 details and reasoning are given for each of the elements in the mind map.

9.2.1 IDENTIFIED TOPICS FOR MODEL AND SIMULATION STRUCTURE

To gain enough of an understanding if two model or simulations are suitable for integration there requires an understanding about the structure of the models and simulations in question.

IDENTIFIED TOPIC	DESCRIPTION AND REASONING
Structure	The structure of a model or simulation covers the way it is put together. Depending on the methods used to create it has an impact on the way in which it will behave when run. This is a critical area for integration. Structure is a broad topic that has been decomposed to identify individual aspects which require specific consideration.
Architecture	By understanding the architecture of the model or simulation, an idea of how it communicates and what is indeed available to be communicated, becomes apparent. This understanding includes the interfaces, the way in which the interfaces send and receive information, and if there is any loops or feedback present. As these are three large topics they have be decomposed further.
Feedback loops	Having feedback loops in a simulation means that the simulation has an element memory associated with it. Meaning each output is affected by actions of the previous iteration. Having such feedback loops present can cause issues with integration. This is due to the first number of outputs or iterations having to be disregarded as erroneous, as well as causing issues of having to have the same instance of the model or simulation running for the full duration. Models that have feedback loops in can be a challenge when it comes to updating what were initial conditions during run time or 'on the fly'. If feedback loops are present and such values are intended to be changed during run time, an investigation of the validity of the results of the model or simulation has to be found. The way in which results are pulled out of such a model or simulation that has one or more loops can be interesting as some instantiations will only output a data sets upon completion of a simulation run, not during individual iterations.
Blocking	The way that the simulation passes out data from a queue can have a significant effect the way in which data is handled. If blocking is used in the memory queue, if the queue is full it will block any attempts to add to it. If a queue is empty and an attempt is made to read form it the attempt will be blocked. This can hamper multithreading capabilities if used. This is an implementation architectural concern that affects the implementation environmental concerns.

Response time	When discussing time with regard to models and simulations great care has to be observed. This is due to time being the basis of many of the simulations and also referring to how long a simulation takes to conduct an action. Both of these times are needed to be understood when integrating models and simulations. In this work response time refers to the time it takes for the simulation to conduct an intended action. Response time can vary from less than a second to well over a week. It is recognised that the hardware that the model or simulation is being run on has a part to play in this. For this reason there is a section that defines which hardware the model or simulation has been bench marked using. When conducting large scale co-simulations with many models and simulations, the individual response times can be critical to a meaningful output from the whole.
Time	Time in this respect refers to the time that a model or simulation represents in the real world. A single second in a simulation may take days or even weeks in the real world to compute. The representation of time can take may different forms and can be described in terms of unit, base, and step.
Time - Unit	The unit that is used to describe the amount of time be that seconds, hours, days or otherwise. It is all too easy to make the assumption that a model or simulation used one unit when in fact it uses another, hence they are to be explicitly stated.
Time - Base	The base unit is the smallest increment that the modal or simulation is capable of. This information can be used to ascertain if a common time base can be found across two or more models or simulations.
Time - Step	This is the step that the model or simulation makes after each calculation. This may be fixed or a parameter that can be user defined on execution. The Step is a multiple of the base in the units defined. The length of the step has to be wholly divisible by the base.
Interface	The inputs, outputs, and the means by which communication can occur are considered encapsulated by the umbrella term of 'interface'. The identified necessary information about possible interface types are: Inputs and Outputs, Source, Description, Default Value, Format, Expected Sample, Rate, Accuracy, Range, Unit, Type, Base, Step, and Name

Inputs and Outputs	Information and data flows into an input whereas information and data flows from an output. Understanding the nature of an interface is the understanding of the information that needs to go into a model or simulation and understanding what is represented by the output. To capture an understanding of the inputs or outputs the following need to be known; source, description, default value, format, expected sample rate, accuracy, range, unit, type, base and step.
Inputs and Outputs: Source	This is a description of where the input or output comes from. This may seem straightforward but when there is the situation where there are many hundreds of inputs and outputs having a description of where the input comes from or output goes to can reduce development time.
Inputs and Outputs: Description	The inputs or outputs represent some part of the phenomena being mimicked. An understanding of what this represents can ensure that two interfaces with the same name that represent two different aspects of the phenomena being erroneously connected together. The description only has to be as long as necessary to accurately capture what the input or output represents.
Inputs and Outputs: Default Value	Some models and simulations have default values that operate if there is not a value passed to it. If an input does not have a default value and is left floating it can have a negative effect on the accuracy of the outputs if the simulation functions at all.
Inputs and Outputs: Format	All information captured on a computer takes a format, be that an array of floating point numbers or otherwise. Not all of the information captured by one data type can be completely represented in another. It is important if passing data to or receiving data from a model or simulation to know what format the date is or needs to be. There are well known and documented methods for converting between data types however it also comes with an overhead and often a loss of information.
Inputs and Outputs: Expected Sample Rate	Each port will have an expected sample rate irrespective if it is an input or an output. If the port is not sampled at the rate which it was intended, issues of aliasing can occur. The Nyquist limit will apply if using different sample rates. This can be a significant problem with connecting models or simulations over a distributed network.

Inputs and Outputs: Accuracy	The accuracy of the model or simulation relates to how close the results of the calculations are to the phenomena that is being mimicked. At the start of a project or within an organisation a decision needs to be made as to how the accuracy is going to be measured as there are many different methods for assessing and recording accuracy levels. This is further complicated if the model or simulation has not been compared to the real world yet and as such the accuracy is not a definable amount but rather an estimated value.
Inputs and Outputs: Range	The range is defined as the difference between the minimum possible value and the maximum possible value. Models and simulations that are implemented digitally have finite possible ranges. The limiting factor of the range of the model or simulation may be a design decision that has been made (eg data types used) or it could be the range for which the physics equations are valid (scale issue). If the range of a model or simulation is exceeded the accuracy of the model or simulation is adversely affected in some cases drastically so.
Inputs and Outputs: Unit	All values handed two or taken from a model or simulation represent an aspect of understanding. Each input and output will have a unit (even unit less values have a conceptual name). Defining what the units are will ensure that even if the output of one is the same concept as the input of another the same units are used. For instance temperature, one model could be in Kelvin and the other in Celsius.
Inputs and Outputs: Required	If an input is it required for the model or simulation to run.
Inputs and Outputs: Base	The base unit is the smallest increment that the model or simulation is capable of for the specific input or output that is in question. This information can be used to ascertain if a common base for a unit can be found across two or more models or simulations.
Inputs and Outputs: Step	This is the step that the model or simulation uses or makes after each calculation of a specific input or output respectively. This may be fixed or a perimeter that can be user defined however it is often strongly linked to the time step. The Step is a multiple of the base in the units defined. The length of the step has to be wholly devisable by the base.

Inputs and Outputs: Name	All inputs and outputs are to have a unique name or identifier. This does not necessarily need to be words it can easily be an alphameric code. However if an alphameric code is used a document of what each code represents needs to be kept for reference. There are instances where single models and simulations have hundreds of inputs and outputs so when it comes to integrating them with another model or simulation all of the inputs and outputs need to indefinable.
Stop command	For each modal or simulation it is preferable if a means of stopping it mid execution is available. Having such a stop command can save engineers a great deal of time when setting up co-simulation as testing does not necessarily require the full simulation run as well if an error is identified during execution all component models or simulations can be terminated. Documentation of how to stop the model or simulation mid execution is recommended. This may be programmatically possible or it may require a direct user input.
Start command	To execute a model or simulation there will be a procedure that needs to be followed. This may be possible programmatically or it could require some user defined steps. This information is needed so that a means of stating the co-simulation can be designed.
User Interface Type	Define how the user interacts with the model or simulation. Is it command line, GUI , or another type.
Constant	A constant is considered to be a static value that is defined at the start of a model or simulation. Some models and simulations have constants that can be altered within a set range. To define such a constant the following information is required; description, default value, format, range, unit, type, base, and name. These have the same definition and require the same information as detailed above.
Constant: Unit	The SI unit that the value represents.
Constant: Value	The numerical value that the constant is to take.
Constant: Format	All information captured on a computer takes a format, be that an array of floating point numbers or otherwise. Not all of the information captured by one data type can be completely represented in another. It is important if passing data to or receiving data from a model or simulation to know what format the date is or needs to be. There are well known and documented methods for converting between data types however it also comes with an overhead and often a loss of information.

Constant: Name	All constants are to have a unique name or identifier. This does not necessarily need to be words it can easily be an alphameric code. However if an alphameric code is used a document of what each code represents needs to be kept for reference. There are instances where single models and simulations have hundreds constants so when it comes to integrating them with another model or simulation all of the constants need to indefinable.

Table 9.8 Identified topics for model and simulation integration and reasoning.

9.2.2 VERIFICATION EXPERIMENT

The purpose of this information is to provide the foundation for a verification experiment to allow for known outputs from none inputs to be checked. This allows for an engineer who has had no dealings with the model or simulation, to check if it is set up correctly and is operating as it was originally intended. This verification experiment can be what was used to test the model or simulation before it was considered to be good enough to use. Having verification experiments it is possible to set up an experiment where the variation comes from the integration alone rather than the data entering the models manually. The recorded characteristics of the verification experiment cover how to set up the model or simulation, how the models or simulation behaves while running, and the expected results when it is conducted.

IDENTIFIED TOPIC	DESCRIPTION
Total Run time	Models and simulations take time to execute. The total run time is the length of time it takes for the model or simulation to complete the defined experiment. It is to be noted that the total run time will change dependent on the hardware that it is run on. However this metric will indicate if there is an issue if it is run at a later date and found to operate far quicker or slower than the recorded metric when run on comparable hardware.
Output Frequency	All outputs of a model or simulation have an update frequency, even if the output frequency is at the end of the run. Having an expected output frequency allows for the potential of assessing if the output of the model or simulation in question can feed into the input of another model or simulation.
Inputs and Outputs	All inputs are to have the structure and content as defined in interfaces including: source, description, default value, format, expected sample rate, accuracy, range, unit, type, base, step, and name.
Parameters	A parameter is considered to be an input value that is not necessarily directly accessible to the user and was intended by its creator to not change during run time. All of the parameters that do not have default values are to be specified. The internal parameters that cannot be user defined or accessed at all do not need to be specified.
Environment	The Environment covers all of the supporting software and hardware that is required to enable the simulation to operate. See section 2.62.8 for more information.

Table 9.9 Identified topics for Verification Experiment.

9.2.3 MODEL INFORMATION

To gain enough of an understanding if two model or simulations are suitable for integration there requires an understanding about the models or simulations in question.

IDENTIFIED TOPIC	DESCRIPTION
Documentation	Specific information that is required to be in the documentation and will not be found elsewhere this covers; assumptions, language, file type, location, basis of the model, generation tool, and requirements documentation.
Documentation: Assumptions	The assumptions that the creator of the model or simulation uses to capture the behaviour of the phenomena being mimicked is a key part of assessing if two models or simulations can be integrated and the extent to which the result is meaningful. Identifying the assumptions may be relatively obvious and straightforward whereas others will require a greater understanding of what is going on in the model or simulation. A select few of the possible many examples include; the body is a dimensionless particle, the object is in two dimensional space, or friction is negligible.
Documentation: Language	The language that the model or simulation is documented in rather than the computer language that is used to create it. In multinational organisations the language that is used to document works is not necessarily unilaterally homogeneous. Despite translation programs becoming ever more accurate the subtleties of understanding that are necessary to make an informed decision of if and how two models or simulations can be integrated cannot be relied to be captured at present. Hence the language that is used to document a model may dictate who in the engineering team investigates which models and simulations, or if there is even the necessary linguistic skills to do so are available. Examples of languages that are used for documentation include but at not limited to, English, French, Mandarin etc.
Documentation: File type	All files on a computer have a file extension that the computer uses to know what format to open it in. Knowing the file extension can aid in the decision of which if any integration environments could potentially be used. There are many bespoke file types from the different development environments however some typical file types for models and simulations are .exe, .xml and .m

Documentation: Location	Once a potential model for integration is found documenting the location of where it is stored can save considerable time later on. Digital locations can change over time. If there are multiple locations for the model or simulation it is worth documenting the most likely to remain viable or even detail multiple locations.
Documentation: Basis of model	There are many starting points for producing a model or simulation. Knowing what the modeller used as the basis of understanding can give an insight into if other models are constructed from a similar base behaviour. Common basses for models and simulations to be built from are but not limited to; HIL data, physics principles, ROM, and transient data.
Documentation: Generation tool	For a model or simulation to be created a tool or development environment must be used. Having an understanding of which tool was used in the generation gives the integration engineer an insight into the type of support side integration that is required. If using an integration environment knowing which tool was used in the creation of a model or simulation can instantly rule weather the model or simulation can be used or not. Examples of common tools for model or simulation creation include, Matlab, LabVIEW, Ansys, COMSOL Multiphysics, however there are many more with new specific model and simulation packages being released every month.
Documentation: Requirements Document	The location and name of the requirements document that specified the work to produce the model or simulation.
Version	Having a record of the versions that are created of a model or simulation allows a group of people to ensure that they are all using the exact same model or simulation and not two or more different versions. Having both the name and version number gives two reference points to reduce potential ambiguity.
Date of Publication	Having a date as well as a name or number allows for an understanding of the time between each iteration as well as if any rework of a model or simulation has been done without a new version being created. Publication refers to when the model or simulation was made available to those who were not a part of the development of it.
Name or Number	A common and effective way of keeping track of versions is by using a unique name and time stamp. There are multiple tools that allow for this activity to be automated within an organisation. There is often a logical progression of the naming convention to make it obvious which older the versions were created in.

Type of model	There are many types of models and simulations that can be used to mimic or represent different behaviours knowing which type has been used gives those who are integrating them an understanding of what information to expect, look for and what they can indeed gain from using it. Examples of different types of model in this case are but not limited to, free body, fluid flow, chemical, CFD, Finite Element Analysis.
Description	The description of the model or simulation is intended for the capture an overview of the subject, behaviour, and reasoning behind the implementation of the model or simulation while giving the opportunity to capture information that does not fit into the other sections of model information without the need to create a new section for each new model.
Author	Having information as to who created the model or simulation can aid in the integration. If the engineer who has been tasked with the integration of the models or simulations has no specific understanding and cannot find any information about the model or simulation, then having the possibility of communication with the author could be beneficial. It is recognised that if COTS is use by the organisation then having at least a company name can aid in the tracking down of who made the model or simulation in the first place, as well as tracking any lost documentation.
Standard compliance	If the model or simulation complies to a standard and those integrating it are aware it can reduce the amount of time that needs to be investigated trying to understand the model or simulation in question.
Source Files	Having access to the source files potentially allows for the integration engineer to make modifications to change aspects of the behaviour of the model or simulation. If modification is required then having the location for any such files and if they can be edited could drastically save time in the long run.
Source Filed: Editable	Not all source files to models and simulations are in a form that an integration engineer can edited. If a source file cannot be edited then the behaviour that is the result of the file is also fixed. If the file cannot be edited it may be necessary to produce replacement source files, however this may affect the confidence level of the validity of the model or simulation. In some circumstances the source file may be completely unavailable or restricted.

Source Files: Location	If the model and simulation source files are available then the location of them is to be captured. When it comes to integrating the models and simulations having the location of the source files without having to search for them can reduce the integration time.
Source Files: Name	The name of the source files of the model or the simulation source files reduces ambiguity when looking for and potentially modifying the files.
Intellectual Property	Who owns the intellectual property of the model or simulation can change where, when, and how, the model or simulation can be used. When COTS is involved there can be stipulations placed on the model or simulation limiting how it can be used.
Intellectual Property: Owner	The owner of the model or simulation has ultimate say on how it can be used. With this identified they can be contacted if there are any issues or questions as to use be they licencing or otherwise.
Licence	Some models and simulations are available to use but only in accordance with a licence. There are many different licences that all have their own terms and conditions. If a usage licence is in pace on a model or simulation the details of how it can be used are to be located and agreed.

Table 9.10 Identified topics for Model Information.

9.2.4 MODEL ENVIRONMENT

To gain enough of an understanding if two model or simulations are suitable for integration there requires an understanding about the environment that the models or simulations in question operate within.

IDENTIFIED TOPIC	DESCRIPTION
Hardware Requirements	The model or simulation requires a minimum amount of hardware to operate. The hardware that the verification experiment is run on affects the characteristics of run time as well as inputs and output frequency. The information regarding the hardware the verification test includes, Graphics cards (if used), Network requirements, Peripherals (if any are needed), Storage, Memory, and processor.
Hardware Requirements: Graphics	If the model or simulation uses any specific graphics cards or even just the on board graphics card this is to be specified. If the model or simulation does not require any graphics capability this is to be noted.
Hardware Requirements: Graphics: GPU Clock Speed (MHz)	The clock rate of the GPU used when to run the model or simulation.
Hardware Requirements: Graphics: Size of the memory bus (bits)	The transfer rate of the memory buss used for the GPU.
Hardware Requirements: Graphics: Amount of available memory (MB)	The size of the GPU on-board memory that is accessible during programming.
Hardware Requirements: Graphics: memory clock rate	The clock rate of the GPU on-board memory.
Hardware Requirements: Network	Some models and Simulations require network access to execute. If the model needs access to a network or the internet this is to be captured. This can be because of licences used by some software packages.
Hardware Requirements: Peripherals	If the model or simulation requires any peripherals to operate then the exact name, make and model are to be recorded. This could include any required peripherals such as but not limited to, data cards, instrumentation, dongles, or any special user interfaces.
Hardware Requirements: Peripherals: Interface	The interface by which the hardware connects e.g. USB, parallel port, etc
Hardware Requirements: Peripherals: Name	The official manufacture and vendor name (if different) given to the peripheral.
Hardware Requirements: Peripherals: Driver	Details of any drivers that are necessary for the peripheral to function. All relevant information is to be captured. This information is often found in the peripherals supplied documentation.

Hardware Requirements: Peripherals: Vendor	The vendor used to purchase the peripheral for the project.
Hardware Requirements: Storage	The consideration of storage for the model or simulation is not limited to the size of the file that the model or simulation is captured by but also any files that it produces. The output files of some models and simulations can be many terabytes where as the model or simulation may only be a few megabytes.
Hardware Requirements: Memory	To operate the model or simulation needs to be loaded into working memory. Different models and simulations have differing memory requirements. The amount of memory that the model or simulation needs to run smoothly without issues are to be recorded.
Hardware Requirements: Processor	The model or simulation will require processing power and time to run. The speed and architecture of the processor used it to be recorded e.g. quad core 6.3GHz. If the model or simulation is run on a cluster or other such high performance machine (rather than a general purpose computer) more detail is needed so a similar specked machine that would be suitable could be found or emulated at a later date.
Environment software	To run a model or simulation may require other programs to be running or installed on the computer that it is executing on. This includes but not limited to operating systems (OS), runtime engines and dependences which will be explained in more detail below.
Environment software: Operating system	Many models and simulations only function correctly when run on a machine which has a specific operating system installed. The full name and version number of the operating system is to be recorded including any major service packs or updates.
Environment software: Runtime Engine	Some models and simulations that were constructed in a development environment may need that development environments runtime engine to operate. Often these runtime engines are free or come at a reduced cost and will require installation separately to the model or simulation. The full name and version and vendor are to be recorded.
Environment software: Dependencies	The model or simulation may call for other files or programs to be installed on the same machine and be able to access them. Dependencies can cause a lot of trouble when reusing a model or simulation as they often get missed out of documentation and are not stored in the same location across different machines. To capture the necessary information the name, location, how it is called, and the version are all required information.

Name	However the dependency is named it is to be recorded. This is not a description of the file but rather what it is actually referred to as in the modal or simulation. This may not be a unique identify if so this is to be noted and an additional reference appended to the description for later use.
Environment software: Dependencies: Version	Whichever method or means by which the producer of the dependency uses to identify which version the file is, is to be recorded. This may be but not limited to, a name, a library that it is part of, or an alphameric code.
Environment software: Dependencies: Static or Dynamics	The way in which a dependency is referenced in the model or simulation will define where the dependency will need to be on a different machine to the one the model or simulation was originally developed on. This is due to the way in which the model or simulation looks for the dependency in memory. If it uses a static reference then it will only look in the defined path location, whereas if it is a dynamic reference it will follow the directory that the modal or simulation is placed in and search for the dependency. There are strengths and caveats for both of the methods. The developer of the model or simulation will have made a decision for a reason. Whether the dependency is statically or dynamically defined is to be captured.
Environment software: Dependencies: Location	Regardless of whether the dependency is statically or dynamically defined a location is to be recorded of where a copy of the dependency can be found, be that a location on a network drive or even a website where it can be downloaded. The former is preferable to the latter due to the likelihood of it being accessible at a later date.
Environment software: Dependencies: Version	The version of the dependency that is called.
Environment software: Dependencies: Static or Dynamic	Is the path to the dependencies statically or dynamically defined in the model or simulation?

9.3 INTEGRATION TABLES

The integration tables that form part of the proposed process.

9.3.1 BLANK INTEGRATION TABLES

Name:	
Version:	
Date of publication	
Version Number	
Type:	
Descriotion:	
Author:	
IP:	
Owner:	
Licence:	
Standard complience:	
Documentation:	
Assumptions:	
Language:	
File type:	
Location:	
Basis of model:	
Generation tool:	

Source Files	**Source file name**	**Location**	**Editable**
			Yes
			No
			NA

Figure 9.1Model Information

User interface type														
Time														
Base														
Step														
Response time														
Architecture														
Feedback loops														
Blocking														
Constants	Name	Format	Unit	Value										
NA														
Stop Command														
Start command														
Inputs and Outputs	**Name**	**Input or Output**	**Base**	**Step**	**Unit**	**Range min**	**Range max**	**Accuracy**	**Expected sample rate**	**Format**	**Default Value**	**Description**	**Source**	**Required**

Figure 9.2 Model Structure

Operating sytem:																		
Runtime Engine:																		
Hardware Requirements																		
Graphics																		
GPU Clock speed																		
Size of memory bus																		
Amount of avalible memory																		
Memory clock rate																		
Network																		
Disk space																		
RAM																		
CPU																		
Peripherals	**Name**														**Vender**	**Driver**	**Interface**	
NA																		
Dependencey	**Name**														**Location**	**Static or dynamic**	**Version**	
NA																		

Figure 9.3 Model Environment

Enviroment									
Time									
total run time									
output frequency									
Parameters	Name	Format	Unit	Value					
NA									
Inputs and Outputs	**Name**	**Input or Output**	**Base**	**Step**	**Unit**	**Format**	**Input Value or expected output value**	**Description**	**Required**

Figure 9.4 Verification Experiment

9.3.2 Developmental Case Study Completed Integration Tables

The tables that were used as part of the developmental case study concerning the virtual testing of the effects of differing compounds on the behaviour of a squash court.

Particle Moving in Free Space Integration Tables

Name:	Particle Moving in Free Space
Version:	
Date of publication	01/03/2015
Version Number	V1
Type:	Newtonian equations of motion
Descriotion:	Model requires final velocity, starting velocity, distance and sample rate to be known and it calculates acceleration, time that it takes for the ball to get from one end to the other, time sample, displacement steps in the time sample and the distance that ball travels in a single time sample.First the acceleration of the particle is calculated. Second the time for the particle to complete the distance is calculated. Third the time sample step is calculated. Forth the displacement steps in meters
Author:	Robert Luff
IP:	
Owner:	Loughborough University
Licence:	Academic
Standard compliience:	None
Documentation:	
Assumptions:	Simulation uses Newtonian equations of motion. The ball is considered to be a particle. All of the inputs are known values. There is no wind resistance. The ball is only moving in one plane. No obstacles in the path of movement. SI units are used throughout meters, seconds, meters per second, and meters per second per second
Language:	English
File type:	.vi
Location:	C:\Users\elr2\Google Drive\PHD\My Work\LabVIEW Files\NLP models
Basis of model:	Physics
Generation tool:	LabVIEW

Source Files	**Source file name**	**Location**	**Editable**
	Model of Ball moving	C:\Users\elr2\Google Drive\PHD\My Work\LabVIEW Files\NLP models	Yes
			NA

Figure 9.5Particle moving in free space model information.

User interface type	GUI
Time	
Base	1 second
Step	0.000101812
Response time	2 seconds
Architecture	1 single loop publishes output on competion of program
Feedback loops	One loop
Blocking	NA

Constants	Name	Format	Unit	Value
NA				
Stop Command	Stop Loop (input)	Boolean	True False	
Start command				

Inputs and Outputs	**Name**	**Input or Output**	**Base**	**Step**	**Unit**	**Range min**	**Range max**	**Accuracy**	**Expected sample rate**	**Format**	**Default Value**	**Description**	**Source**	**Required**
	Starting Velocity U (mps)	Input	10	0.000001	Meters per second	0.000001	999999		0.5 Hz	floating-point	0	The starting velocity of the particle	Front panel	Yes
	Distance (S) (m)	Input	10	0.000001	Meters	0.000001	999999		0.5 Hz	floating-point	0	The distance the particle has to travel	Front panel	Yes
	Final Valocity (V) (mps)	Input	10	0.000001	Meters per second	0.000001	999999		0.5 Hz	floating-point	0	Final velocity of the particle	Front panel	Yes
	Sample Rate (NB samples ofcer Time)	Input	10	1	Interger	1	99999		0.5 Hz	floating-point	0	Number of samples to take during the flight time of the particle	Front panel	Yes
	Stop Loop	Input	1	1	Boolean	0	1		0.5 Hz	Boolean	0	Command to stop the simulation prematurely	Front panel	No
	Loop counts	Output	10	1	Interger	1	999999		0.5 Hz	Interger	0	Displays the number of iterations of calculation loop	Front panel	NA
	Time (T) (Seconds)	Output	10	0.000001	Seconds	0.000001	999		0.5 Hz	floating-point	0	Time taken for particle to cover distance	Front panel	NA
	Acceleration (A) (meters per second per second)	Output	10	0.000001	Meters per second per second	0.000001	300		0.5 Hz	floating-point	0	Acceleration of the ball from start of flight to the end	Front panel	NA
	Time Sample (Ts) (seconds)	Output	10	0.000000001	Seconds	0.000000001	1		0.5 Hz	floating-point	0	Calculated time sample rate	Front panel	NA
	Displcement step St (meters)	Output	10	0.000001	Meters	0.000001	5		0.5 Hz	floating-point	0	Calculated displacement step	Front panel	NA
	Displacement step St (meters) Array	Output	10	0.000001	Meters	0.000001	5		0.5 Hz	Array of floating pint numbers	0	Displacement steps from each iteration of a calculation loop	Front panel	NA

Figure 9.6 Particle moving in free space Structure.

Operating sytem:	Windows 7																		
Runtime Engine:	LabVIEW 2013 (32bit)																		
Hardware Requirements																			
Graphics																			
GPU Clock speed	400MHz																		
Size of memory bus	64 Bit																		
Amount of avalible memory	1 GB																		
Memory clock rate	450 MHz																		
Network	NA																		
Disk space	38 KB																		
RAM	162,352 K																		
CPU	Intel® Core™ i5-3380M CPU @ 2.90GHz (4 CPUs)																		
Peripherals	**Name**															**Vender**	**Driver**	**Interface**	
NA																			
Dependencey	**Name**															**Location**	**Static or dynamic**	**Version**	
NA																			

Figure 9.7 Particle moving in free space Environment

Enviroment	LabVIEW									
Time										
total run time	4s									
output frequency	0.25 Hz									
Parameters	Name	Format	Unit	Value						
NA										
Inputs and Outputs	**Name**	**Input or Output**	**Base**	**Step**	**Unit**	**Format**	**Input Value or expected output value**	**Description**	**Required**	
Starting Velocity U (mps)	Input	10	0.000001	Meters per second	0.000001	floating-point	50	The starting velocity of the particle	Yes	
Distance (S) (m)	Input	10	0.000001	Meters	0.000001	floating-point	4	The distance the particle has to travel	Yes	
Final Valocity (V) (mps)	Input	10	0.000001	Meters per second	0.000001	floating-point	48.22	Final velocity of the particle	Yes	
Sample Rate (NB sampels ofcer Time)	Input	10	1	Interger	1	floating-point	1000	Number of samples to take during the flight time of the particle	Yes	
Stop Loop	Input	1	1	Boolean	0	Boolean	0	Command to stop the simulation prematurely	Yes	
Loop counts	Output	10	1	Interger	1	Interger	982	Displays the number of iterations of calculation loop	Yes	
Time (T) (Seconds)	Output	10	0.000001	Seconds	0.000001	floating-point	0.0814498	Time taken for particle to cover distance	Yes	
Acecleration (A) (meters per second per second)	Output	10	0.000001	Meters per second per second	0.000001	floating-point	-21.8539	Acceleration of the ball from start of flight to the end	Yes	
Time Sample (Ts) (seconds)	Output	10	1E-09	Seconds	1E-09	floating-point	8.14E-05	Calculated time sample rate	Yes	
Displcement step St (meters)	Output	10	0.000001	Meters	0.000001	floating-point	4.00319	Calculated displacement step	Yes	
Displacement step St (meters) Array	Output	10	0.000001	Meters	0.000001	Array of floating pint numbers	E1 0.00407241	Displacement steps from each iteration of a calculation loop	Yes	

Figure 9.8 Particle Moving in Free Space verification Experiment

Energy Transfer Model Integration Tables

Name:	Energy Transfer model
Version:	
Date of publication	01/03/2015
Version Number	V1
Type:	Energy transfer model Newtonean
Descriotion:	Energy transfer model of a ball hitting a surface and losing energy on impact and rebound
Author:	Robert Luff
IP:	
Owner:	Loughborough University
Licence:	Academic
Standard complience:	None
Documentation:	
Assumptions:	The energy loss from the impact is specified by the user as a percentage No loss of energy from air resistance The surfaces are completely smooth with no protrusions The ball is a particle The ball can move freely within the three dimensional space
Language:	English
File type:	.vi
Location:	C:\Users\elrl2\Google Drive\PHD\My Work\LabVIEW Files\NLP models
Basis of model:	Physics
Generation tool:	LabVIEW

Source Files	**Source file name**	**Location**	**Editable**
	Energy transfer model.vi	C:\Users\elrl2\Google Drive\PHD\My Work\LabVIEW Files\NLP models	NA
			NA
			NA
			NA
			NA
			NA
			NA
			NA

Figure 9.9 Energy Transfer Model, Model Information

User interface type	GUI												
Time	NA												
Base	NA												
Step	NA												
Response time	0.5 seconds												
Architecture													
Feedback loops	None												
Blocking	NA												
Constants	Name	Format	Unit	Value									
Stop Command													
Start command													
Inputs and Outputs	Name	Input or Output	Base	Step	Unit	Range min	Range max	Accuracy	Expected sample rate	Format	Default Value	Description	Source	Required
	Mass (Kg)	Input	10	0.0001	Kilogram	0.001	999999	0.001	50	floating-point	0	Mass of the ball	Front panel	Yes
	Velocity (meters per second mps)	Input	10	0.0001	Mps	0.001	999999	0.001	50	floating-point	0	Velocity of the ball	Front panel	Yes
	Energy transfer efficiency (%)	Input	10	0.0001	Meters per Second	0.001	999999	0.001	50	floating-point	0	Percentage of energy out	Front panel	Yes
	Remaining energy	Output	10	0.0001	Jules	0.001	999999	0.001	50	floating-point	0	Energy remaining held by the ball	Front panel	NA
	Kinetic energy (Jules)	Output	10	0.0001	Jules	0.001	999999	0.001	50	floating-point	0	Kinetic energy of the ball after rebound	Front panel	NA
	Energy lost (Jules)	Output	10	0.0001	Jules	0.001	999999	0.001	50	floating-point	0	Energy lost by rebound	Front panel	NA
	Returning velocity (mps)	Output	10	0.0001	Meters per Second	0.001	999999	0.001	50	floating-point	0	Velocity of ball after rebound	Front panel	NA

Figure 9.10 Energy Transfer Model, Structure

Operating sytem:	Windows 7
Runtime Engine:	LabVIEW 2013 (32bit)
Hardware Requirements	
Graphics	
GPU Clock speed	400MHz
Size of memory bus	64 Bit
Amount of avalible memory	1 GB
Memory clock rate	450 MHz
Network	NA
Disk space	38 KB
RAM	162,352 K
CPU	Intel® Core™ i5-3380M CPU @ 2.90GHz (4 CPUs)

Peripherals	**Name**	**Vender**	**Driver**	**Interface**
NA				

Dependencey	**Name**	**Location**	**Static or dynamic**	**Version**
NA				

Figure 9.11 Energy Transfer Model, Environment

Enviroment	LabVIEW										
Time											
total run time	0.5 seconds										
output frequency	single point										
Parameters	Name	Format	Unit	Value							
NA											
Inputs and Outputs	**Name**	**Input or Output**	**Base**	**Step**	**Unit**	**Format**	**Input Value or expected output value**	**Description**	**Required**		
	Mass (Kg)	Input	10	0.0001	Kilogram	floating-point	0.3	The mass of the particle	Yes		
	Velocity (meters per second mps)	Input	10	0.0001	Mps	floating-point	50	The velocity of the particle	Yes		
	Energy transfer efficiency (%)	Input	10	0.0001	Meters per Second	floating-point	5	The energy loss as a percentage of a rebound from a surface	Yes		
	Remaining energy	Output	10	0.0001	Jules	floating-point	356.25	The energy remaining after a rebound	NA		
	Kinetic energy (Jules)	Output	10	0.0001	Jules	floating-point	375	The kinetic energy before interaction with surface	NA		
	Energy lost (Jules)	Output	10	0.0001	Jules	floating-point	18.75	The energy lost from the interaction with the surface	NA		
	Returning velocity (mps)	Output	10	0.0001	Meters per Second	floating-point	48.734	The velocity of the particle after rebounding off the surface	NA		

Figure 9.12 Energy Transfer Model, Verification

Squash Court in or Out Model Integration Tables

Name:	Squash Court in or Out Model
Version:	
Date of publication	01/03/2015
Version Number	V1
Type:	Cartesian representaion of a squash court
Descriotion:	Physical body being modelled is based of the specifications laid out in the document World Squash Federation (WSF) Recommended Standards Approved by the WSF January 2013. The simulation used the (X,Y,Z) Cartesian notation of three dimensional space. (0,0,0) is taken to be the bottom left corner as if a player had just walked through the door.
Author:	Robert Luff
Requiremet document name	NA
Requiremet document locaiton	NA
IP	
Owner:	Loughborough University
Licence:	Acedemic
Standard complience:	None
Documentation:	
Assumptions:	Any ball hitting any of the lines is considered "in" The dimensions of the ball are not considered The ball is a particle The Tin does not stand out of the wall The ball can move freely within the three dimensional space The surfaces are completely smooth with no protrusions The players are not considered
Language:	English
File type:	.VI
Location:	C:\Users\elr12\Google Drive\PHD\My Work\LabVIEW Files\NLP models
Basis of model:	Physics
Generation tool:	LabVIEW
Source Files	**Source file name**
	Squash Court model

Location	**Editable**
C:\Users\elr12\Google Drive\PHD\My Work\LabVIEW Files\NLP models	Yes
	NA

Figure 9.13 Squash Court in or Out Model, Model Information

User interface type	GUI														
Time	NA														
Base	NA														
Step	NA														
Response time	0.5 seconds														
Architecture															
Feedback loops	None														
Blocking	NA														
Constants	Name	Format	Unit	Value											
Stop Command															
Start command															
Inputs and Outputs	Name	Input or Output	Base	Step	Unit	Range min	Range Max	Accuracy	Expected sample rate	Format	Default Value	Description	Source	Required
	Total Length	NA	10	0.001	milimeters	0	100	0.001	50	floating-point	9750	Court total length	Front panel	Yes
	Total With	input	10	0.001	millimeters	0	100	0.001	50	floating-point	6400	Court total with	Front panel	Yes
	Total Height	input	10	0.001	millimeters	0	100	0.001	50	floating-point	5640	Court total height	Front panel	Yes
	Height of Tin	Input	10	0.001	millimeters	0	100	0.001	50	floating-point	530	The height of the tin from the floor	Front panel	Yes
	Front Wall Bottom of Service Line	input	10	0.001	millimeters	0	100	0.001	50	floating-point	1830	height from the floor to bottom sercice line	Front panel	Yes
	Front wall top of Service Line	input	10	0.001	millimeters	0	100	0.001	50	floating-point	4620	height from the floor to the top service line	Front panel	Yes
	Length to short line	Input	10	0.001	millimeters	0	100	0.001	50	floating-point	5490	length from front wall to dhort line	Front panel	Yes
	With of service box	input	10	0.001	millimeters	0	100	0.001	50	floating-point	1650	with of service box from side wall	Front panel	Yes
	With of back Quarter	input	10	0.001	millimeters	0	100	0.001	50	floating-point	3200	with of back quarter from side wall	Front panel	Yes
	Depth of service box	Input	10	0.001	millimeters	0	100	0.001	50	floating-point	1650	depth of sercice box from short line	Front panel	Yes
	Height of back wall line	input	10	0.001	millimeters	0	100	0.001	50	floating-point	2180	hight from the floor to back wall line	Front panel	Yes
	X Coordinate	input	10	0.001	millimeters	0	100	0.001	50	floating-point	0	the x coordinate position within the court	Front panel	No
	Y Coordinate	Input	10	0.001	millimeters	0	100	0.001	50	floating-point	0	the y coordinate position within the court	Front panel	No
	Z Coordinate	input	10	0.001	millimeters	0	100	0.001	50	floating-point	0	the z coordinate position within the court	Front panel	No
	Front Half	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball in the front half	Front panel	NA
	Left Service box	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball in the left service box	Front panel	NA
	Right Service box	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball in the right service box	Front panel	NA
	Left Quarter	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in the left quarter	Front panel	NA
	Right Quarter	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in the right quarter	Front panel	NA
	Back half	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in the back half	Front panel	NA
	Top wall	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in contact with the top wall	Front panel	NA
	Front wall	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in contact with front wall	Front panel	NA
	Bottom wall	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in contact with bottom wall	Front panel	NA
	Tin	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in contact with the tin	Front panel	NA
	Out Bounds side wall left	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in contact with wall left out of bounds	Front panel	NA
	In bounds side wall left	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in contact with wall left in bounds	Front panel	NA
	Out bounds side wall right	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in contact with right wall in bounds	Front panel	NA
	In bounds side wall right	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in contact with left wall in bounds	Front panel	NA
	Out bound height	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is out bounds in height	Front panel	NA
	In bound height	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in bounds in height	Front panel	NA
	Back wall out bound	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in contact with back wall out bounds	Front panel	NA
	Back wall in bound	Output	1	1	True False	0	1	1	50	Boolean	0	Indecates if ball is in contact with back wall in bounds	Front panel	NA
	Height of line at point x	Output	10	0.001	millimeters	0	100	0.0001	50	floating-point	0	hight of side wall bound line at point x	Front panel	NA

Figure 9.14 Squash Court in or Out Model, Model Structure

Operating sytem:	Windows 7			
Runtime Engine:	LabVIEW 2013 (32bit)			
Hardware Requirements				
Graphics				
GPU Clock speed	400MHz			
Size of memory bus	64 Bit			
Amount of avalible memory	1 GB			
Memory clock rate	450 MHz			
Network	NA			
Disk space	38 KB			
RAM	162,352 K			
CPU	Intel® Core™ i5-3380M CPU @ 2.90GHz (4 CPUs)			
Peripherals	**Name**	**Vender**	**Driver**	**Interface**
None				
Dependencey	**Name**	**Location**	**Static or dynamic**	**Version**
None				

Figure 9.15Squash Court in or Out Model, Environment

Enviroment	LabVIEW										
Time											
total run time	0.5 seconds										
output frequency	single output										
Parameters	Name	Format	Unit	Value							
NA											
Inputs and Outputs	**Name**	**Input or Output**	**Base**	**Step**	**Unit**	**Format**	**Input Value or expected output value**	**Description**	**Required**		
	Total Length	NA	10	0.001	milimeters	floating-point	9750	Court total length	Yes		
	Total With	Input	10	0.001	milimeters	floating-point	6400	Court total with	Yes		
	Total Height	Input	10	0.001	millimeters	floating-point	5640	Court total height	Yes		
	Height of Tin	Input	10	0.001	millimeters	floating-point	530	The height of the tin from the floor	Yes		
	Front Wall Bottom of Service Line	Input	10	0.001	millimeters	floating-point	1830	height from the floor to bottom sercice line	Yes		
	Front wall top of Service Line	Input	10	0.001	millimeters	floating-point	4620	height from the floor to the top service line	Yes		
	Length to short line	Input	10	0.001	millimeters	floating-point	5490	length from front wall to dhort line	Yes		
	With of service box	Input	10	0.001	millimeters	floating-point	1650	with of service box from side wall	Yes		
	With of back Quarter	Input	10	0.001	millimeters	floating-point	3200	with of back quarter from side wall	Yes		
	Depth of service box	Input	10	0.001	millimeters	floating-point	1650	depth of sercice box from short line	Yes		
	Height of back wall line	Input	10	0.001	millimeters	floating-point	2180	hight from the floor to back wall line	Yes		
	X Coordinate	Input	10	0.001	millimeters	floating-point	0	the x coordinate position within the court	No		
	Y Coordinate	Input	10	0.001	millimeters	floating-point	0	the y coordinate position within the court	No		
	Z Coordinate	Input	10	0.001	millimeters	floating-point	0	the z coordinate position within the court	No		
	Front Half	Output	1	1	True False	Boolean	0	Indecates if ball in the front half	NA		
	Left Service box	Output	1	1	True False	Boolean	0	Indecates if ball in the left service box	NA		
	Right Service box	Output	1	1	True False	Boolean	0	Indecates if ball in the right service box	NA		
	Left Quarter	Output	1	1	True False	Boolean	0	Indecates if ball is in the left quarter	NA		
	Right Quarter	Output	1	1	True False	Boolean	0	Indecates if ball is in the right quarter	NA		
	Back half	Output	1	1	True False	Boolean	0	Indecates if ball is in the back half	NA		
	Top wall	Output	1	1	True False	Boolean	0	Indecates if ball is in contact with the top wall	NA		
	Front wall	Output	1	1	True False	Boolean	0	Indecates if ball is in contact with front wall	NA		
	Bottom wall	Output	1	1	True False	Boolean	0	Indecates if ball is in contact with bottom wall	NA		
	Tin	Output	1	1	True False	Boolean	0	Indecates if ball is in contact with the tin	NA		
	Out Bounds side wall left	Output	1	1	True False	Boolean	0	Indecates if ball is in contact with wall left out of bounds	NA		
	In bounds side wall left	Output	1	1	True False	Boolean	0	Indecates if ball is in contact with wall left in bounds	NA		
	Out bounds side wall right	Output	1	1	True False	Boolean	0	Indecates if ball is in contact with right wall in bounds	NA		
	In bounds side wall right	Output	1	1	True False	Boolean	0	Indecates if ball is in contact with left wall in bounds	NA		
	Out bound height	Output	1	1	True False	Boolean	0	indecates if ball is out bounds in height	NA		
	In bound height	Output	1	1	True False	Boolean	0	indecates if ball is in bounds in height	NA		
	Back wall out bound	Output	1	1	True False	Boolean	0	indecates if ball is in contact with back wall out bounds	NA		
	Back wall in bound	Output	1	1	True False	Boolean	0	indecates if ball is in contact with back wall in bounds	NA		
	Height of line at point x	Output	10	0.001	millimeters	floating-point	0	hight of side wall bound line at point x	NA		

Figure 9.16 Squash Court in or Out Model, Verification Experiment

9.4 NLP APPLICATION POC CODE

Each of the functions of the NLP POC is detailed in sections9.4.1through 9.4.14.

9.4.1 SETTING_UP_FILES_TO_COMPAIR.PY

```
#Set up two files to be compared
#Read in file 1 and classify
#Read in file 2 and classify
#Name all of the constants with the prefix of 1 or 2 respectively

Doc_A_path='C:\\Anaconda3\\Text_files\\Documentation_of_a_Particle_Moving_in_Free_
Space.txt'
Doc_B_path='C:\\Anaconda3\\Text_files\\Documentation_of_Squash_Court_in_or_Out_Mod
el.txt'

from Text_charactaristics_fn import Text_charactaristics
Doc_A_token_words_full,Doc_A_word_list,Doc_A_tagcount,Doc_A_taglists,Doc_A_number_
of_lines,Doc_A_NB_tags,Doc_A_NB_words,Doc_A_percent_of_tag,Doc_A_percent_of_words=
Text_charactaristics(Doc_A_path)
Doc_B_token_words_full,Doc_B_word_list,Doc_B_tagcount,Doc_B_taglists,Doc_B_number_
of_lines,Doc_B_NB_tags,Doc_B_NB_words,Doc_B_percent_of_tag,Doc_B_percent_of_words=
Text_charactaristics(Doc_B_path)

#print("Doc_A_percent_of_tag",Doc_A_percent_of_tag,'\n')
#print("Doc_B_percent_of_tag",Doc_B_percent_of_tag,'\n')

from Actual_Dif import Actual_mod_Dif

#To produce a bar chart form the two dictionaries of the documents being tested
from Bar_Chart_compair_two_dics_fn import Barchart_compair_two_dics
Barchart_compair_two_dics(Doc_A_percent_of_tag,Doc_B_percent_of_tag)

#Sentence with the same Noun Verb combination in
from Noun_Verb_search_and_record_fn import Noun_verb_search_and_record
Saved_Noun_Verb_sentence_path=
"C:\\Anaconda3\\Text_files\\recording_noun_verb_sentence_pairs.txt"
Noun_verb_pair_sentences_pairs, number_of_noun_verb_sentences =
Noun_verb_search_and_record(Doc_A_path,Doc_B_path,Saved_Noun_Verb_sentence_path)
print("Noun Verb Sentences analysis completed and results saved to file")
print("Number of sentences that contain the same Nouns and Verbs
=",number_of_noun_verb_sentences)

from Comparing_identifiec_company_words_fn import Comparing_docs_for_company_words
Identified_common_company_words =
Comparing_docs_for_company_words(Doc_A_path,Doc_B_path)
print("Identified_common_company_words =",Identified_common_company_words,'\n')

from Same_tag_identifyer_fn import Same_tags_in_two_docs
dict_of_words_in_both_docs,Doc_A_word_frequency,Doc_B_word_frequency =
Same_tags_in_two_docs(Doc_A_path,Doc_B_path)
print("dict_of_words_in_both_docs =",dict_of_words_in_both_docs,'\n')

print("Program End")
```

9.4.2 A1_VERB_NOUN_SENTENCE_PAIR_FN

```
# function that extracts the sentence that have at least one noun and one verb in
#inputs are a text file of tagged
#Outputs A list of sentences with at least one Noun and one Verb
    # A list of noun verb sentences with nouns and verbs behind them

def A1_VB_NN(file_path):
    #print("Importing needed packages")
    import nltk
    from nltk.tokenize import word_tokenize
    from nltk.tag import pos_tag
    from Reading_and_tagging_files  import Read_Textfile_andtag

    #print("Import completed")

    token_words_full, word_list = Read_Textfile_andtag(file_path,'r')

    #print('token_words_full =', token_words_full)

    length_token_words_full=len(token_words_full)
    #print('length of word list', length_token_words_full)

    #Noun verb pair list to put the lines with noun verb pairs in
    noun_verb_pair_line=[]
    noun_verb_pair_line_noun_and_verb=[]
    #Constent defornintion
    nouncount=0
    verbcount=0

    #tag type list declarations
    Noun_common_list=[]
    Noun_propper_singular_list=[]
    Noun_proper_plural_list=[]
    Noun_common_plural_list=[]
    noun_list=[]
    Verb_base_list=[]
    Verb_past_tense_list=[]
    Verb_present_participle_list=[]
    Verb_past_participle_list=[]
    Verb_present_tense_not_3rd_person_list=[]
    Verb_present_tense_person_singular_list=[]
    not_counted_word=0
    Noun_list=[]
    Verb_list=[]

    WLWL=0
    while WLWL<length_token_words_full:
        #print("length of line",WLWL,"=",len(token_words_full[WLWL]))
        line_under_test_length=len(token_words_full[WLWL])
        line_under_test =token_words_full[WLWL]
        #print("sentence",WLWL,"is",line_under_test)
        IWLWL=0
        while IWLWL<line_under_test_length:
            #print("word number",IWLWL,"word",line_under_test[IWLWL])

            if (line_under_test[IWLWL][1])=="NN":
                        Noun_common_list.append(line_under_test[IWLWL][0])
                        #print("Noun,common,singular list=",Noun_common_list)
                        Noun_list.append(line_under_test[IWLWL][0])
                        nouncount+=1
            elif  (line_under_test[IWLWL][1])=="NNP":
```

```
Noun_propper_singular_list.append(line_under_test[IWLWL][0])
                        #print("Noun,proper,singular
list=",Noun_propper_singular_list)
                        Noun_list.append(line_under_test[IWLWL][0])
                        nouncount+=1
            elif  (line_under_test[IWLWL][1])=="NNPS":
                        Noun_proper_plural_list.append(line_under_test[IWLWL][0])
                        #print("Noun,proper,plural list=",Noun_proper_plural_list)
                        Noun_list.append(line_under_test[IWLWL][0])
                        nouncount+=1
            elif  (line_under_test[IWLWL][1])=="NNS":
                        Noun_common_plural_list.append(line_under_test[IWLWL][0])
                        #print("Noun, common, plural,
list=",Noun_common_plural_list)
                        Noun_list.append(line_under_test[IWLWL][0])
                        nouncount+=1
            elif  (line_under_test[IWLWL][1])=="VB":
                        Verb_base_list.append(line_under_test[IWLWL][0])
                        #print("Verb_base_list=",Verb_base_list)
                        Verb_list.append(line_under_test[IWLWL][0])
                        verbcount+=1
            elif  (line_under_test[IWLWL][1])=="VBD":
                        Verb_past_tense_list.append(line_under_test[IWLWL][0])
                        #print("Verb_past_tense_list=",Verb_past_tense_list)
                        Verb_list.append(line_under_test[IWLWL][0])
                        verbcount+=1
            elif  (line_under_test[IWLWL][1])=="VBG":

Verb_present_participle_list.append(line_under_test[IWLWL][0])

#print("Verb_present_participle_list=",Verb_present_participle_list)
                        Verb_list.append(line_under_test[IWLWL][0])
                        verbcount+=1
            elif  (line_under_test[IWLWL][1])=="VBN":
                        Verb_past_participle_list.append(line_under_test[IWLWL][0])

#print("Verb_past_participle_list=",Verb_past_participle_list)
                        Verb_list.append(line_under_test[IWLWL][0])
                        verbcount+=1
            elif  (line_under_test[IWLWL][1])=="VBP":

Verb_present_tense_not_3rd_person_list.append(line_under_test[IWLWL][0])

#print("Verb_present_tense_not_3rd_person_list=",Verb_present_tense_not_3rd_person
_list)
                        Verb_list.append(line_under_test[IWLWL][0])
                        verbcount+=1
            elif  (line_under_test[IWLWL][1])=="VBZ":

Verb_present_tense_person_singular_list.append(line_under_test[IWLWL][0])

#print("Verb_present_tense_person_singular_list=",Verb_present_tense_person_singul
ar_list)
                        Verb_list.append(line_under_test[IWLWL][0])
                        verbcount+=1
            else:not_counted_word+=1
                        #print('Not interesting in the word in question','\n')
            IWLWL+=1

        if nouncount>=1 and verbcount>=1:
            #print('\n'"Noun verb lines",line_under_test)
```

```
            noun_verb_pair_line.append(line_under_test)
            #print('noun_verb_pair_line', noun_verb_pair_line)
            noun_verb_pair_line_noun_and_verb.append(line_under_test)
            noun_verb_pair_line_noun_and_verb.append(Noun_list)
            noun_verb_pair_line_noun_and_verb.append(Verb_list)

        #Constant definition
        nouncount=0
        verbcount=0
        Noun_common_list=[]
        Noun_propper_singular_list=[]
        Noun_proper_plural_list=[]
        Noun_common_plural_list=[]
        noun_list=[]
        Verb_base_list=[]
        Verb_past_tense_list=[]
        Verb_present_participle_list=[]
        Verb_past_participle_list=[]
        Verb_present_tense_not_3rd_person_list=[]
        Verb_present_tense_person_singular_list=[]
        not_counted_word=0
        Noun_list=[]
        Verb_list=[]

        WLWL+=1

    #print('\n','\n','noun_verb_pair_line =', noun_verb_pair_line)
    #print('\n','noun_verb_pair_line_noun_and_verb
=',noun_verb_pair_line_noun_and_verb)


#length_noun_verb_pair_line_noun_and_verb=len(noun_verb_pair_line_noun_and_verb)
    #print('length of noun_verb_pair_line_noun_and_verb
list',length_noun_verb_pair_line_noun_and_verb)

    '''
    LNVPLZ=len(noun_verb_pair_line_noun_and_verb[0])
    print('line zero',noun_verb_pair_line_noun_and_verb[0])
    print('length of line zero',LNVPLZ)
    LNVPLO=len(noun_verb_pair_line_noun_and_verb[1])
    print('line zero',noun_verb_pair_line_noun_and_verb[1])
    print('length of line one',LNVPLO)
    LNVPLT=len(noun_verb_pair_line_noun_and_verb[2])
    print('line zero',noun_verb_pair_line_noun_and_verb[2])
    print('length of line two',LNVPLT)
    LNVPLTH=len(noun_verb_pair_line_noun_and_verb[3])
    print('line zero',noun_verb_pair_line_noun_and_verb[3])
    print('length of line two',LNVPLTH)
    '''
    return (noun_verb_pair_line, noun_verb_pair_line_noun_and_verb)
```

9.4.3 ACTUAL_DIF

```
#Function for calculating actual (modular) difference between two data sets
#This function may not give the actual difference but rather a number very close
# Inputs    a = 0.5
#           b = 0.6
# Output    dif = 0.09999999999999998
#The output is close to 0.1 this is due to floating point arithmetic

def Actual_mod_Dif(Data_A, Data_B):
```

```
    Diff=Data_A - Data_B
    Diff=abs(Diff)
    return(Diff)
```

9.4.4 BAR_CHART_COMPAIR_TWO_DICS

```
#To produce a bar chart form the two dictionaries of the documents being tested
import numpy as np
import matplotlib.pyplot as pyplot #note that the pyplot is name for plotting
#For loop that pulls the values from Doc_A_percent_of_tag and Doc_B_percent_of_tag
and combines into one dict

def Barchart_compair_two_dics(Doc_A_percent_of_tag,Doc_B_percent_of_tag):
    actual_dif_dict={}
    for key in Doc_A_percent_of_tag and Doc_B_percent_of_tag:
        #print("Doc A key =",key,"doc A percent of tag
=",Doc_A_percent_of_tag[key])
        #print("Doc B key =",key,"doc B percent of tag
=",Doc_B_percent_of_tag[key])
        actual_dif_dict[key]=[Doc_A_percent_of_tag[key],  Doc_B_percent_of_tag[key]]
        #print("actual_dif_dict =",actual_dif_dict,'\n')

    #Declare variables and their types needed for the plot
    Dictkey=[]
    Doc_A_percent_of_tags=[]
    Doc_B_percent_of_tags=[]
    number_of_keys=0
    xtick=[]
    xtick2=[]

    #Loop to create a list of the keys used in the actual_dif_dict
    for keys in actual_dif_dict:
        #print("dict key =", keys)
        Dictkey.append(keys)
        Doc_A_percent_of_tags.append(Doc_A_percent_of_tag[keys])
        #print("Doc_A_percent_of_tags key",Doc_A_percent_of_tag[keys],'\n')
        Doc_B_percent_of_tags.append(Doc_B_percent_of_tag[keys])

    #Find the number of tags in the dict
    length_of_tags=len(Doc_B_percent_of_tag)
    #Produce an array called index which starts at 0 and ends and the number of
dict keys
    index= np.arange(length_of_tags)
    #Set the with of the bar in the barchart
    bar_width = 0.35
    #Set the opacity of the bar colour
    opacity = 0.4

    #Define the y values for the bar chart
    rectsA = pyplot.bar(index, Doc_A_percent_of_tag.values(), bar_width,
                    alpha=opacity,
                    color='b',
                    label='Doc_A')

    rectsB = pyplot.bar(index + bar_width, Doc_B_percent_of_tag.values(),
bar_width,
                    alpha=opacity,
                    color='r',
                    label='Doc_B')

    #Lables and tiltels
    pyplot.xlabel('Tags')
```

```
    pyplot.ylabel('Tag percentages')
    pyplot.title('Comparison of tag percentages')

pyplot.xticks(range(len(Doc_A_percent_of_tag)),list(Doc_A_percent_of_tag.keys()),r
otation=90)
    pyplot.legend(loc=0,title='Data files under test')
    pyplot.tight_layout()
    pyplot.show()
    return()
```

9.4.5 COMMON_IDENTIFIED_WORDS

```
#Function to identify identical words from two lists
#Inputs two lists of identified words
#Output a list of words that appear in both input lists

def common_words_from_lists(identified_list_A,identified_list_B):

    length_identified_list_A = len(identified_list_A)
    length_identified_list_B = len(identified_list_B)
    common_identified_words=[]

    identified_list_WLA=0
    while identified_list_WLA<length_identified_list_A:
        identified_list_WLB=0
        while identified_list_WLB<length_identified_list_B:
            if
identified_list_A[identified_list_WLA]==identified_list_B[identified_list_WLB]:

common_identified_words.append(identified_list_B[identified_list_WLB])
            identified_list_WLB+=1
        identified_list_WLA+=1

    #print("common identified words",common_identified_words)
    return(common_identified_words)
```

9.4.6 COMPANEY_DICTONARY

```
#Company specific terms to search for
#A function that returns key integration related words
#Input is the file path of the text file to be analysed

def Companey_dict(file_path):

    #print("company specific terms",'\n')
    from Reading_and_tagging_files import Read_Textfile_andtag
    #print("Imported functions")

    #tag first then data

    #Development environment dictionary

dev_environment_list=["labview","Labview","LabVIEW",["lab","view"],["Lab","View"],
["Lab","VIEW"],
                          "matlab","Matlab",["Mat","lab"],["Mat","Lab"],"CarMaker",

"carmaker","Carmaker",["Car","Maker"],["car","maker"],"Abaqus","abaqus",

"ABAQUS","axisuite","Axisuite",["axi","suite"],"Simulink","simulink",
                          "IPG
CarMaker",["IPG","Car","Maker"],["IPG","car","maker"],["IPG","carmaker"],

["ipg","carmaker"],"Dymola","dymola","AMESim","amesim",["Simulation","X"],

["simulaiton","x"],["simulaiton","X"],"SimulationX","simulationx"

"GT","gt","SIMPACK","simpack","AVL","avl",["Python","2"],"Python"]
    #print("dev environment list",dev_environment_list,'\n')

    #model terms

modeling_term_list=["input","inputs","output","outputs","interface","architecture"
,"modelled","GB","Gb",
```

```
"libary","libraries","saved","behaviour","Behaviour","format","Format",["time","st
eps"],
                        "GHz","ghz",["quad", "core"],"Intel","Arm",["Core",
"i7"],["Core","i5"], "processor"]
    #print("modeling_term_list",modeling_term_list,'\n')

    #project terms
    project_term_list=["kick off","KO",["start","date"],"validated","verified"]
    #print("project term list",project_term_list,'\n')

    #langagues
    prog_language_list=["C","C++","C#","G","g","java","M"]
    #print("prog language list",prog_language_list,'\n')

    #file type

file_types_list=[["M","file"],["m","file"],"mat","mdl","txt","csv","jpg","prj","xl
s","doc","docx","htm",
                     "pdf","m50",["m","50"],"c","C","ppt","pptx","exe","dll","py"]
    #print("file types list",file_types_list,'\n')

    #Calling file to be analised
    token_words_full, word_list = Read_Textfile_andtag(file_path,'r')

    #Lenghts of company specific turms
    lenth_word_list=len(word_list)
    lenth_dev_environment_list=len(dev_environment_list)
    lenth_modeling_term_list=len(modeling_term_list)
    lenth_project_term_list=len(project_term_list)
    lenth_prog_language_list=len(prog_language_list)
    lenth_file_types_list=len(file_types_list)

    #blank lists for identified words
    identified_dev_enviroments=[]
    identified_modeling_tuerms=[]
    identified_project_tuerms=[]
    identified_prog_languages=[]
    identified_file_types=[]

    word_list_WLC=0
    while word_list_WLC<lenth_word_list:

        line_under_test=word_list[word_list_WLC]
        lenth_of_line=len(line_under_test)
        #print('length of line under test =',lenth_of_line)
        #print("line_under_test =", line_under_test)

        word_in_line_WLC=0
        while word_in_line_WLC<lenth_of_line:
            word_under_test=line_under_test[word_in_line_WLC]
            #print("word under test =",word_under_test)
            word_in_line_WLC+=1

            dev_enviroment_list_WLC=0
            while dev_enviroment_list_WLC<lenth_dev_environment_list:
                #print("word under test =",word_under_test)
                #print("dev_enviromen word =",dev_environment_list[dict_list_WLC])
                if word_under_test ==
dev_environment_list[dev_enviroment_list_WLC]:
                    identified_dev_enviroments.append(word_under_test)
```

```
                    #print("Same words identified.","\n" "word under test
=",word_under_test,
                            #"dev environment word
=",dev_environment_list[dev_enviroment_list_WLC],
                            #"\n","\n")
                dev_enviroment_list_WLC+=1

            modeling_term_list_WLC=0
            while modeling_term_list_WLC<lenth_modeling_term_list:
                #print("modeling term word
=",modeling_term_list[modeling_term_list_WLC])
                if word_under_test == modeling_term_list[modeling_term_list_WLC]:
                    identified_modeling_tuerms.append(word_under_test)
                modeling_term_list_WLC+=1

            project_term_list_WLC=0
            while project_term_list_WLC<lenth_project_term_list:
                #print("project_term_list
=",project_term_list[project_term_list_WLC])
                if word_under_test == project_term_list[project_term_list_WLC]:
                    identified_project_tuerms.append(word_under_test)
                project_term_list_WLC+=1

            prog_language_list_WLC=0
            while prog_language_list_WLC<lenth_prog_language_list:
                #print("prog_language_list
=",prog_language_list[prog_language_list_WLC])
                if word_under_test == prog_language_list[prog_language_list_WLC]:
                    identified_prog_languages.append(word_under_test)
                prog_language_list_WLC+=1

            file_types_list_WLC=0
            while file_types_list_WLC<lenth_file_types_list:
                #print("file_types_list =", file_types_list[file_types_list_WLC])
                if word_under_test == file_types_list[file_types_list_WLC]:
                    identified_file_types.append(word_under_test)
                file_types_list_WLC+=1


        word_list_WLC+=1

    #print("Identified_dev_enviroments =", identified_dev_enviroments)
    #print("identified_modeling_tuerms =", identified_modeling_tuerms)
    #print("identified_project_tuerms =", identified_project_tuerms)
    #print("identified_prog_languages =", identified_prog_languages)
    #print("identified_file_types =", identified_file_types)

    return
(identified_dev_enviroments,identified_modeling_tuerms,identified_project_tuerms,
            identified_prog_languages,identified_file_types)
```

9.4.7 COMPARING_IDENTIFIEC_COMPANY_WORDS

```
#Comparing_identifiec_company_words_fn
#The purpose of this function is to compare if two text files have the same words
identified words
#in them and what the words are.

def Comparing_docs_for_company_words(Doc_A_path,Doc_B_path):
    Identified_common_company_words ={}

    from Companey_dictonary_fn import Companey_dict
```

```
    identified_dev_enviroments_A, identified_modeling_tuerms_A,
identified_project_tuerms_A, identified_prog_languages_A, identified_file_types_A
= Companey_dict(Doc_A_path)

identified_dev_enviroments_B,identified_modeling_tuerms_B,identified_project_tuerm
s_B,  identified_prog_languages_B,identified_file_types_B=  Companey_dict(Doc_B_path)
    #print("DEBUGGING     Identified development
enviroments",'\n',"identified_dev_enviroments_A",identified_dev_enviroments_A)
          #,'\n',"identified_dev_enviroments_B",identified_dev_enviroments_B)

    from common_identified_words_fn import common_words_from_lists
    common_dev_enviroments =
common_words_from_lists(identified_dev_enviroments_A,identified_dev_enviroments_B)
    number_of_common_dev_enviroments=len(common_dev_enviroments)
    print("common_dev_enviroments =",common_dev_enviroments,'\n','number of words
=',number_of_common_dev_enviroments)

Identified_common_company_words['common_dev_enviroments']=common_dev_enviroments

Identified_common_company_words['number_of_common_dev_enviroments']=number_of_comm
on_dev_enviroments

    common_identified_modeling_tuerms =
common_words_from_lists(identified_modeling_tuerms_A,identified_modeling_tuerms_B)
    number_of_common_identified_modeling_tuerms =
len(common_identified_modeling_tuerms)
    print("common_identified_modeling_tuerms
=",common_identified_modeling_tuerms,'\n','number of
common_identified_modeling_tuerms =',number_of_common_identified_modeling_tuerms)

Identified_common_company_words['common_identified_modeling_tuerms']=common_identi
fied_modeling_tuerms

Identified_common_company_words['number_of_common_identified_modeling_tuerms']=num
ber_of_common_identified_modeling_tuerms

    common_identified_project_tuerms =
common_words_from_lists(identified_project_tuerms_A,identified_project_tuerms_B)
    number_of_common_identified_project_tuerms =
len(common_identified_project_tuerms)
    print("common_identified_project_tuerms =",
common_identified_project_tuerms,'\n','number of common_identified_project_tuerms
=',number_of_common_identified_modeling_tuerms)

Identified_common_company_words['common_identified_project_tuerms']=common_identif
ied_project_tuerms

Identified_common_company_words['number_of_common_identified_project_tuerms']=numb
er_of_common_identified_project_tuerms

    common_identified_prog_languages =
common_words_from_lists(identified_prog_languages_A,identified_prog_languages_B)
    number_of_identified_prog_languages = len(common_identified_prog_languages)
    print("common_identified_prog_languages
=",common_identified_prog_languages,'\n','number of
common_identified_prog_languages =',number_of_identified_prog_languages)

Identified_common_company_words['common_identified_prog_languages']=common_identif
ied_prog_languages

Identified_common_company_words['number_of_identified_prog_languages']=number_of_i
dentified_prog_languages
```

```
    common_identified_file_types =
common_words_from_lists(identified_file_types_A,identified_file_types_B)
    number_of_identified_file_types = len(common_identified_file_types)
    print("common_identified_file_types
=",common_identified_file_types,'\n','number of common_identified_file_types
=',number_of_identified_file_types)

Identified_common_company_words['common_identified_file_types']=common_identified_
file_types

Identified_common_company_words['number_of_identified_file_types']=number_of_ident
ified_file_types

    return (Identified_common_company_words)
```

9.4.8 NOUN_VERB_SEARCH_AND_RECORD

```
#A function to compare the sentences in two files for similarities
#Function searches for sentences with both nouns and verbs in.
#If both documents have the a sentence with the same nouns and verbs in they are
recorded in a text file

def Noun_verb_search_and_record( file_path_Doc_A,
file_path_Doc_B,Noun_verb_sentence_pair_textfile ):
    #print('\n','\n'," Starting Noun Verb Search and record function")

    #Loading the functions needed for A1 algorithm
    from A1_Verb_Noun_sentence_pair_fn import A1_VB_NN
    #print("calling A1")
    #print("functions loaded")

    #Use A1_VB_NN algorithm and set up the output variables for file A input file
location for file A
    FA_line , FA_line_VB_NN = A1_VB_NN(file_path_Doc_A)
    #print("FA_Line", FA_line,'\n',"FA_line_VB_NN", FA_line_VB_NN)
    #print('File A Verb Noun lines =',FA_line_VB_NN,'\n')

    #C:\Python34\Documentation_of_modelling_the_movement_of_a_ball.txt
    #Use A1_VB_NN algorithm and set up the output variables for file B input file
location for file B
    FB_line, FB_line_VB_NN = A1_VB_NN(file_path_Doc_B)
    #print("FB_Line", FB_line,'\n',"FB_line_VB_NN", FB_line_VB_NN)
    #print('File B Verb Noun lines =',FB_line_VB_NN,'\n')

    #Verification section of code collapsed

    #verification exercise takes the file A tagged data set and saves to a text
file
    FA_save_textfile = open("C:\\Anaconda3\\Text_files\\Document_A_tagged.txt","w")
    FA_save_textfile.write(str(FA_line))
    #FA_save_textfile.write(str(FA_line_VB_NN))
    FA_save_textfile.flush
    FA_save_textfile.close()
    #print("File A Noun Verb Sentences successfully saved to text file")

    #verification exercise takes the file B tagged data set and saves to a text
file
    FB_textfile=open("C:\Anaconda3\\Text_files\\Document_B_tagged.txt","w")
    FB_textfile.write(str(FB_line))
    #FB_textfile.write(str(FB_line_VB_NN))
```

```
    FB_textfile.flush
    FB_textfile.close()
    #print("File B Noun Verb Sentences successfully saved to text file")


    #File A verb list
    FA_VB=[]
    #File A Noun list
    FA_NN=[]
    #File B verb list
    FB_VB=[]
    #File B Noun list
    FB_NN=[]
    #Noun verb pair sentences pairs
    Noun_verb_pair_sentences_pairs=[]

    #Define the length of File A Verb noun line
    length_FA_VB = len(FA_line_VB_NN)
    FAWL=1
    while FAWL<length_FA_VB:
        FA_NN.append(FA_line_VB_NN[FAWL])
        FA_VB.append(FA_line_VB_NN[FAWL+1])
        FAWL+=3

    #Define the length of File B Verb noun line
    length_FB_VB= len(FB_line_VB_NN)
    FBWL=1
    while FBWL<length_FB_VB:
        FB_NN.append(FB_line_VB_NN[FBWL])
        FB_VB.append(FB_line_VB_NN[FBWL+1])
        FBWL+=3

    length_FA_VB= len(FA_VB)
    #print("length FA_VB=", length_FA_VB)
    FAVB=0
    while FAVB< length_FA_VB:
        #print('\n','\n',"line number =",FAVB)
        length_Verb_set=len(FA_VB[FAVB])
        #print("length of Verb set =",length_Verb_set)
        length_Noun_set=len(FA_NN[FAVB])
        #print("length of Noun set=",length_Noun_set)
        FAVB+=1

    #print('FA_NN =',FA_NN,'\n')
    #print('FA_VB =',FA_VB,'\n')
    #print('FB_NN =',FB_NN,'\n')
    #print('FB_VB =',FB_VB,'\n')

    #variables for comparison section of code
    same_verbs=[]
    same_nouns=[]
    noun_verb_pair=[]

    #While loop iterates through the two tagged text lines
        #loops are in the following structure with nested loops
            #File A Noun loop for length of file A noun set
            #File B Noun loop length of file B noun set
            #if file A noun is the same as file B noun move to the next loop
            #File A Verb loop for the length of file A Verb set
            #File B Verb loop for the length of file B Verb set
            #if the file A verb matches the file B verb then it is a noun verb
phrase match
```

```
            #the matching noun verb phrases are then written to file


            #Temporary line to see if the while loop will work
    curent_noun_verb_sentences=[0]
    #while loop for file A Noun
    wlfaN=0
    identified_lines=0
    length_File_A_noun= len(FA_NN)
    #print("length of file a noun list =",length_File_A_noun)
    while wlfaN<length_File_A_noun:
        identified_lines=0
        wlfaN+=1
        file_A_nouns_under_test=FA_NN[wlfaN-1]
        length_FA_NN_wlfaN=len(FA_NN[wlfaN-1])
        #print('Length of file A noun under test FA_NN[wlfaN]
='',length_FA_NN_wlfaN)
        #print("file A noun",wlfaN,"under test =",file_A_nouns_under_test)
        #print("while loop file a Noun count =",wlfaN-1)

        #while loop for if there is more than one Noun in a sentence
        #while loop file a noun multiple nouns set to zero
        wlfaNmN=0
        while wlfaNmN<length_FA_NN_wlfaN:
            wlfaNmN+=1
            file_A_noun_under_test=file_A_nouns_under_test[wlfaNmN-1]
            #print("file_A_noun_under_test",file_A_noun_under_test,'\n')

            #While loop for file B Noun sets
            wlfbN=0
            length_File_B_Noun= len(FB_NN)
            #print("length file b noun =",length_File_B_Noun)
            noun_same_count=0

            while wlfbN < length_File_B_Noun:
                wlfbN+=1
                file_B_nouns_under_test=FB_NN[wlfbN-1]
                #print("while loop file b nouns =",wlfbN-1)
                length_FB_NN_wlfbN=len(FB_NN[wlfbN-1])
                #print('Length of file B noun under test FB_NN[wlfbN]
='',length_FB_NN_wlfbN)
                #print("file B noun",wlfbN-1, "under test
="",file_B_nouns_under_test)
                wlfbNmN=0

                #While loop for if there are more than one Nouns in a sentence
                while wlfbNmN < length_FB_NN_wlfbN:
                    wlfbNmN+=1
                    file_B_noun_under_test=file_B_nouns_under_test[wlfbNmN-1]
                    #print("file_B_noun_under_test",file_B_noun_under_test,'\n')

                    #If statement to act if the file A identified noun is the same
and the Identified B noun
                    if file_A_noun_under_test==file_B_noun_under_test:
                        #print("same noun detected now searching verb searching
commencing")
                        same_nouns.append(file_A_noun_under_test)
                        noun_same_count+=1

                        #While loop for file A Verb
                        wlfaV=0
                        length_File_A_Verb=len(FA_VB)
```

```
                        #print("lenfth file A verb list =",length_File_A_Verb)
                        while wlfaV<length_File_A_Verb:
                            file_A_verbs_under_test=FA_VB[wlfaV]
                            wlfaV+=1
                            #print("file a verbs under
test",file_A_verbs_under_test)
                            #print("while loop file a verb =",wlfaV)
                            length_FA_VB_wlfaV=len(FA_VB[wlfaV-1])
                            #print('Length file A wlfaV verb =',length_FA_VB_wlfaV)

                            #while loop for if there are more than one Verb in the
file A sentence
                            wlfaVbN=0
                            while wlfaVbN <length_FA_VB_wlfaV:
                                wlfaVbN+=1

file_A_verb_under_test=file_A_verbs_under_test[wlfaVbN-1]
                                #print('file A Verb under test
=',file_A_verb_under_test)

                                #while loop for file B Verb
                                length_File_B_Verb=len(FB_VB)
                                #print("length file B verb list
=",length_File_B_Verb)
                                wlfbV=0
                                while wlfbV<length_File_B_Verb:
                                    file_B_verbs_under_test=FB_VB[wlfbV]
                                    wlfbV+=1
                                    #print("while loop file b verb=", wlfbV)
                                    #print("file b verb under
test",file_B_verbs_under_test)
                                    length_FB_VB_wlfbV=len(FB_VB[wlfbV-1])
                                    #while loop for if there are more than on Verb
in the file B sentence
                                    wlfbVbN=0
                                    while wlfbVbN < length_FB_VB_wlfbV:
                                        wlfbVbN+=1

file_B_verb_under_test=file_B_verbs_under_test[wlfbVbN-1]
                                        #print('File B Verb under test
=',file_B_verb_under_test)

                                        if identified_lines == 0:
                                            if
file_A_verb_under_test==file_B_verb_under_test:
                                                #print("Noun and Verb the same in
both sentences detected!")

same_verbs.append(file_A_verb_under_test)

noun_verb_pair.append([file_A_noun_under_test,file_B_verb_under_test])
                                                #print("noun verb pair
detected",[file_A_noun_under_test,file_B_verb_under_test])
                                                #print("line from file A
=",FA_line[wlfaN-1])
                                                #print("line from file B
=",FB_line[wlfbN-1])
                                                curent_noun_verb_sentences=[]

#print('\n','\n',"curent_noun_verb_sentences",curent_noun_verb_sentences)

curent_noun_verb_sentences.append(FA_line[wlfaN-1])
```

```
#print('\n',"curent_noun_verb_sentences FA_line[wlfaN-
1]",curent_noun_verb_sentences)

curent_noun_verb_sentences.append(FB_line[wlfbN-1])

#print('\n',"curent_noun_verb_sentences FB_line[wlfbN-
1]",curent_noun_verb_sentences)

Noun_verb_pair_sentences_pairs.append(curent_noun_verb_sentences)
                                                    #print
("Noun_verb_pair_sentences_pairs",Noun_verb_pair_sentences_pairs)
                                                    identified_lines=1


    #Interrogation of identified Noun_verb_pair_sentences_pairs set
    #Length of Noun_verb_pair_sentences_pairs
    #print('Noun_verb_pair_sentences_pairs =',Noun_verb_pair_sentences_pairs)
    length_noun_verb_pair_sentences_pairs=len(Noun_verb_pair_sentences_pairs)
    #print("Number of Noun_verb_pair_sentences_pairs
=",length_noun_verb_pair_sentences_pairs)
    #print("noun verb sentence pairs",Noun_verb_pair_sentences_pairs,'\n','\n')

    #Interrogation of curent_noun_verb_sentences set
    #curent_noun_verb_sentences
    #print('curent_noun_verb_sentences',curent_noun_verb_sentences)
    length_curent_noun_verb_sentences=len(curent_noun_verb_sentences)
    #print("length or curent_noun_verb_sentences set
=",length_curent_noun_verb_sentences)
    #print("curent noun verb sentences =",curent_noun_verb_sentences)

    #Create file and wright context of identified noun_verb_pair_sentences_pairs
    #print("writing to recording_noun_verb_sentence_pairs text file")
    save_textfile=open(Noun_verb_sentence_pair_textfile,"w")
    save_textfile.write(str(Noun_verb_pair_sentences_pairs))
    save_textfile.flush
    save_textfile.close()
    #print("noun Verb Sentences sucsesfully saved to text file")

    #print('\n','\n',"End of noun verb search and record function",'\n','\n')
    return (Noun_verb_pair_sentences_pairs,length_noun_verb_pair_sentences_pairs)
```

9.4.9 NUMBER_OF_TIMES_A_WORD_APPEARS

```
#This function is for the purpose of asserting how many times the identical words
#appear in two documents
#Inputs:
    #dict_of_words_in_both_docs, A dictionary of words that appear in two
documents
    #Doc_A_taglists, A dictionary of all identified words that appear in one
document

#outputs:
    #word_frequency, A dictionary of words, A dictionary that has a word and the
number of times that word appears

word_frequency={}
def Number_of_words_in_one_doc_in_both(dict_of_words_in_both_docs,Doc_A_taglists):

    for word_tags,Identical_words in dict_of_words_in_both_docs.items():
        #print("dict_of_words_in_both_docs word_tags",word_tags)
```

```
        #print("dict_of_words_in_both_docs Identical_words",Identical_words,'\n')
        #Create an empty list for the current word_tags
        tag_under_test=[]
        #Right word tags to the empty list
        tag_under_test = word_tags
        #Create an empty list for the common_words
        Identified_common_words=[]
        #Current words associated with the current word tag written to
Identified_common_words
        Identified_common_words = Identical_words
        #Find the number of words in Identified_common_words
        length_Identified_common_words=len(Identified_common_words)
        #Raw taglist words for current tags brought in
        Doc_A_taged_list=Doc_A_taglists[word_tags]
        #Find the number of words in Dpc_A_taged_list
        length_Doc_A_taged_list=len(Doc_A_taged_list)
        word_frequency[word_tags]=[]
        #Set up a while loop counter for the number of same words
        WLC_number_of_same_words=0
        #while the loop counter is less than the number of identified common words
iterate
        while WLC_number_of_same_words < length_Identified_common_words:
            #count the number of times the specific word appears in the word list
under test

number_of_word=Doc_A_taged_list.count(Identified_common_words[WLC_number_of_same_w
ords])
            #if there is more than one instance of a word then do the thing
            if number_of_word >1:
                #for the current tag and word rite number of times the word
appears

word_frequency[word_tags].append([number_of_word,Identified_common_words[WLC_numbe
r_of_same_words]])
            #If the number of times that the word appears is only once then
            elif number_of_word==1:
                #for the current tag and word rite number of times the word
appears

word_frequency[word_tags].append([1,Identified_common_words[WLC_number_of_same_wor
ds]])

            #Increment the while loop
            WLC_number_of_same_words+=1

    return(word_frequency)
```

9.4.10 SAME_TAG_IDENTIFYER

```
#A function to identify same words in two tag sets
#Take file paths
#Tag the files with Text_charactaristics_fn
#Return the two tag lists
#Compare the two lists and search the tag sets to see if there are the same words
#Return numbers of same words
######Return percentages of similar words###### still need to do this

def Same_tags_in_two_docs(Doc_A_path,Doc_B_path):

    #import the text characteristics function
    from Text_charactaristics_fn import Text_charactaristics
```

```
    Doc_A_token_words_full, Doc_A_word_list, Doc_A_tagcount, Doc_A_taglists,
Doc_A_number_of_lines, Doc_A_NB_tags, Doc_A_NB_words, Doc_A_percent_of_tag,
Doc_A_percent_of_words = Text_charactaristics(Doc_A_path)
    Doc_B_token_words_full, Doc_B_word_list, Doc_B_tagcount, Doc_B_taglists,
Doc_B_number_of_lines, Doc_B_NB_tags, Doc_B_NB_words, Doc_B_percent_of_tag,
Doc_B_percent_of_words = Text_charactaristics(Doc_B_path)

    #print("Doc_A_taglists",Doc_A_taglists,'\n',"Doc_B_taglists",Doc_B_taglists)

    length_Doc_A_taglists = len(Doc_A_taglists)
    length_Doc_B_taglists = len(Doc_B_taglists)
    #print("length_Doc_A_taglists =",length_Doc_A_taglists)
    #print("length_Doc_B_taglists =",length_Doc_B_taglists)

    Same_identified_words={}
    #print("type of Doc_A_taglists =", type(Doc_A_taglists))
    #taglist is a type Dict so need to call the keys and then treat as lists

    for key, value in Doc_A_taglists.items():
        #print("dict keys =",key, '\n',"Doc_A_taglists[key] =",
Doc_A_taglists[key],'\n',"Doc_B_taglists[key]",Doc_B_taglists[key],'\n','\n')
        A_list=Doc_A_taglists[key]
        B_list=Doc_B_taglists[key]
        length_A_list = len(A_list)
        length_B_list = len(B_list)
        same_words_list=[]
        WLC_step_through_Doc_A_taglists=0
        while WLC_step_through_Doc_A_taglists < length_A_list:
            WLC_step_through_Doc_B_taglists=0
            while WLC_step_through_Doc_B_taglists < length_B_list:
                if
A_list[WLC_step_through_Doc_A_taglists]==B_list[WLC_step_through_Doc_B_taglists]:
                    same_words_list.append(A_list[WLC_step_through_Doc_A_taglists])
                    #print("same word
identified",A_list[WLC_step_through_Doc_A_taglists])
                #print("WLC_step_through_Doc_B_taglists
=",WLC_step_through_Doc_B_taglists)
                WLC_step_through_Doc_B_taglists+=1
            #print("WLC_step_through_Doc_A_taglists
=",WLC_step_through_Doc_A_taglists,'\n','\n')
            WLC_step_through_Doc_A_taglists+=1
        Same_identified_words[key]=same_words_list
        #print("same_words_list", same_words_list)

    #print("Same_identified_words =",Same_identified_words,'\n','\n')
    #print("first section completed")


    #This section creates a dictionary of all of only one instance of each of the
similar words
    #The tags of the words are preserved
    dict_of_words_in_both_docs={}
    #For loop to step through the tags in the dictionary "Same_identified_words"
    for tags, values in Same_identified_words.items():
        #print("tags",tags)
        #print("values",values)
        #tempary empty list
        Same_identified_words_list=[]
        #copy in vlaues from current tags into temporary list
        Same_identified_words_list=values
        #print("Same_identified_words_list",Same_identified_words_list)
        #Sort the order of the list alphabetically
```

```
        Same_identified_words_list.sort()
        #convert the list to a set which only takes one instance of each item in
the list
        Set_of_words_in_both_docs=set(Same_identified_words_list)
        #convert the content of the set back to a list
        Same_identified_words_list_single_instance=list(Set_of_words_in_both_docs)
        #print("Same_identified_words_list_single_instance
=",Same_identified_words_list_single_instance)
        #wright the contents of the list to a dictionary

dict_of_words_in_both_docs[tags]=Same_identified_words_list_single_instance
    #print("dict_of_words_in_both_docs",dict_of_words_in_both_docs,'\n','\n')
    #print("type of dict_of_words_in_both_docs",type(dict_of_words_in_both_docs))


    #The next section of the code is for the purpose of asserting how many times
the identical words
    #appear in both documents
    #Call the function Number_of_words_in_one_doc_in_both
    from Number_of_times_a_word_Appears_fn import
Number_of_words_in_one_doc_in_both
    Doc_A_word_frequency=
Number_of_words_in_one_doc_in_both(dict_of_words_in_both_docs,Doc_A_taglists)
    Doc_B_word_frequency=
Number_of_words_in_one_doc_in_both(dict_of_words_in_both_docs,Doc_B_taglists)

    #print("word_frequency dict Doc_A =",Doc_A_word_frequency)
    #print("word_frequency dict Doc_B =",Doc_A_word_frequency)
    return(dict_of_words_in_both_docs,Doc_A_word_frequency,Doc_B_word_frequency)
```

9.4.11 TAG_COUNT

```
# Function which returns the number of different taggs of a peace of text

def Tag_count(file_path):
    #Loading reading and tagging files
    from Reading_and_tagging_files import Read_Textfile_andtag

    #call the read textfile and tag funciton
    token_words_full , word_list = Read_Textfile_andtag(file_path,'r')
    #print("token_words_full =", token_words_full)
    #print('\n','\n',"word_list =", word_list)

    #Tag lists set to zero tags are those that used in the pos tag function
    noun_common_list=[]
    noun_propper_singular_list=[]
    noun_propper_list=[]
    noun_proper_plural_list=[]
    noun_common_plural_list=[]
    verb_base_list=[]
    verb_past_tense_list=[]
    verb_present_participle_list=[]
    verb_past_participle_list=[]
    verb_present_tense_not_3rd_person_list=[]
    verb_present_tense_person_singular_list=[]
    coordinating_conjunction_list=[]
    cardinal_number_list=[]
    determiner_list=[]
    existential_there_list=[]
    foreign_word_list=[]
    preposition_subordinating_list=[]
```

```
adjective_list=[]
adjective_comparative_list=[]
adjective_superlative_list=[]
list_marker_list=[]
modal_list=[]
predeterminer_list=[]
possessive_ending_list=[]
personal_pronoun_list=[]
possessive_pronoun_list=[]
adverb_list=[]
adverb_comparative_list=[]
adverb_superlative_list=[]
particle_list=[]
to_list=[]
interjection_list=[]
wh_determiner_list=[]
wh_pronoun_list=[]
possessive_wh_pronoun_list=[]
wh_adverb_list=[]
not_tagged_word_list=[]
punctuation_list=[]
parenthesis_list=[]
unbalenced_parenthesis_list=[]

#Tag counters set to zero
#For purposes of counting the different taggs each tag has a counter
noun_common_count=0
noun_propper_count=0
noun_propper_singular_count=0
noun_proper_plural_count=0
noun_common_plural_count=0
verb_base_count=0
verb_past_tense_count=0
verb_present_participle_count=0
verb_past_participle_count=0
verb_present_tense_not_3rd_person_count=0
verb_present_tense_person_singular_count=0
coordinating_conjunction_count=0
cardinal_number_count=0
determiner_count=0
existential_there_count=0
foreign_word_count=0
preposition_subordinating_count=0
adjective_count=0
adjective_comparative_count=0
adjective_superlative_count=0
list_marker_count=0
modal_count=0
predeterminer_count=0
possessive_ending_count=0
personal_pronoun_count=0
possessive_pronoun_count=0
adverb_count=0
adverb_comparative_count=0
adverb_superlative_count=0
particle_count=0
to_count=0
interjection_count=0
wh_determiner_count=0
wh_pronoun_count=0
possessive_wh_pronoun_count=0
wh_adverb_count=0
```

```
    not_tagged_word=0
    punctuation_count=0
    parenthesis_count=0
    unbalenced_parenthesis_count=0

    #the number of lines in the text file that is read in is equal to the length
of the list that is read in
    lenth_token_word_list=len(token_words_full)
    #read down line while loop
    rdlwl=0
    while rdlwl<lenth_token_word_list:
        #print("line", rdlwl, token_words_full [rdlwl],'\n')
        line_under_test= token_words_full [rdlwl]
        length_line_under_test=len(line_under_test)

        #read line word while loop one word at a time
        rdlwlil=0
        while rdlwlil<length_line_under_test:
        #If else chain for all of the tags present in the postag tags
                    if (line_under_test[rdlwlil][1])=="NN":
                        noun_common_list.append(line_under_test[rdlwlil][0])
                        #print("Noun,common,singular list=",Noun_common_list)
                        noun_common_count+=1
                    elif  (line_under_test[rdlwlil][1])=="NNP":

noun_propper_singular_list.append(line_under_test[rdlwlil][0])
                        #print("Noun,proper,singular
list=",Noun_propper_singular_list)
                        noun_propper_count+=1
                    elif  (line_under_test[rdlwlil][1])=="NNPS":
                        noun_proper_plural_list.append(line_under_test[rdlwlil][0])
                        #print("Noun,proper,plural list=",Noun_proper_plural_list)
                        noun_proper_plural_count+=1
                    elif  (line_under_test[rdlwlil][1])=="NNS":
                        noun_common_plural_list.append(line_under_test[rdlwlil][0])
                        #print("Noun, common, plural,
list=",Noun_common_plural_list)
                        noun_common_plural_count+=1
                    elif  (line_under_test[rdlwlil][1])=="VB":
                        verb_base_list.append(line_under_test[rdlwlil][0])
                        #print("Verb_base_list=",Verb_base_list)
                        verb_base_count+=1
                    elif  (line_under_test[rdlwlil][1])=="VBD":
                        verb_past_tense_list.append(line_under_test[rdlwlil][0])
                        #print("Verb_past_tense_list=",Verb_past_tense_list)
                        verb_past_tense_count+=1
                    elif  (line_under_test[rdlwlil][1])=="VBG":

verb_present_participle_list.append(line_under_test[rdlwlil][0])

#print("Verb_present_participle_list=",Verb_present_participle_list)
                        verb_present_participle_count+=1
                    elif  (line_under_test[rdlwlil][1])=="VBN":

verb_past_participle_list.append(line_under_test[rdlwlil][0])

#print("Verb_past_participle_list=",Verb_past_participle_list)
                        verb_past_participle_count+=1
                    elif  (line_under_test[rdlwlil][1])=="VBP":

verb_present_tense_not_3rd_person_list.append(line_under_test[rdlwlil][0])
```

```
#print("Verb_present_tense_not_3rd_person_list=",Verb_present_tense_not_3rd_person
_list)
                        verb_present_tense_not_3rd_person_count+=1
                    elif  (line_under_test[rdlwlil][1])=="VBZ":

verb_present_tense_person_singular_list.append(line_under_test[rdlwlil][0])

#print("Verb_present_tense_person_singular_list=",Verb_present_tense_person_singul
ar_list)
                        verb_present_tense_person_singular_count+=1
                    elif  (line_under_test[rdlwlil][1])=="CC":

coordinating_conjunction_list.append(line_under_test[rdlwlil][0])

#print("Coordinating_conjunction_list=",Coordinating_conjunction_list)
                        coordinating_conjunction_count+=1
                    elif  (line_under_test[rdlwlil][1])=="CD":
                        cardinal_number_list.append(line_under_test[rdlwlil][0])
                        #print("Cardinal_number_list=",Cardinal_number_list)
                        cardinal_number_count+=1
                    elif  (line_under_test[rdlwlil][1])=="DT":
                        determiner_list.append(line_under_test[rdlwlil][0])
                        #print("Determiner_list=",Determiner_list)
                        determiner_count+=1
                    elif  (line_under_test[rdlwlil][1])=="EX":
                        existential_there_list.append(line_under_test[rdlwlil][0])
                        #print("Existential_there=",Existential_there)
                        existential_there_count+=1
                    elif  (line_under_test[rdlwlil][1])=="FW":
                        foreign_word_list.append(line_under_test[rdlwlil][0])
                        #print("Foreign_word_list=",Foreign_word_list)
                        foreign_word_count+=1
                    elif  (line_under_test[rdlwlil][1])=="IN":

preposition_subordinating_list.append(line_under_test[rdlwlil][0])

#print("Preposition_subordinating_list=",Preposition_subordinating_list)
                        preposition_subordinating_count+=1
                    elif  (line_under_test[rdlwlil][1])=="JJ":
                        adjective_list.append(line_under_test[rdlwlil][0])
                        #print("Adjective_list=",Adjective_list)
                        adjective_count+=1
                    elif  (line_under_test[rdlwlil][1])=="JJR":

adjective_comparative_list.append(line_under_test[rdlwlil][0])

#print("Adjective_comparative_list=",Adjective_comparative_list)
                        adjective_comparative_count+=1
                    elif  (line_under_test[rdlwlil][1])=="JJS":

adjective_superlative_list.append(line_under_test[rdlwlil][0])

#print("Adjective_superlative_list=",Adjective_superlative_list)
                        adjective_superlative_count+=1
                    elif  (line_under_test[rdlwlil][1])=="LS":
                        list_marker_list.append(line_under_test[rdlwlil][0])
                        #print("list_marker_list=",list_marker_list)
                        list_marker_count+=1
                    elif  (line_under_test[rdlwlil][1])=="MD":
                        modal_list.append(line_under_test[rdlwlil][0])
                        #print("modal_list=",modal_list)
```

```
                        modal_count+=1
                    elif  (line_under_test[rdlwlil][1])=="PDT":
                        predeterminer_list.append(line_under_test[rdlwlil][0])
                        #print("predeterminer_list=",predeterminer_list)
                        predeterminer_count+=1
                    elif  (line_under_test[rdlwlil][1])=="POS":
                        possessive_ending_list.append(line_under_test[rdlwlil][0])
                        #print("possessive_ending_list=",possessive_ending_list)
                        possessive_ending_count+=1
                    elif  (line_under_test[rdlwlil][1])=="PRP":
                        personal_pronoun_list.append(line_under_test[rdlwlil][0])
                        #print("personal_pronoun_list=",personal_pronoun_list)
                        personal_pronoun_count+=1
                    elif  (line_under_test[rdlwlil][1])=="PRP$":
                        possessive_pronoun_list.append(line_under_test[rdlwlil][0])
                        #print("possessive_pronoun_list=",possessive_pronoun_list)
                        possessive_pronoun_count+=1
                    elif  (line_under_test[rdlwlil][1])=="RB":
                        adverb_list.append(line_under_test[rdlwlil][0])
                        #print("adverb_list=",adverb_list)
                        adverb_count+=1
                    elif  (line_under_test[rdlwlil][1])=="RBR":
                        adverb_comparative_list.append(line_under_test[rdlwlil][0])
                        #print("adverb_comparative_list=",adverb_comparative_list)
                        adverb_comparative_count+=1
                    elif  (line_under_test[rdlwlil][1])=="RBS":
                        adverb_superlative_list.append(line_under_test[rdlwlil][0])
                        #print("adverb_superlative_list=",adverb_superlative_list)
                        adverb_superlative_count+=1
                    elif  (line_under_test[rdlwlil][1])=="RP":
                        particle_list.append(line_under_test[rdlwlil][0])
                        #print("particle_list=",particle_list)
                        particle_count+=1
                    elif  (line_under_test[rdlwlil][1])=="TO":
                        to_list.append(line_under_test[rdlwlil][0])
                        #print("to_list=",to_list)
                        to_count+=1
                    elif  (line_under_test[rdlwlil][1])=="UH":
                        interjection_list.append(line_under_test[rdlwlil][0])
                        #print("interjection_list=",interjection_list)
                        interjection_count+=1
                    elif  (line_under_test[rdlwlil][1])=="WDT":
                        wh_determiner_list.append(line_under_test[rdlwlil][0])
                        #print("wh_determiner_list=",wh_determiner_list)
                        wh_determiner_count+=1
                    elif  (line_under_test[rdlwlil][1])=="WP":
                        wh_pronoun_list.append(line_under_test[rdlwlil][0])
                        #print("wh_pronoun_list=",wh_pronoun_list)
                        wh_pronoun_count+=1
                    elif  (line_under_test[rdlwlil][1])=="WP$":

possessive_wh_pronoun_list.append(line_under_test[rdlwlil][0])

#print("possessive_wh_pronoun_list=",possessive_wh_pronoun_list)
                        possessive_wh_pronoun_count+=1
                    elif  (line_under_test[rdlwlil][1])=="WRB":
                        wh_adverb_list.append(line_under_test[rdlwlil][0])
                        #print("wh_adverb_list=",wh_adverb_list)
                        wh_adverb_count+=1
                    elif  (line_under_test[rdlwlil][1])==".":
                        punctuation_list.append(line_under_test[rdlwlil][0])
                        #print("wh_adverb_list=",wh_adverb_list)
```

```
                        punctuation_count+=1
                    elif  (line_under_test[rdlwlil][1])==":":
                        parenthesis_list.append(line_under_test[rdlwlil][0])
                        #print("parenthesis_list=",parenthesis_list)
                        parenthesis_count+=1
                    elif  (line_under_test[rdlwlil][1])=="``":

unbalenced_parenthesis_list.append(line_under_test[rdlwlil][0])

#print("unbalenced_parenthesis_list=",unbalenced_parenthesis_list)
                        unbalenced_parenthesis_count+=1

elif(line_under_test[rdlwlil][1])!=":"or"``"or"."or"WRB"or"WP$"or"WP"or"WDT"or"VBZ
"or"VBP"or"VBN"or"VBG"or"VBD"or"VB"or"UH"or"TO"or"RP"or"RBS"or"RBR"or"RB"or"PRP$"o
r"PRP"or"POS"or"PDT"or"NNPS"or"NNP"or"NNS"or"NN"or"MD"or"LS"or"JJS"or"JJR"or"JJ"or
"IN"or"FW"or"EX"or"DT"or"CD"or"CC":
                        not_tagged_word+=1
                        not_tagged_word_list.append(line_under_test[rdlwlil][0])
                        #print('Not interesting in the word in question','\n')


                    rdlwlil+=1

        rdlwl+=1

    #Input the tag name and number of times that tag is in the file into a
dictionary
    tagcount={}

tagcount.update({"noun_common":noun_common_count,"noun_propper":noun_propper_count,
"noun_proper_plural":noun_proper_plural_count,

"verb_base":verb_base_count,"noun_common_plural":noun_common_plural_count,

"verb_past_tense":verb_past_tense_count,"verb_present_participle":verb_present_par
ticiple_count,

"verb_past_participle":verb_past_participle_count,"verb_present_tense_not_3rd_pers
on":verb_present_tense_not_3rd_person_count,

"verb_present_tense_person_singular":verb_present_tense_person_singular_count,"coo
rdinating_conjunction":coordinating_conjunction_count,

"cardinal_number":cardinal_number_count,"determiner":determiner_count,"existential
_there":existential_there_count,

"foreign_word":foreign_word_count,"preposition_subordinating":preposition_subordin
ating_count,"adjective":adjective_count,

"adjective_comparative":adjective_comparative_count,"adjective_superlative":adject
ive_superlative_count,

"list_marker":list_marker_count,"modal":modal_count,"predeterminer":predeterminer_
count,"possessive_ending":possessive_ending_count,

"personal_pronoun":personal_pronoun_count,"possessive_pronoun":possessive_pronoun_
count,"adverb_count":adverb_count,

"adverb_comparative":adverb_comparative_count,"adverb_superlative":adverb_superlat
ive_count,"particle_count":particle_count,
```

```
"to_count":to_count,"interjection":interjection_count,"wh_determiner":wh_determine
r_count,"wh_pronoun":wh_pronoun_count,

"possessive_wh_pronoun":possessive_wh_pronoun_count,"punctuation":punctuation_coun
t,"parenthesis":parenthesis_count,"wh_adverb":wh_adverb_count,

"unbalenced_parenthesis":unbalenced_parenthesis_count,"not_tagged_word":not_tagged
_word})

    #print(tagcount)

    #Input the tag name and the words that have that tag input into a dictonary
    taglists={}

taglists.update({"noun_common_list":noun_common_list,"noun_propper_singular_list":
noun_propper_singular_list,

"noun_propper_list":noun_propper_list,"noun_proper_plural_list":noun_proper_plural
_list,

"noun_common_plural_list":noun_common_plural_list,"verb_base_list":verb_base_list,

"verb_past_tense_list":verb_past_tense_list,"verb_present_participle_list":verb_pr
esent_participle_list,
                    "verb_past_participle_list":verb_past_participle_list,

"verb_present_tense_not_3rd_person_list":verb_present_tense_not_3rd_person_list,

"verb_present_tense_person_singular_list":verb_present_tense_person_singular_list,

"coordinating_conjunction_list":coordinating_conjunction_list,"cardinal_number_lis
t":cardinal_number_list,

"determiner_list":determiner_list,"existential_there_list":existential_there_list,

"foreign_word_list":foreign_word_list,"preposition_subordinating_list":preposition
_subordinating_list,

"adjective_list":adjective_list,"adjective_comparative_list":adjective_comparative
_list,

"adjective_superlative_list":adjective_superlative_list,"list_marker_list":list_ma
rker_list,

"modal_list":modal_list,"predeterminer_list":predeterminer_list,"possessive_ending
_list":possessive_ending_list,

"personal_pronoun_list":personal_pronoun_list,"possessive_pronoun_list":possessive
_pronoun_list,

"adverb_list":adverb_list,"adverb_comparative_list":adverb_comparative_list,

"adverb_superlative_list":adverb_superlative_list,"particle_list":particle_list,"t
o_list":to_list,

"interjection_list":interjection_list,"wh_determiner_list":wh_determiner_list,

"wh_pronoun_list":wh_pronoun_list,"possessive_wh_pronoun_list":possessive_wh_prono
un_list,

"wh_adverb_list":wh_adverb_list,"not_tagged_word_list":not_tagged_word_list,
```

```
"punctuation_list":punctuation_list,"parenthesis_list":parenthesis_list,
                    "unbalenced_parenthesis_list":unbalenced_parenthesis_list})

    #Function returns the two dictionaries
    return (tagcount, taglists)

    #print("taglists",taglists)
```

9.4.12 TAG_PERCENTAGE

```
#Function to calculate the percentage of a tag in a dictionary
#inputs:
#       tagcount = integer number of the number of times a tag is present in a
peace of text
#       NB_tags = number of total tags
#       Key= tag in question
#       NB_words= total number of words in text

def percentage_of_tag(tagcount,NB_tags,NB_words,key):

    nb_key_tag =int(tagcount[key])
    #print("nb_key_tag",nb_key_tag)
    #print("type of nb_key_tag", type(nb_key_tag))
    #print("type of NB_tags", type(NB_tags))
    NB_tags_int=int(NB_tags[0])
    NB_words_int=int(NB_words[0])
    #print("NB_tags_int =",NB_tags_int,"type of NB_tags_int =", type(NB_tags_int))
    percent_key_tag = (nb_key_tag / NB_tags_int)*100
    percent_word_tag = (nb_key_tag / NB_words_int)*100
    #print("percent of key", percent_key_tag)

    return(percent_key_tag,percent_word_tag)
```

9.4.13 TEXT_CHARACTARISTICS

```
# A function that is used to characterise text
#Requires functions:    Tag_count
#                       Read_Textfile_andtag
#                       percentage_of_tag
#Inputs: file path (of the file being analysed)
#Outputs:  token_words_full
#          word_list
#          tagcount
#          taglists
#          number_of_lines
#          NB_tags
#          NB_words
#          percent_of_tag
#          percent_of_words


def Text_charactaristics(file_path):
    #Algorithm that calculates the characteristics of a text file
    #print("Starting program and loading functions")
    #Loading the functions needed for A1 algorithm
    from Tag_count_fn import Tag_count
    from Reading_and_tagging_files import Read_Textfile_andtag
    #Import function to calculate the percentage of a tag or word compared to that
of the whole document
    from tag_percentage_fn import percentage_of_tag
```

```
    #feed the text file into read_textfile_andtag function
    token_words_full, word_list = Read_Textfile_andtag(file_path,'r')
    tagcount, taglists = Tag_count(file_path)

    #print("token words full=",token_words_full,'\n')
    #print("word list=",word_list,'\n')
    #print("tagcount =",tagcount,'\n')
    #print("taglists =",taglists)

    #number of lines in text file under test
    number_of_lines=len(word_list)
    #print("number_of_lines =",number_of_lines)

    #Number on tags counted

NB_tags=[tagcount["noun_common"]+tagcount["noun_propper"]+tagcount["noun_proper_pl
ural"]

+tagcount["noun_common_plural"]+tagcount["verb_base"]+tagcount["verb_past_tense"]
             +tagcount["verb_present_participle"]+tagcount["verb_past_participle"]
             +tagcount["verb_present_tense_not_3rd_person"]+
tagcount["verb_present_tense_person_singular"]

+tagcount["coordinating_conjunction"]+tagcount["cardinal_number"]+tagcount["determ
iner"]

+tagcount["existential_there"]+tagcount["foreign_word"]+tagcount["preposition_subo
rdinating"]

+tagcount["adjective"]+tagcount["adjective_comparative"]+tagcount["adjective_super
lative"]
             +tagcount["list_marker"]+tagcount["modal"]+tagcount["predeterminer"]

+tagcount["possessive_ending"]+tagcount["personal_pronoun"]+tagcount["possessive_p
ronoun"]

+tagcount["adverb_count"]+tagcount["adverb_comparative"]+tagcount["adverb_superlat
ive"]

+tagcount["particle_count"]+tagcount["to_count"]+tagcount["interjection"]

+tagcount["wh_determiner"]+tagcount["wh_pronoun"]+tagcount["possessive_wh_pronoun"]
             +tagcount["wh_adverb"]+tagcount["punctuation"]]

    #Number of tags of just words

NB_words=[tagcount["noun_common"]+tagcount["noun_propper"]+tagcount["noun_proper_p
lural"]

+tagcount["noun_common_plural"]+tagcount["verb_base"]+tagcount["verb_past_tense"]
              +tagcount["verb_present_participle"]+tagcount["verb_past_participle"]
              +tagcount["verb_present_tense_not_3rd_person"]+
tagcount["verb_present_tense_person_singular"]
              +tagcount["coordinating_conjunction"]+tagcount["determiner"]

+tagcount["existential_there"]+tagcount["foreign_word"]+tagcount["preposition_subo
rdinating"]

+tagcount["adjective"]+tagcount["adjective_comparative"]+tagcount["adjective_super
lative"]
              +tagcount["list_marker"]+tagcount["modal"]+tagcount["predeterminer"]
```

```
+tagcount["possessive_ending"]+tagcount["personal_pronoun"]+tagcount["possessive_p
ronoun"]

+tagcount["adverb_count"]+tagcount["adverb_comparative"]+tagcount["adverb_superlat
ive"]

+tagcount["particle_count"]+tagcount["to_count"]+tagcount["interjection"]

+tagcount["wh_determiner"]+tagcount["wh_pronoun"]+tagcount["possessive_wh_pronoun"]
              +tagcount["wh_adverb"]]

    #print("number of tags NB_tags =",NB_tags)
    #print("number of words NB_words =",NB_words)

    # %of each tag as part of the text dictionaries
    percent_of_tag={}
    percent_of_words={}

    #print("testing iterating over a dictionary")
    #A for loop that steps through the dictionary of tags and calculated the
percentages
    for key in tagcount:
        #call function and feed in tag "key"
        function_percent_key_tag,funciton_percent_word_tag
=percentage_of_tag(tagcount,NB_tags,NB_words,key)
        #right calculated percentages to the appropriate dictionaries
        percent_of_tag.update({key:function_percent_key_tag})
        percent_of_words.update({key:funciton_percent_word_tag})

    #print ("percent_of_tag dictionary",percent_of_tag)
    #print ("percent_of_word dictionary",percent_of_words)

    #calculate the average words per line
    average_words_per_line= ((NB_words[0])/number_of_lines)
    #print("Average words per line",average_words_per_line)

    return(token_words_full, word_list, tagcount, taglists, number_of_lines,
NB_tags, NB_words, percent_of_tag, percent_of_words)
```

9.4.14 READ_TEXTFILE_ANDTAG

```
#A function for the reading in of a text file and producing a tagged version of
the file
#inputs (file to be tagged file path and the open type for the file read, wright
or both)
#outputs(the read words and a tagged version of the lines of text)

def Read_Textfile_andtag(file_path, type):
    #print("Importing needed packages")
    import nltk
    from nltk.tokenize import word_tokenize
    from nltk.tag import pos_tag
    #print("Import completed")

    #Read in data from specified file
    #print("Reading in text file")
    with open (file_path,type) as f:
        data = f.readlines()
        #print("data read in from text file")

    #splits the text document into a list of lines
    #Takes lines and places them in a list
        word_list=[]
        for line in data:
            words = line.split()
            word_list.append(words)
         #print(word_list,'\n')
    #print("text file read in and word list created",'\n')
    #defines lenth_word_list as the size of word_list
    lenth_word_list= len(word_list)
    #empty list token_words_full
    token_words_full=[]

    #define loop counter
    ilwl=0
    while ilwl<(lenth_word_list):
        Read_lines_string=' '.join(word_list[ilwl])
        #print(Read_lines_string,'\n')
        #Tokens the string from the list
        tokens=nltk.word_tokenize(Read_lines_string)
        token_words=nltk.pos_tag(tokens)
        token_words_full.append(token_words)
        ilwl+=1

    #print the contents of all of tokenised words list
    #print("The tokenized word list")
    #print(token_words_full,'\n''\n''\n')
    #print("word_list veriable from function =", word_list,'\n')
    #print("token_words_full from function =", token_words_full)
    return (token_words_full, word_list)
```

9.5 TEST FILES FOR PROOF OF CONCEPT VERIFICATION

Test files were generated to use for the verification of the NLP POC they contain words and phrases that are likely to appear in the simulation integration domain.

9.5.1 TEST FILE ONE

The development environments were Matlab and LabVIEW.
FFTS were used as part of the analysis
Time for a spurious word contrafibularities.
Longitudinal model of a vehicle.
S functions were used as part of the integration.
Part of these models were produced by Bosh and DSpace.
Testing the tagging uhhuhhuhh
1) List marker
2) List marker
S Fuction
sFunction
dSPACE
RADAR

fuifius dsuifhdskjfh skdfsduifsd sodufsd skfhsk djfhkdsj sdkfjhsdkjf wetert

how can_WE_confuse the tagger?

What if there were some eroneosley spellt wordss and some jibberish sldkfjd dkfjk dkfjd akdfjd all in one line!

ok so that worked so lets put some others stuff in:
sdlkfj_Adfsdf
this and thatt
some part_hyphonated Wor_ds

body specifications docuemtn world squash Federation (WSF)

labview and matlab were used.

inputs inculde all of the
input one 10 herts floating point
Language C
start date
start
KO

This file holds many words.

The dog jumps over the wall.

9.5.2 TEST FILE TWO

LabVIEW is a modelling language.

FFTS were used.

9.5.3 Test File Three

The development environments is LabVIEW.

FFTS were used as part of the analysis

Time for a spurious word contrafibularities.

Longitudinal model of a vehicle.

S functions were used as part of the integration.

Part of these models were produced by Bosh and DSpace.

Testing the tagging uhhuhhuhh

1) List marker

2) List marker

S Fuction

sFunction

dSPACE

RADAR

9.5.4 Result of the Proof of Concept Test File

The NLP test files were used throughout the development of the NLP POC to verify that the outputs are correct.

File inputs:

A. NLP_Test_File.txt

B. NLP_Test_File.txt

Outputs from the NLP application:

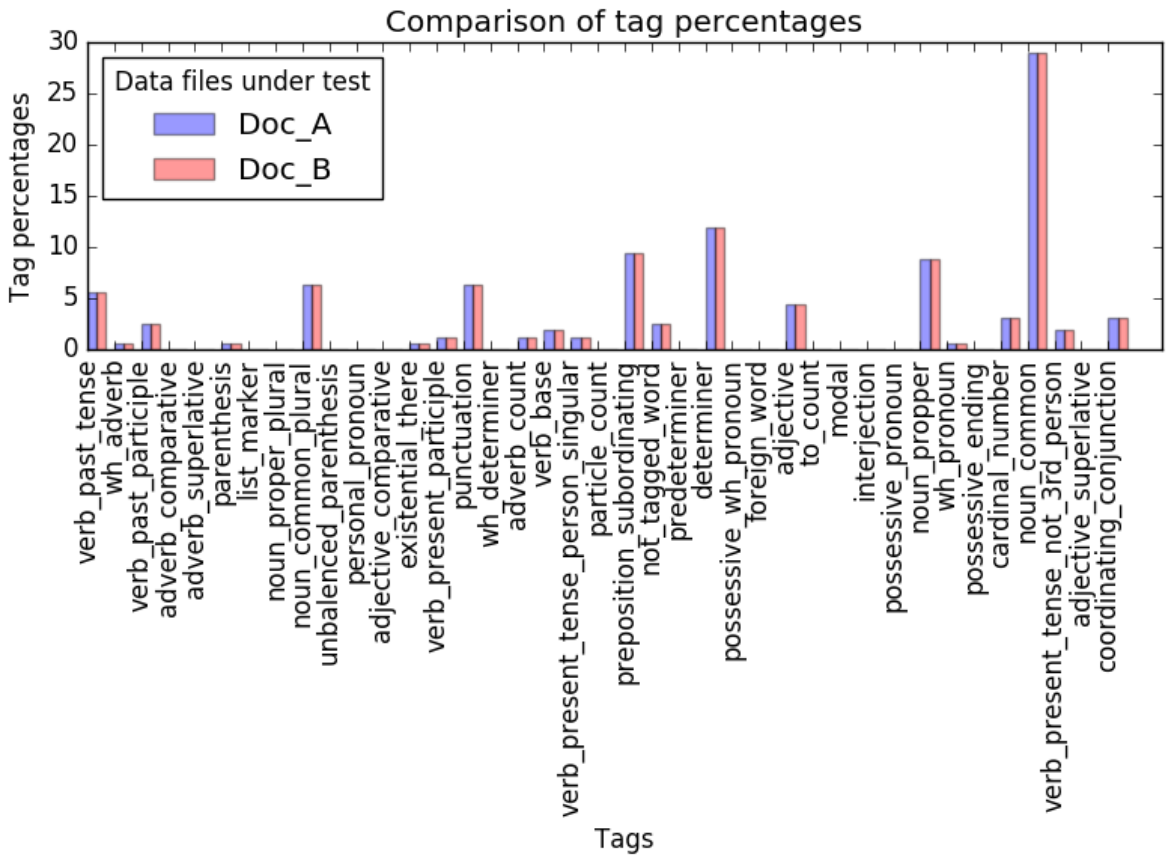

Figure 9.17: Output from the NLP POC when the test file is both document A and B.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 15
common_dev_enviroments = ['Matlab', 'labview', 'matlab']
number of words = 3
common_identified_modeling_tuerms = ['inputs', 'input']
number of common_identified_modeling_tuerms = 2
common_identified_project_tuerms = ['KO']
number of common_identified_project_tuerms = 2
common_identified_prog_languages = ['C']
```

number of common_identified_prog_languages = 1
common_identified_file_types = ['C']
number of common_identified_file_types = 1
Identified_common_company_words =
'number_of_common_identified_modeling_tuerms': 2,
'number_of_common_identified_project_tuerms': 1,
'common_identified_project_tuerms': ['KO'],
'number_of_common_dev_enviroments': 3,
'number_of_identified_prog_languages': 1,
'number_of_identified_file_types': 1,
'common_identified_prog_languages': ['C'],
'common_dev_enviroments': ['Matlab', 'labview', 'matlab'],
'common_identified_modeling_tuerms': ['inputs', 'input'],
'common_identified_file_types': ['C']}

dict_of_words_in_both_docs = {'wh_pronoun_list': ['What'], 'to_list': [], 'adjec
tive_list': ['Longitudinal', 'many', 'eroneosley', 'spurious', 'lets', 'S', 'jib
berish'], 'coordinating_conjunction_list': ['and'], 'existential_there_list': ['
there'], 'verb_past_tense_list': ['put', 'worked', 'part_hyphonated', 'were'], '
adverb_superlative_list': [], 'verb_present_tense_person_singular_list': ['holds
', 'jumps'], 'cardinal_number_list': ['one', '10', '1', '2'], 'preposition_subor
dinating_list': ['that', 'in', 'of', 'akdfjd', 'if', 'as', 'for', 'by', 'over'],
'list_marker_list': [], 'adverb_comparative_list': [], 'interjection_list': [],
'noun_proper_plural_list': [], 'parenthesis_list': [':'], 'noun_common_plural_l
ist': ['words', 'functions', 'contrafibularities', 'models', 'others', 'specific
ations', 'inputs', 'environments', 'herts', 'ok'], 'foreign_word_list': [], 'per
sonal_pronoun_list': [], 'noun_propper_list': [], 'noun_common_list': ['developm
ent', 'dkfjk', 'labview', 'sFunction', 'djfhkdsj', 'line', 'skdfsduifsd', 'start
', 'matlab', 'part', 'wordss', 'dSPACE', 'world', 'wetert', 'fuifius', 'tagging'
, 'sodufsd', 'tagger', 'List', 'integration', 'squash', 'sdlkfj_Adfsdf', 'dsuifh
dskjfh', 'analysis', 'marker', 'dog', 'sdkfjhsdkjf', 'body', 'wall', 'date', 've
hicle', 'RADAR', 'Part', 'file', 'sldkfjd', 'uhhuhhuhh', 'spellt', 'point', 'dkf
jd', 'word', 'model', 'skfhsk'], 'verb_past_participle_list': ['used', 'produced
'], 'punctuation_list': ['.', '!', '?'], 'verb_present_tense_not_3rd_person_list
': ['docuemtn', 'inculde', 'stuff'], 'verb_present_participle_list': ['Testing',
'floating'], 'particle_list': [], 'not_tagged_word_list': [')', '('], 'noun_pro
pper_singular_list': ['DSpace', 'C', 'KO', 'Time', 'Matlab', 'Fuction', 'Languag
e', 'FFTS', 'Bosh', 'Federation', 'S', 'LabVIEW', 'WSF', 'Wor_ds'], 'adjective_c
omparative_list': [], 'predeterminer_list': [], 'adjective_superlative_list': []
, 'verb_base_list': ['input', 'can_WE_confuse', 'thatt'], 'possessive_pronoun_li
st': [], 'possessive_ending_list': [], 'wh_determiner_list': [], 'unbalenced_par
enthesis_list': [], 'adverb_list': ['so'], 'possessive_wh_pronoun_list': [], 'wh
_adverb_list': ['how'], 'modal_list': [], 'determiner_list': ['a', 'the', 'The',
'this', 'This', 'these', 'some', 'all']}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('The', 'DT'), ('development', 'NN'), ('environments', 'NNS'), ('were', 'VBD'), ('Matlab', 'NNP'), ('and', 'CC'), ('LabVIEW', 'NNP'), ('.', '.')], [('The', 'DT'), ('development', 'NN'), ('environments', 'NNS'), ('were', 'VBD'), ('Matlab', 'NNP'), ('and', 'CC'), ('LabVIEW', 'NNP'), ('.', '.')]], [[('FFTS', 'NNP'), ('were', 'VBD'), ('used', 'VBN'), ('as', 'IN'), ('part', 'NN'), ('of', 'IN'), ('the', 'DT'), ('analysis', 'NN')], [('FFTS', 'NNP'), ('were', 'VBD'), ('used', 'VBN'), ('as', 'IN'), ('part', 'NN'), ('of', 'IN'), ('the', 'DT'), ('analysis', 'NN')]], [[('S', 'JJ'), ('functions', 'NNS'), ('were', 'VBD'), ('used', 'VBN'), ('as', 'IN'), ('part', 'NN'), ('of', 'IN'), ('the', 'DT'), ('integration', 'NN'), ('.', '.')], [('S', 'JJ'), ('functions', 'NNS'), ('were', 'VBD'), ('used', 'VBN'), ('as', 'IN'), ('part', 'NN'), ('of', 'IN'), ('the', 'DT'), ('integration', 'NN'), ('.', '.')]], [[('Part', 'NN'), ('of', 'IN'), ('these', 'DT'), ('models', 'NNS'), ('were', 'VBD'), ('produced', 'VBN'), ('by', 'IN'), ('Bosh', 'NNP'), ('and', 'CC'), ('DSpace', 'NNP'), ('.', '.')], [('Part', 'NN'), ('of', 'IN'), ('these', 'DT'), ('models', 'NNS'), ('were', 'VBD'), ('produced', 'VBN'), ('by', 'IN'), ('Bosh', 'NNP'), ('and', 'CC'), ('DSpace', 'NNP'), ('.', '.')]], [[('Testing', 'VBG'), ('the', 'DT'), ('tagging', 'NN'), ('uhhuhhuhh', 'NN')], [('Testing', 'VBG'), ('the', 'DT'), ('tagging', 'NN'), ('uhhuhhuhh', 'NN')]], [[('how', 'WRB'), ('can_WE_confuse', 'VB'), ('the', 'DT'), ('tagger', 'NN'), ('?', '.')], [('how', 'WRB'), ('can_WE_confuse', 'VB'), ('the', 'DT'), ('tagger', 'NN'), ('?', '.')]], [[('What', 'WP'), ('if', 'IN'), ('there', 'EX'), ('were', 'VBD'), ('some', 'DT'), ('eroneosley', 'JJ'), ('spellt', 'NN'), ('wordss', 'NN'), ('and', 'CC'), ('some', 'DT'), ('jibberish', 'JJ'), ('sldkfjd', 'NN'), ('dkfjk', 'NN'), ('dkfjd', 'NN'), ('akdfjd', 'IN'), ('all', 'DT'), ('in', 'IN'), ('one', 'CD'), ('line', 'NN'), ('!', '.')], [('What', 'WP'), ('if', 'IN'), ('there', 'EX'), ('were', 'VBD'), ('some', 'DT'), ('eroneosley', 'JJ'), ('spellt', 'NN'), ('wordss', 'NN'), ('and', 'CC'), ('some', 'DT'), ('jibberish', 'JJ'), ('sldkfjd', 'NN'), ('dkfjk', 'NN'), ('dkfjd', 'NN'), ('akdfjd', 'IN'), ('all', 'DT'), ('in', 'IN'), ('one', 'CD'), ('line', 'NN'), ('!', '.')]], [[('ok', 'NNS'), ('so', 'RB'), ('that', 'IN'), ('worked', 'VBD'), ('so', 'RB'), ('lets', 'JJ'), ('put', 'VBD'), ('some', 'DT'), ('others', 'NNS'), ('stuff', 'VBP'), ('in', 'IN'), (':', ':')], [('ok', 'NNS'), ('so', 'RB'), ('that', 'IN'), ('worked', 'VBD'), ('so', 'RB'), ('lets', 'JJ'), ('put', 'VBD'), ('some', 'DT'), ('others', 'NNS'), ('stuff', 'VBP'), ('in', 'IN'), (':', ':')]], [[('some', 'DT'), ('part_hyphonated', 'VBD'), ('Wor_ds', 'NNP')], [('some', 'DT'), ('part_hyphonated', 'VBD'), ('Wor_ds', 'NNP')]], [[('body', 'NN'), ('specifications', 'NNS'), ('docuemtn', 'VBP'), ('world', 'NN'), ('squash', 'NN'), ('Federation', 'NNP'), ('(', '('), ('WSF', 'NNP'), (')', ')')], [('body', 'NN'), ('specifications', 'NNS'), ('docuemtn', 'VBP'), ('world', 'NN'), ('squash', 'NN'), ('Federation', 'NNP'), ('(', '('), ('WSF', 'NNP'), (')', ')')]], [[('labview', 'NN'), ('and', 'CC'), ('matlab', 'NN'), ('were', 'VBD'), ('used', 'VBN'), ('.', '.')], [('labview', 'NN'), ('and', 'CC'), ('matlab', 'NN'), ('were', 'VBD'), ('used', 'VBN'), ('.', '.')]], [[('inputs', 'NNS'), ('inculde', 'VBP'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT')], [('inputs', 'NNS'), ('inculde', 'VBP'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT')]], [[('input', 'VB'), ('one', 'CD'), ('10', 'CD'), ('herts', 'NNS'), ('floating', 'VBG'), ('point', 'NN')], [('input', 'VB'), ('one', 'CD'), ('10', 'CD'), ('herts', 'NNS'), ('floating', 'VBG'), ('point', 'NN')]], [[('This', 'DT'), ('file', 'NN'), ('holds', 'VBZ'), ('many', 'JJ'), ('words', 'NNS'), ('.', '.')], [('This', 'DT'), ('file', 'NN'), ('holds', 'VBZ'), ('many', 'JJ'), ('words',

'NNS'), ('.', '.')]], [[('The', 'DT'), ('dog', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('wall', 'NN'), ('.', '.')], [('The', 'DT'), ('dog', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('wall', 'NN'), ('.', '.')]]]

9.5.5 VERIFICATION ANALYSIS TWO

The NLP test files were used throughout the development of the NLP POC to verify that the outputs are correct.

File inputs:

NLP_Test_File.txt

NLP_Test_File_1.txt

Outputs from the NLP application:

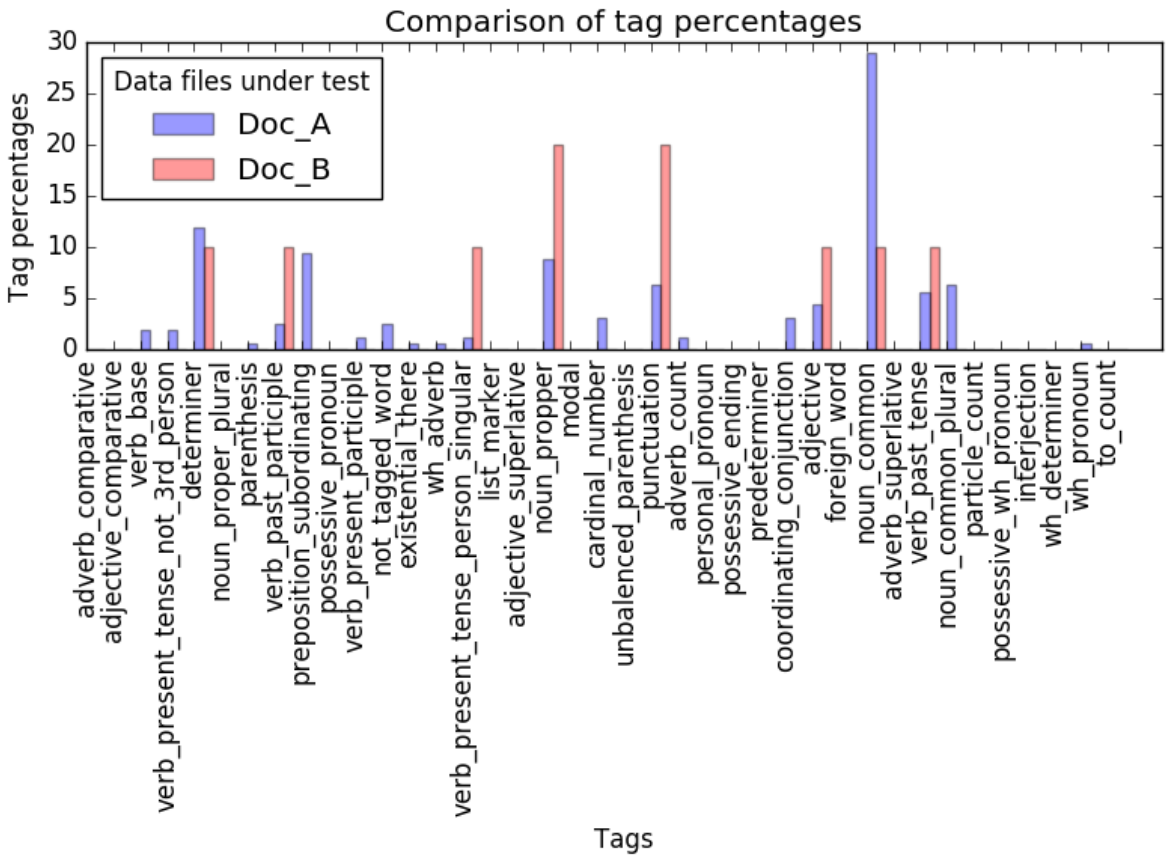

Figure 9.18 : Output from the NLP POC when the NLP_Test_File is both document A and B.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 2
common_dev_enviroments = []
 number of words = 0
common_identified_modeling_tuerms = []
 number of common_identified_modeling_tuerms = 0
common_identified_project_tuerms = []
 number of common_identified_project_tuerms = 0
common_identified_prog_languages = []
```

number of common_identified_prog_languages = 0
common_identified_file_types = []
number of common_identified_file_types = 0
Identified_common_company_words = {
'common_dev_enviroments': [],
'number_of_identified_file_types': 0,
'number_of_common_dev_enviroments': 0,
'common_identified_modeling_tuerms': [],
'common_identified_project_tuerms': [],
'common_identified_file_types': [],
'common_identified_prog_languages': [],
'number_of_identified_prog_languages': 0,
'number_of_common_identified_modeling_tuerms': 0,
'number_of_common_identified_project_tuerms': 0}

dict_of_words_in_both_docs = {
'adverb_comparative_list': [],
'particle_list': [],
'punctuation_list': ['.'],
'modal_list': [],
'noun_propper_singular_list': ['LabVIEW', 'FFTS'],
'interjection_list': [],
'predeterminer_list': [],
'wh_determiner_list': [],
'adjective_comparative_list': [],
'coordinating_conjunction_list': [],
'verb_present_tense_person_singular_list': [],
'wh_pronoun_list': [],
'verb_past_tense_list': ['were'],
'existential_there_list': [],
'foreign_word_list': [],
'determiner_list': ['a'],
'unbalenced_parenthesis_list': [],
'noun_proper_plural_list': [],
'adverb_superlative_list': [],
'to_list': [],
'verb_past_participle_list': ['used'],
'adjective_superlative_list': [],
'possessive_wh_pronoun_list': [],
'noun_common_plural_list': [],
'possessive_ending_list': [],
'parenthesis_list': [],
'wh_adverb_list': [],
'personal_pronoun_list': [],
'noun_propper_list': [],

'verb_base_list': [],
'verb_present_tense_not_3rd_person_list': [],
'list_marker_list': [],
'preposition_subordinating_list': [],
'adjective_list': [],
'not_tagged_word_list': [],
'verb_present_participle_list': [],
'possessive_pronoun_list': [],
'adverb_list': [],
'noun_common_list': [],
'cardinal_number_list': []}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('The', 'DT'), ('development', 'NN'), ('environments', 'NNS'), ('were', 'VBD'), ('Matlab', 'NNP'), ('and', 'CC'), ('LabVIEW', 'NNP'), ('.', '.')], [('LabVIEW', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('modelling', 'JJ'), ('language', 'NN'), ('.', '.')]], [[('FFTS', 'NNP'), ('were', 'VBD'), ('used', 'VBN'), ('as', 'IN'), ('part', 'NN'), ('of', 'IN'), ('the', 'DT'), ('analysis', 'NN')], [('FFTS', 'NNP'), ('were', 'VBD'), ('used', 'VBN'), ('.', '.')]]]

9.5.6 VERIFICATION ANALYSIS THREE

The NLP test files were used throughout the development of the NLP POC to verify that the outputs are correct.

File inputs

NLPTestFile

NLPTestFile_2

Outputs from the NLP application:

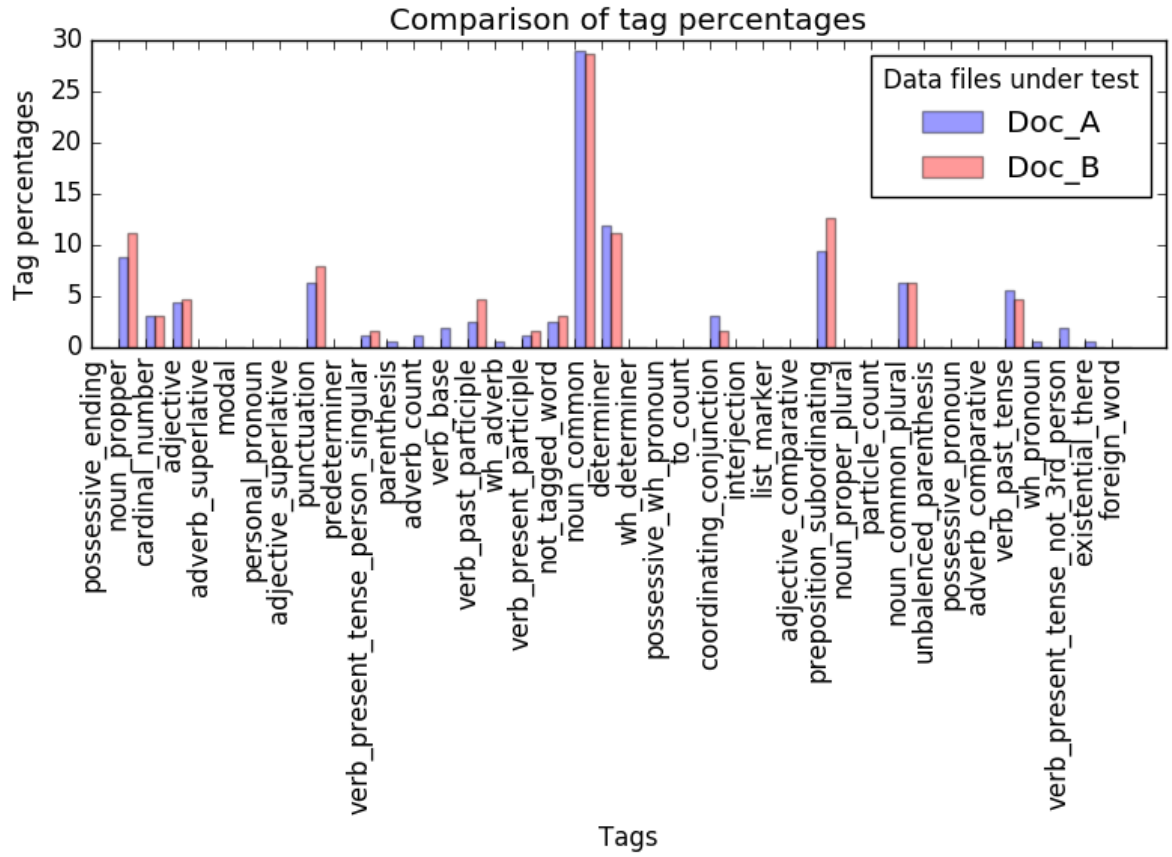

Figure 9.19 : Output from the NLP POC when document A is NLPTestFile and NLPTestFile_2 is input B.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 5
common_dev_enviroments = []
number of words = 0
common_identified_modeling_tuerms = []
number of common_identified_modeling_tuerms = 0
```

common_identified_project_tuerms = []
number of common_identified_project_tuerms = 0
common_identified_prog_languages = []
number of common_identified_prog_languages = 0
common_identified_file_types = []
number of common_identified_file_types = 0
Identified_common_company_words = {
'common_identified_prog_languages': [],
'common_identified_project_tuerms': [],
'number_of_common_identified_project_tuerms': 0,
'number_of_identified_file_types': 0,
'number_of_identified_prog_languages': 0,
'common_identified_file_types': [],
'common_identified_modeling_tuerms': [],
'common_dev_enviroments': [],
'number_of_common_identified_modeling_tuerms': 0,
'number_of_common_dev_enviroments': 0}

dict_of_words_in_both_docs = {
'noun_common_plural_list': ['environments', 'functions', 'models',
'contrafibularities'],
unbalenced_parenthesis_list': [],
'adverb_list': [],
'punctuation_list': ['.'],
'possessive_ending_list': [],
'adjective_list': ['spurious', 'S', 'Longitudinal'],
'modal_list': [],
'verb_present_tense_person_singular_list': [],
'possessive_wh_pronoun_list': [],
'list_marker_list': [],
'adverb_superlative_list': [],
'verb_present_tense_not_3rd_person_list':[],
'verb_past_participle_list': ['used', 'produced'],
'coordinating_conjunction_list': ['and'],
'preposition_subordinating_list': ['as', 'by', 'for', 'of'],
'wh_pronoun_list': [],
'adjective_comparative_list': [],
'to_list': [],
'determiner_list': ['The', 'these', 'the', 'a'],
'adjective_superlative_list': [],
'noun_proper_plural_list': [],
'adverb_comparative_list': [],
'interjection_list': [],
'noun_propper_list': [],
'cardinal_number_list': ['1', '2'],

'existential_there_list': [],
'parenthesis_list': [],
'wh_determiner_list': [],
'personal_pronoun_list': [],
'wh_adverb_list': [],
'predeterminer_list': [],
'noun_common_list': ['part', 'sFunction', 'development', 'word', 'tagging',
'vehicle', 'uhhuhhuhh', 'analysis', 'RADAR', 'integration', 'List', 'dSPACE', 'model',
'Part', 'marker'],
'not_tagged_word_list': [')'],
'verb_base_list': [],
'particle_list': [],
'verb_past_tense_list': ['were'],
'noun_propper_singular_list': ['Bosh', 'Fuction', 'FFTS', 'LabVIEW', 'S', 'Time',
'DSpace'],
'verb_present_participle_list': ['Testing'],
'foreign_word_list': [],
'possessive_pronoun_list': []}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('The', 'DT'), ('development', 'NN'), ('environments', 'NNS'), ('were', 'VBD'),
('Matlab', 'NNP'), ('and', 'CC'), ('LabVIEW', 'NNP'), ('.', '.')], [('The', 'DT'),
('development', 'NN'), ('environments', 'NNS'), ('is', 'VBZ'), ('LabVIEW', 'NNP'), ('.',
'.')]], [[('FFTS', 'NNP'), ('were', 'VBD'), ('used', 'VBN'), ('as', 'IN'), ('part', 'NN'), ('of',
'IN'), ('the', 'DT'), ('analysis', 'NN')], [('FFTS', 'NNP'), ('were', 'VBD'), ('used', 'VBN'),
('as', 'IN'), ('part', 'NN'), ('of', 'IN'), ('the', 'DT'), ('analysis', 'NN')]], [[('S', 'JJ'),
('functions', 'NNS'), ('were', 'VBD'), ('used', 'VBN'), ('as', 'IN'), ('part', 'NN'), ('of', 'IN'),
('the', 'DT'), ('integration', 'NN'), ('.', '.')], [('S', 'JJ'), ('functions', 'NNS'), ('were',
'VBD'), ('used', 'VBN'), ('as', 'IN'), ('part', 'NN'), ('of', 'IN'), ('the', 'DT'), ('integration',
'NN'), ('.', '.')]], [[('Part', 'NN'), ('of', 'IN'), ('these', 'DT'), ('models', 'NNS'), ('were',
'VBD'), ('produced', 'VBN'), ('by', 'IN'), ('Bosh', 'NNP'), ('and', 'CC'), ('DSpace',
'NNP'), ('.', '.')], [('Part', 'NN'), ('of', 'IN'), ('these', 'DT'), ('models', 'NNS'), ('were',
'VBD'), ('produced', 'VBN'), ('by', 'IN'), ('Bosh', 'NNP'), ('and', 'CC'), ('DSpace',
'NNP'), ('.', '.')]], [[('Testing', 'VBG'), ('the', 'DT'), ('tagging', 'NN'), ('uhhuhhuhh',
'NN')], [('Testing', 'VBG'), ('the', 'DT'), ('tagging', 'NN'), ('uhhuhhuhh', 'NN')]]]

9.6 Outputs From The Proof of Concept Application

The NLP application reads in two text files and outputs information regarding the text files and the similarities between them. In the tests that are conducted the file inputs A and B relate to the two file inputs to the application. The output of the application include a bar chart of the percentages of the identified word types, common identified words, common words which exist in both documents, and a text file of phrases that contain the same noun and verb phrases that exist in both documents.

Both of the case studies that were conducted to test the proposed methods have been used to test these NLP methods as well.

9.6.1 Case study One Squash Court

The first analysis is the comparison between the model documentation and the simulation requirements. There are three potential simulation components therefore there are three sets of data. The subsequent three tasks were comparing each potential component with every other component to ascertain if there is the potential for meaningful integration.

9.6.2 Analysis one

File inputs:

A. Requirements_For_Case_Study_One_Squash_Ball_Moving_Around_A_Court.txt

B. Documentation_of_a_Particle_Moving_in_Free_Space.txt

Outputs From NLP application:

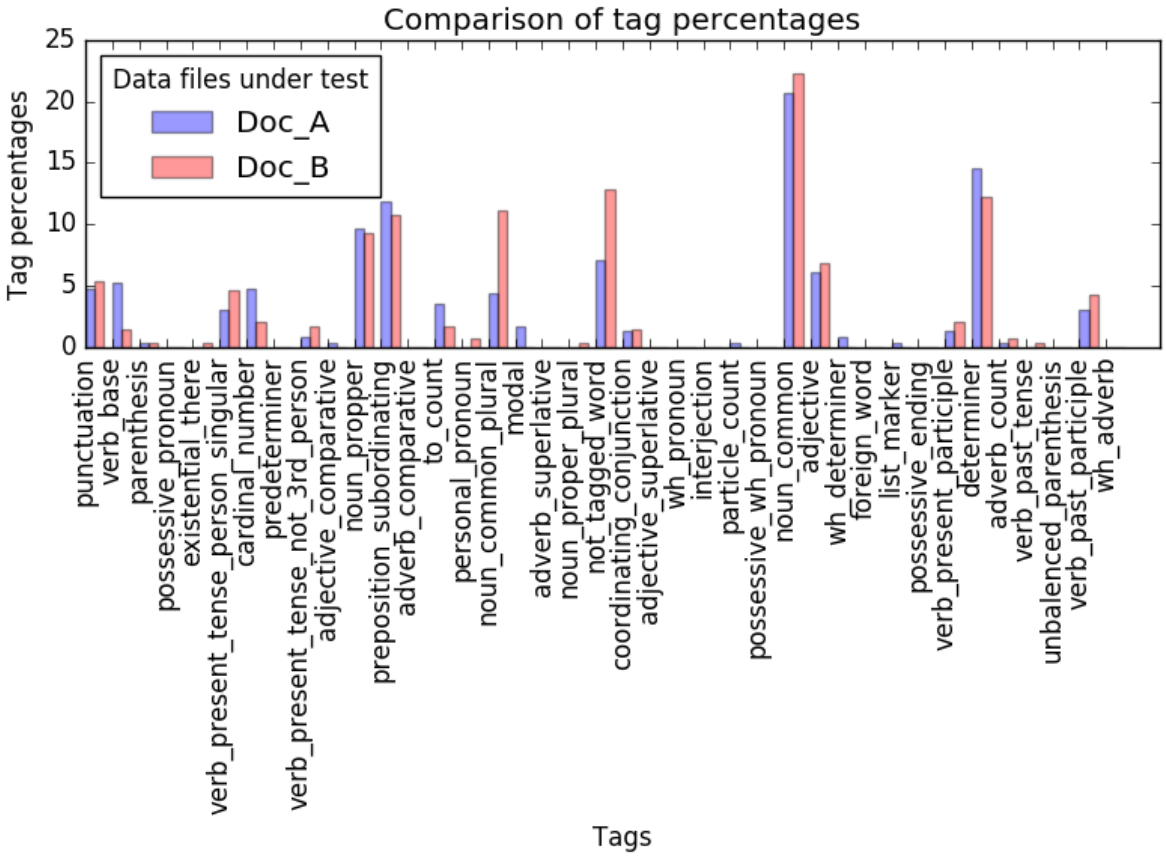

Figure 9.20: The percentage distribution of tags within the two compared documents from case study one analysis one.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 10

common_dev_enviroments = ['LabVIEW']
number of words = 1
common_identified_modeling_tuerms = []
number of common_identified_modeling_tuerms = 0
common_identified_project_tuerms = []
number of common_identified_project_tuerms = 0
common_identified_prog_languages = []
number of common_identified_prog_languages = 0
common_identified_file_types = []
number of common_identified_file_types = 0

Identified_common_company_words ={
number_of_common_identified_modeling_tuerms': 0,
number_of_identified_file_types': 0,
'common_identified_project_tuerms': [],
'common_identified_prog_languages': [],
'common_dev_enviroments': ['LabVIEW'],
'common_identified_modeling_tuerms': [],
'number_of_identified_prog_languages': 0,
'number_of_common_dev_enviroments': 1,
'number_of_common_identified_project_tuerms': 0,
'common_identified_file_types': []
}

dict_of_words_in_both_docs = {
'adverb_superlative_list': [],
'possessive_ending_list': [],
'adverb_list': [],
'wh_pronoun_list': [],
'verb_base_list': ['be'],
 'cardinal_number_list': ['7'],
 'possessive_pronoun_list': [],
'verb_past_tense_list': [],
'wh_adverb_list': [],
'foreign_word_list': [],
'noun_common_list': ['space', 'date', 'time', 'file', 'ball'],
'noun_propper_singular_list': ['LabVIEW', 'A'],
'interjection_list': [],
'noun_proper_plural_list': [],
'noun_propper_list': [],
```

'personal_pronoun_list': [],
'adjective_superlative_list': [],
'adverb_comparative_list': [],
'modal_list': [],
'not_tagged_word_list': [')', ','],
'verb_present_participle_list': [],
'verb_past_participle_list': ['used'],
'unbalenced_parenthesis_list': [],
'existential_there_list': [],
'preposition_subordinating_list': ['in', 'of'],
'particle_list': [],
'adjective_list': [],
'list_marker_list': [],
'noun_common_plural_list': ['steps', 'seconds'],
'to_list': ['to'], '
coordinating_conjunction_list': ['and'],
'adjective_comparative_list': [],
'possessive_wh_pronoun_list': [],
'verb_present_tense_person_singular_list': ['is'],
'punctuation_list': ['.'],
'predeterminer_list': [],
'verb_present_tense_not_3rd_person_list': ['are'],
'parenthesis_list': [],
'wh_determiner_list': [],
'determiner_list': ['the', 'a', 'The']}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('1', 'CD'), ('(', '('), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('squash', 'NN'), ('ball', 'NN'), ('in', 'IN'), ('flight', 'NN'), ('.', '.')], [('Model', 'NNP'), ('requires', 'VBZ'), ('final', 'JJ'), ('velocity', 'NN'), (',', ','), ('starting', 'VBG'), ('velocity', 'NN'), (',', ','), ('distance', 'NN'), ('and', 'CC'), ('sample', 'NN'), ('rate', 'NN'), ('to', 'TO'), ('be', 'VB'), ('known', 'VBN'), ('and', 'CC'), ('it', 'PRP'), ('calculates', 'VBZ'), ('acceleration', 'NN'), (',', ','), ('time', 'NN'), ('that', 'IN'), ('it', 'PRP'), ('takes', 'VBZ'), ('for', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('to', 'TO'), ('get', 'VB'), ('from', 'IN'), ('one', 'CD'), ('end', 'NN'), ('to', 'TO'), ('the', 'DT'), ('other', 'JJ'), (',', ','), ('time', 'NN'), ('sample', 'NN'), (',', ','), ('displacement', 'JJ'), ('steps', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('time', 'NN'), ('sample', 'NN'), ('and', 'CC'), ('the', 'DT'), ('distance', 'NN'), ('that', 'IN'), ('ball', 'NN'), ('travels', 'NNS'), ('in', 'IN'), ('a', 'DT'), ('single', 'JJ'), ('time', 'NN'), ('sample', 'NN'), ('.', '.')]], [[('2', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is',

'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('squash', 'NN'), ('ball', 'NN'), ('bouncing', 'VBG'), ('off', 'RP'), ('the', 'DT'), ('surface', 'NN'), ('of', 'IN'), ('a', 'DT'), ('wall', 'NN'), ('.', '.')], [('Model', 'NNP'), ('requires', 'VBZ'), ('final', 'JJ'), ('velocity', 'NN'), (',', ','), ('starting', 'VBG'), ('velocity', 'NN'), (',', ','), ('distance', 'NN'), ('and', 'CC'), ('sample', 'NN'), ('rate', 'NN'), ('to', 'TO'), ('be', 'VB'), ('known', 'VBN'), ('and', 'CC'), ('it', 'PRP'), ('calculates', 'VBZ'), ('acceleration', 'NN'), (',', ','), ('time', 'NN'), ('that', 'IN'), ('it', 'PRP'), ('takes', 'VBZ'), ('for', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('to', 'TO'), ('get', 'VB'), ('from', 'IN'), ('one', 'CD'), ('end', 'NN'), ('to', 'TO'), ('the', 'DT'), ('other', 'JJ'), (',', ','), ('time', 'NN'), ('sample', 'NN'), (',', ','), ('displacement', 'JJ'), ('steps', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('time', 'NN'), ('sample', 'NN'), ('and', 'CC'), ('the', 'DT'), ('distance', 'NN'), ('that', 'IN'), ('ball', 'NN'), ('travels', 'NNS'), ('in', 'IN'), ('a', 'DT'), ('single', 'JJ'), ('time', 'NN'), ('sample', 'NN'), ('.', '.')]], [[('3', 'LS'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('ball', 'NN'), ('interacting', 'VBG'), ('with', 'IN'), ('a', 'DT'), ('player's', 'NN'), ('racket', 'NN'), ('.', '.')], [('Model', 'NNP'), ('requires', 'VBZ'), ('final', 'JJ'), ('velocity', 'NN'), (',', ','), ('starting', 'VBG'), ('velocity', 'NN'), (',', ','), ('distance', 'NN'), ('and', 'CC'), ('sample', 'NN'), ('rate', 'NN'), ('to', 'TO'), ('be', 'VB'), ('known', 'VBN'), ('and', 'CC'), ('it', 'PRP'), ('calculates', 'VBZ'), ('acceleration', 'NN'), (',', ','), ('time', 'NN'), ('that', 'IN'), ('it', 'PRP'), ('takes', 'VBZ'), ('for', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('to', 'TO'), ('get', 'VB'), ('from', 'IN'), ('one', 'CD'), ('end', 'NN'), ('to', 'TO'), ('the', 'DT'), ('other', 'JJ'), (',', ','), ('time', 'NN'), ('sample', 'NN'), (',', ','), ('displacement', 'JJ'), ('steps', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('time', 'NN'), ('sample', 'NN'), ('and', 'CC'), ('the', 'DT'), ('distance', 'NN'), ('that', 'IN'), ('ball', 'NN'), ('travels', 'NNS'), ('in', 'IN'), ('a', 'DT'), ('single', 'JJ'), ('time', 'NN'), ('sample', 'NN'), ('.', '.')]], [[('4', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('needs', 'VBZ'), ('to', 'TO'), ('be', 'VB'), ('run', 'VBN'), ('multiple', 'JJ'), ('times', 'NNS'), ('with', 'IN'), ('the', 'DT'), ('only', 'JJ'), ('change', 'NN'), ('being', 'VBG'), ('the', 'DT'), ('compound', 'NN'), ('of', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('.', '.')], [('Model', 'NNP'), ('requires', 'VBZ'), ('final', 'JJ'), ('velocity', 'NN'), (',', ','), ('starting', 'VBG'), ('velocity', 'NN'), (',', ','), ('distance', 'NN'), ('and', 'CC'), ('sample', 'NN'), ('rate', 'NN'), ('to', 'TO'), ('be', 'VB'), ('known', 'VBN'), ('and', 'CC'), ('it', 'PRP'), ('calculates', 'VBZ'), ('acceleration', 'NN'), (',', ','), ('time', 'NN'), ('that', 'IN'), ('it', 'PRP'), ('takes', 'VBZ'), ('for', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('to', 'TO'), ('get', 'VB'), ('from', 'IN'), ('one', 'CD'), ('end', 'NN'), ('to', 'TO'), ('the', 'DT'), ('other', 'JJ'), (',', ','), ('time', 'NN'), ('sample', 'NN'), (',', ','), ('displacement', 'JJ'), ('steps', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('time', 'NN'), ('sample', 'NN'), ('and', 'CC'), ('the', 'DT'), ('distance', 'NN'), ('that', 'IN'), ('ball', 'NN'), ('travels', 'NNS'), ('in', 'IN'), ('a', 'DT'), ('single', 'JJ'), ('time', 'NN'), ('sample', 'NN'), ('.', '.')]], [[('5', 'CD'), (')', ')'), ('The', 'DT'), ('flight', 'NN'), ('of', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('constrained', 'VBN'), ('by', 'IN'), ('a', 'DT'), ('regulation', 'NN'), ('sized', 'VBN'), ('squash', 'JJ'), ('court', 'NN'), ('.', '.')], [('Model', 'NNP'), ('requires', 'VBZ'), ('final', 'JJ'), ('velocity', 'NN'), (',', ','), ('starting', 'VBG'), ('velocity', 'NN'), (',', ','), ('distance', 'NN'), ('and', 'CC'), ('sample', 'NN'), ('rate', 'NN'), ('to', 'TO'), ('be', 'VB'), ('known', 'VBN'), ('and', 'CC'), ('it', 'PRP'), ('calculates', 'VBZ'), ('acceleration', 'NN'), (',', ','), ('time', 'NN'), ('that', 'IN'), ('it', 'PRP'), ('takes', 'VBZ'), ('for', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('to', 'TO'), ('get',

'VB'), ('from', 'IN'), ('one', 'CD'), ('end', 'NN'), ('to', 'TO'), ('the', 'DT'), ('other', 'JJ'), (',', ','), ('time', 'NN'), ('sample', 'NN'), (',', ','), ('displacement', 'JJ'), ('steps', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('time', 'NN'), ('sample', 'NN'), ('and', 'CC'), ('the', 'DT'), ('distance', 'NN'), ('that', 'IN'), ('ball', 'NN'), ('travels', 'NNS'), ('in', 'IN'), ('a', 'DT'), ('single', 'JJ'), ('time', 'NN'), ('sample', 'NN'), ('.', '.')]], [[('A', 'DT'), (')', ')'), ('The', 'DT'), ('total', 'JJ'), ('run', 'NN'), ('time', 'NN'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('should', 'MD'), ('take', 'VB'), ('less', 'JJR'), ('than', 'IN'), ('five', 'CD'), ('minutes', 'NNS'), ('to', 'TO'), ('fully', 'RB'), ('execute', 'VB'), ('.', '.')], [('Model', 'NNP'), ('requires', 'VBZ'), ('final', 'JJ'), ('velocity', 'NN'), (',', ','), ('starting', 'VBG'), ('velocity', 'NN'), (',', ','), ('distance', 'NN'), ('and', 'CC'), ('sample', 'NN'), ('rate', 'NN'), ('to', 'TO'), ('be', 'VB'), ('known', 'VBN'), ('and', 'CC'), ('it', 'PRP'), ('calculates', 'VBZ'), ('acceleration', 'NN'), (',', ','), ('time', 'NN'), ('that', 'IN'), ('it', 'PRP'), ('takes', 'VBZ'), ('for', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('to', 'TO'), ('get', 'VB'), ('from', 'IN'), ('one', 'CD'), ('end', 'NN'), ('to', 'TO'), ('the', 'DT'), ('other', 'JJ'), (',', ','), ('time', 'NN'), ('sample', 'NN'), (',', ','), ('displacement', 'JJ'), ('steps', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('time', 'NN'), ('sample', 'NN'), ('and', 'CC'), ('the', 'DT'), ('distance', 'NN'), ('that', 'IN'), ('ball', 'NN'), ('travels', 'NNS'), ('in', 'IN'), ('a', 'DT'), ('single', 'JJ'), ('time', 'NN'), ('sample', 'NN'), ('.', '.')]], [[('B', 'NNP'), (')', ')'), ('The', 'DT'), ('overall', 'JJ'), ('simulation', 'NN'), ('and', 'CC'), ('analysis', 'NN'), ('should', 'MD'), ('be', 'VB'), ('possible', 'JJ'), ('on', 'IN'), ('a', 'DT'), ('mid-range', 'JJ'), ('laptop', 'NN'), ('with', 'IN'), ('the', 'DT'), ('maximum', 'JJ'), ('capability', 'NN'), ('of', 'IN'), ('8GB', 'CD'), ('of', 'IN'), ('Ram', 'NNP'), (',', ','), ('2.5', 'CD'), ('GHz', 'NNP'), ('quad', 'NN'), ('core', 'NN'), ('Intel', 'NNP'), ('Core', 'NNP'), ('i', 'NN'), ('7', 'CD'), ('processor', 'NN'), (',', ','), ('500GB', 'CD'), ('of', 'IN'), ('hard', 'JJ'), ('drive', 'NN'), ('space', 'NN'), ('.', '.')], [('Documentation', 'NN'), ('of', 'IN'), ('a', 'DT'), ('particle', 'NN'), ('moving', 'VBG'), ('in', 'IN'), ('free', 'JJ'), ('space', 'NN')]], [[('D', 'NNP'), (')', ')'), ('The', 'DT'), ('Modelling', 'NNP'), ('software', 'NN'), ('which', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('used', 'VBN'), ('includes', 'VBZ'), (':', ':'), ('Matlab', 'NNP'), (',', ','), ('LabVIEW', 'NNP'), (',', ','), ('C', 'NNP'), ('with', 'IN'), ('standard', 'JJ'), ('libraries', 'NNS'), (',', ','), ('or', 'CC'), ('Python', 'NNP'), ('2', 'CD'), ('with', 'IN'), ('standard', 'JJ'), ('libraries', 'NNS'), ('.', '.')], [('This', 'DT'), ('code', 'NN'), ('was', 'VBD'), ('developed', 'VBN'), ('in', 'IN'), ('LabVIEW', 'NNP'), ('2013', 'CD'), ('32bit', 'CD'), ('.', '.')]], [[('E', 'NN'), (')', ')'), ('The', 'DT'), ('resolution', 'NN'), ('of', 'IN'), ('the', 'DT'), ('time', 'NN'), ('steps', 'NNS'), ('across', 'IN'), ('the', 'DT'), ('models', 'NNS'), ('is', 'VBZ'), ('to', 'TO'), ('be', 'VB'), ('at', 'IN'), ('0.001', 'CD'), ('seconds', 'NNS'), ('.', '.')], [('Model', 'NNP'), ('requires', 'VBZ'), ('final', 'JJ'), ('velocity', 'NN'), (',', ','), ('starting', 'VBG'), ('velocity', 'NN'), (',', ','), ('distance', 'NN'), ('and', 'CC'), ('sample', 'NN'), ('rate', 'NN'), ('to', 'TO'), ('be', 'VB'), ('known', 'VBN'), ('and', 'CC'), ('it', 'PRP'), ('calculates', 'VBZ'), ('acceleration', 'NN'), (',', ','), ('time', 'NN'), ('that', 'IN'), ('it', 'PRP'), ('takes', 'VBZ'), ('for', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('to', 'TO'), ('get', 'VB'), ('from', 'IN'), ('one', 'CD'), ('end', 'NN'), ('to', 'TO'), ('the', 'DT'), ('other', 'JJ'), (',', ','), ('time', 'NN'), ('sample', 'NN'), (',', ','), ('displacement', 'JJ'), ('steps', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('time', 'NN'), ('sample', 'NN'), ('and', 'CC'), ('the', 'DT'), ('distance', 'NN'), ('that', 'IN'), ('ball', 'NN'), ('travels', 'NNS'), ('in', 'IN'), ('a', 'DT'), ('single', 'JJ'), ('time', 'NN'), ('sample', 'NN'), ('.', '.')]], [[('F', 'NNP'), (')', ')'), ('The', 'DT'), ('output', 'NN'), ('results', 'NNS'), ('of', 'IN'), ('the',

'DT'), ('simulation', 'NN'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('saved', 'VBN'), ('in', 'IN'), ('a', 'DT'), ('file', 'NN'), ('format', 'NN'), ('that', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('interrogated', 'VBN'), ('at', 'IN'), ('a', 'DT'), ('later', 'JJ'), ('date', 'NN'), ('.', '.')], [('All', 'DT'), ('of', 'IN'), ('the', 'DT'), ('subVIs', 'NN'), ('are', 'VBP'), ('held', 'VBN'), ('within', 'IN'), ('a', 'DT'), ('common', 'JJ'), ('project', 'NN'), ('file', 'NN'), ('NLP_test_cases', 'NNS')]]]

9.6.3 ANALYSIS TWO

File inputs:

A. Requirements_For_Case_Study_One_Squash_Ball_Moving_Around_A_Court.txt

B. Documentation_of_Energy_transfer_model.txt

Outputs From NLP application:

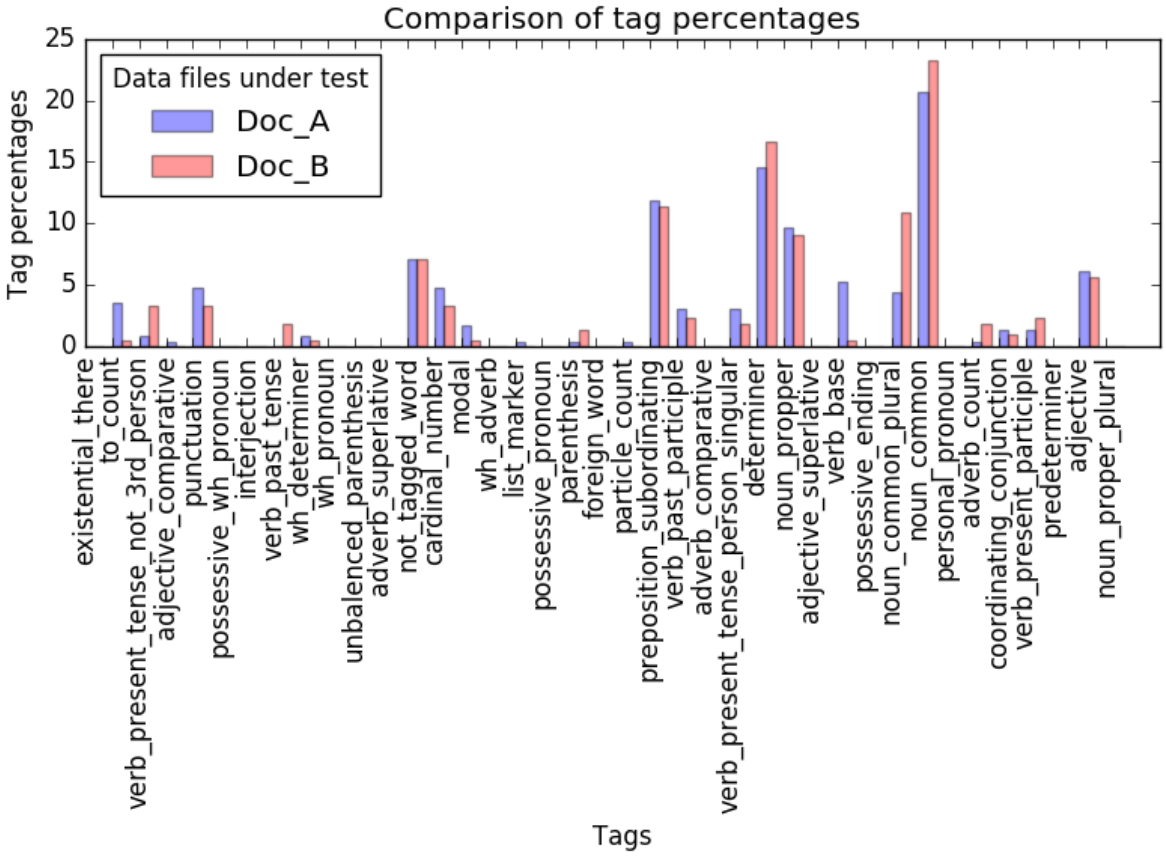

Figure 9.21: The percentage distribution of tags within the two compared documents from case study one analysis two.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 8

common_dev_enviroments = ['LabVIEW']
number of words = 1
common_identified_modeling_tuerms = []
number of common_identified_modeling_tuerms = 0
common_identified_project_tuerms = []
 number of common_identified_project_tuerms = 0
common_identified_prog_languages = []
```

number of common_identified_prog_languages = 0
common_identified_file_types = []
number of common_identified_file_types = 0

Identified_common_company_words = {
'common_identified_modeling_tuerms': [],
'number_of_identified_prog_languages': 0,
'common_identified_prog_languages': [],
'common_dev_enviroments': ['LabVIEW'],
'number_of_common_dev_enviroments': 1,
'number_of_identified_file_types': 0,
'common_identified_file_types': [],
'common_identified_project_tuerms': [],
'number_of_common_identified_modeling_tuerms': 0,
'number_of_common_identified_project_tuerms': 0
}

dict_of_words_in_both_docs = {
'verb_present_tense_person_singular_list': ['is'],
'verb_past_participle_list': [],
'list_marker_list': [],
'adverb_list': [],
'noun_propper_list': [],
'cardinal_number_list': ['7'],
'adjective_list': [],
'noun_propper_singular_list': ['LabVIEW'],
'noun_proper_plural_list': [],
'personal_pronoun_list': [],
'preposition_subordinating_list': ['in', 'with', 'of', 'by', 'on'],
'predeterminer_list': [],
'verb_present_tense_not_3rd_person_list': ['are'],
'wh_pronoun_list': [],
'parenthesis_list': [],
'particle_list': [],
'to_list': ['to'],
'adjective_comparative_list': [],
'possessive_pronoun_list': [],
'verb_past_tense_list': [],
'punctuation_list': ['.'],
'modal_list': ['can'],
'determiner_list': ['the', 'A', 'The', 'a'],
'adverb_comparative_list': [],
'foreign_word_list': [],
'possessive_wh_pronoun_list': [],
'existential_there_list': [],

'verb_base_list': [],
'adjective_superlative_list': [],
'adverb_superlative_list': [],
'possessive_ending_list': [],
'interjection_list': [],
'not_tagged_word_list': [')', ','],
'coordinating_conjunction_list': ['and'],
'noun_common_list': ['surface', 'space', 'date', 'file', 'ball', 'simulation'],
'unbalenced_parenthesis_list': [],
'wh_determiner_list': ['which'],
'wh_adverb_list': [],
'noun_common_plural_list': [],
'verb_present_participle_list': []
}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('1', 'CD'), ('(', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('squash', 'NN'), ('ball', 'NN'), ('in', 'IN'), ('flight', 'NN'), ('.', '.')], [('Energy', 'NNP'), ('transfer', 'NN'), ('model', 'NN'), ('of', 'IN'), ('a', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('a', 'DT'), ('surface', 'NN'), ('and', 'CC'), ('losing', 'VBG'), ('energy', 'NN'), ('on', 'IN'), ('impact', 'NN'), ('and', 'CC'), ('rebound', 'NN'), ('.', '.')]], [[('2', 'CD'), ('(', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('squash', 'NN'), ('ball', 'NN'), ('bouncing', 'VBG'), ('off', 'RP'), ('the', 'DT'), ('surface', 'NN'), ('of', 'IN'), ('a', 'DT'), ('wall', 'NN'), ('.', '.')], [('Energy', 'NNP'), ('transfer', 'NN'), ('model', 'NN'), ('of', 'IN'), ('a', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('a', 'DT'), ('surface', 'NN'), ('and', 'CC'), ('losing', 'VBG'), ('energy', 'NN'), ('on', 'IN'), ('impact', 'NN'), ('and', 'CC'), ('rebound', 'NN'), ('.', '.')]], [[('3', 'LS'), ('(', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('ball', 'NN'), ('interacting', 'VBG'), ('with', 'IN'), ('a', 'DT'), ('player's', 'NN'), ('racket', 'NN'), ('.', '.')], [('Energy', 'NNP'), ('transfer', 'NN'), ('model', 'NN'), ('of', 'IN'), ('a', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('a', 'DT'), ('surface', 'NN'), ('and', 'CC'), ('losing', 'VBG'), ('energy', 'NN'), ('on', 'IN'), ('impact', 'NN'), ('and', 'CC'), ('rebound', 'NN'), ('.', '.')]], [[('4', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('needs', 'VBZ'), ('to', 'TO'), ('be', 'VB'), ('run', 'VBN'), ('multiple', 'JJ'), ('times', 'NNS'), ('with', 'IN'), ('the', 'DT'), ('only', 'JJ'), ('change', 'NN'), ('being', 'VBG'), ('the', 'DT'), ('compound', 'NN'), ('of', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('.', '.')], [('Energy', 'NNP'), ('transfer', 'NN'), ('model', 'NN'), ('of', 'IN'), ('a', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('a', 'DT'), ('surface', 'NN'), ('and', 'CC'), ('losing', 'VBG'), ('energy', 'NN'), ('on', 'IN'), ('impact', 'NN'), ('and', 'CC'),

('rebound', 'NN'), ('.', '.')]], [[('5', 'CD'), (')', ')'), ('The', 'DT'), ('flight', 'NN'), ('of', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('constrained', 'VBN'), ('by', 'IN'), ('a', 'DT'), ('regulation', 'NN'), ('sized', 'VBN'), ('squash', 'JJ'), ('court', 'NN'), ('.', '.')], [('Energy', 'NNP'), ('transfer', 'NN'), ('model', 'NN'), ('of', 'IN'), ('a', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('a', 'DT'), ('surface', 'NN'), ('and', 'CC'), ('losing', 'VBG'), ('energy', 'NN'), ('on', 'IN'), ('impact', 'NN'), ('and', 'CC'), ('rebound', 'NN'), ('.', '.')]], [[('B', 'NNP'), (')', ')'), ('The', 'DT'), ('overall', 'JJ'), ('simulation', 'NN'), ('and', 'CC'), ('analysis', 'NN'), ('should', 'MD'), ('be', 'VB'), ('possible', 'JJ'), ('on', 'IN'), ('a', 'DT'), ('mid-range', 'JJ'), ('laptop', 'NN'), ('with', 'IN'), ('the', 'DT'), ('maximum', 'JJ'), ('capability', 'NN'), ('of', 'IN'), ('8GB', 'CD'), ('of', 'IN'), ('Ram', 'NNP'), (',', ','), ('2.5', 'CD'), ('GHz', 'NNP'), ('quad', 'NN'), ('core', 'NN'), ('Intel', 'NNP'), ('Core', 'NNP'), ('i', 'NN'), ('7', 'CD'), ('processor', 'NN'), (',', ','), ('500GB', 'CD'), ('of', 'IN'), ('hard', 'JJ'), ('drive', 'NN'), ('space', 'NN'), ('.', '.')], [('The', 'DT'), ('ball', 'NN'), ('can', 'MD'), ('move', 'VB'), ('freely', 'RB'), ('within', 'IN'), ('the', 'DT'), ('three', 'CD'), ('dimensional', 'JJ'), ('space', 'NN')]], [[('D', 'NNP'), (')', ')'), ('The', 'DT'), ('Modelling', 'NNP'), ('software', 'NN'), ('which', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('used', 'VBN'), ('includes', 'VBZ'), (';', ':'), ('Matlab', 'NNP'), (',', ','), ('LabVIEW', 'NNP'), (',', ','), ('C', 'NNP'), ('with', 'IN'), ('standard', 'JJ'), ('libraries', 'NNS'), (',', ','), ('or', 'CC'), ('Python', 'NNP'), ('2', 'CD'), ('with', 'IN'), ('standard', 'JJ'), ('libraries', 'NNS'), ('.', '.')], [('This', 'DT'), ('code', 'NN'), ('was', 'VBD'), ('developed', 'VBN'), ('in', 'IN'), ('LabVIEW', 'NNP'), ('2013', 'CD'), ('32bit', 'CD')]], [[('F', 'NNP'), (')', ')'), ('The', 'DT'), ('output', 'NN'), ('results', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('saved', 'VBN'), ('in', 'IN'), ('a', 'DT'), ('file', 'NN'), ('format', 'NN'), ('that', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('interrogated', 'VBN'), ('at', 'IN'), ('a', 'DT'), ('later', 'JJ'), ('date', 'NN'), ('.', '.')], [('All', 'DT'), ('of', 'IN'), ('the', 'DT'), ('subVIs', 'NN'), ('are', 'VBP'), ('held', 'VBN'), ('within', 'IN'), ('a', 'DT'), ('common', 'JJ'), ('project', 'NN'), ('file', 'NN'), ('NLP_test_cases', 'NNS')]]]

9.6.4 ANALYSIS THREE

File inputs:

A. Requirements_For_Case_Study_One_Squash_Ball_Moving_Around_A_Court.txt

B. Documentation_of_Squash_Court_in_or_Out_Model.txt

Outputs From NLP application:

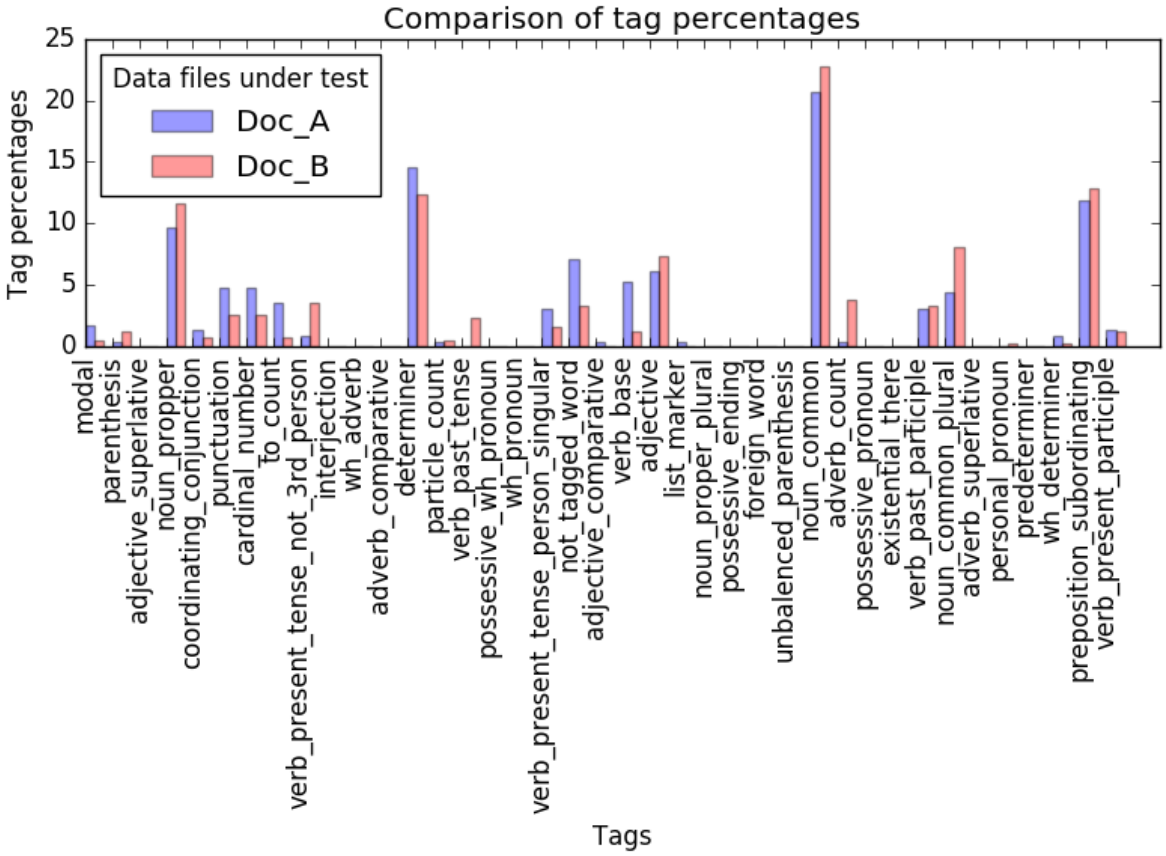

Figure 9.22: The percentage distribution of tags within the two compared documents from case study one analysis three.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 9

common_dev_enviroments = ['LabVIEW']
number of words = 1
common_identified_modeling_tuerms = []
number of common_identified_modeling_tuerms = 0
common_identified_project_tuerms = []
number of common_identified_project_tuerms = 0
common_identified_prog_languages = []
```

number of common_identified_prog_languages = 0
common_identified_file_types = []
number of common_identified_file_types = 0

Identified_common_company_words = {
'common_identified_prog_languages': [],
'number_of_common_dev_enviroments': 1,
'number_of_identified_file_types': 0,
'common_identified_project_tuerms': [],
'common_identified_modeling_tuerms': [],
'number_of_common_identified_modeling_tuerms': 0,
'number_of_identified_prog_languages': 0,
'common_dev_enviroments': ['LabVIEW'],
'number_of_common_identified_project_tuerms': 0,
'common_identified_file_types': []
}

dict_of_words_in_both_docs = {
'cardinal_number_list': ['7'],
'noun_common_list': ['court', 'date', 'space', 'squash', 'wall', 'file', 'simulation',
'ball'],
'list_marker_list': [],
'unbalenced_parenthesis_list': [],
'noun_propper_singular_list': ['LabVIEW', 'Squash'],
'interjection_list': [],
'parenthesis_list': [],
'wh_adverb_list': [],
'existential_there_list': [],
'personal_pronoun_list': [],
'particle_list': [],
'preposition_subordinating_list': ['with', 'of', 'at', 'in', 'by'],
'to_list': ['to'],
'verb_present_tense_person_singular_list': ['is'],
'wh_pronoun_list': [],
'verb_present_participle_list': ['being'],
'adverb_superlative_list': [],
'adverb_comparative_list': [],
'foreign_word_list': [],
'adverb_list': [],
'not_tagged_word_list': [')', ','],
'punctuation_list': ['.'],
'possessive_pronoun_list': [],
'possessive_wh_pronoun_list': [],
'verb_present_tense_not_3rd_person_list': ['are'],
'adjective_comparative_list': [],

'determiner_list': ['the', 'The', 'A', 'a'],
'predeterminer_list': [],
'noun_proper_plural_list': [],
'modal_list': ['can'],
'noun_propper_list': [],
'adjective_superlative_list': [],
'verb_past_tense_list': [],
'wh_determiner_list': ['which'],
'verb_past_participle_list': [],
'verb_base_list': ['be'],
'coordinating_conjunction_list':['or', 'and'],
'noun_common_plural_list': [],
'possessive_ending_list': [],
'adjective_list': ['squash']
}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('1', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('squash', 'NN'), ('ball', 'NN'), ('in', 'IN'), ('flight', 'NN'), ('.', '.')], [('The', 'DT'), ('simulation', 'NN'), ('used', 'VBD'), ('the', 'DT'), ('(', '('), ('X', 'NNP'), (',', ','), ('Y', 'NNP'), (',', ','), ('Z', 'NNP'), (')', ')'), ('Cartesian', 'JJ'), ('notation', 'NN'), ('of', 'IN'), ('three', 'CD'), ('dimensional', 'JJ'), ('space', 'NN'), ('.', '.'), ('(', '('), ('0,0,0', 'CD'), (')', ')'), ('is', 'VBZ'), ('taken', 'VBN'), ('to', 'TO'), ('be', 'VB'), ('the', 'DT'), ('bottom', 'JJ'), ('left', 'JJ'), ('corner', 'NN'), ('as', 'IN'), ('if', 'IN'), ('a', 'DT'), ('player', 'NN'), ('had', 'VBD'), ('just', 'RB'), ('walked', 'VBN'), ('through', 'IN'), ('the', 'DT'), ('door', 'NN'), ('.', '.')]], [[('2', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('squash', 'NN'), ('ball', 'NN'), ('bouncing', 'VBG'), ('off', 'RP'), ('the', 'DT'), ('surface', 'NN'), ('of', 'IN'), ('a', 'DT'), ('wall', 'NN'), ('.', '.')], [('The', 'DT'), ('simulation', 'NN'), ('used', 'VBD'), ('the', 'DT'), ('(', '('), ('X', 'NNP'), (',', ','), ('Y', 'NNP'), (',', ','), ('Z', 'NNP'), (')', ')'), ('Cartesian', 'JJ'), ('notation', 'NN'), ('of', 'IN'), ('three', 'CD'), ('dimensional', 'JJ'), ('space', 'NN'), ('.', '.'), ('(', '('), ('0,0,0', 'CD'), (')', ')'), ('is', 'VBZ'), ('taken', 'VBN'), ('to', 'TO'), ('be', 'VB'), ('the', 'DT'), ('bottom', 'JJ'), ('left', 'JJ'), ('corner', 'NN'), ('as', 'IN'), ('if', 'IN'), ('a', 'DT'), ('player', 'NN'), ('had', 'VBD'), ('just', 'RB'), ('walked', 'VBN'), ('through', 'IN'), ('the', 'DT'), ('door', 'NN'), ('.', '.')]], [[('3', 'LS'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('ball', 'NN'), ('interacting', 'VBG'), ('with', 'IN'), ('a', 'DT'), ('player's', 'NN'), ('racket', 'NN'), ('.', '.')], [('The', 'DT'), ('simulation', 'NN'), ('used', 'VBD'), ('the', 'DT'), ('(', '('), ('X', 'NNP'), (',', ','), ('Y', 'NNP'), (',', ','), ('Z', 'NNP'), (')', ')'), ('Cartesian', 'JJ'), ('notation', 'NN'), ('of', 'IN'), ('three', 'CD'), ('dimensional', 'JJ'), ('space', 'NN'), ('.', '.'), ('(', '('), ('0,0,0',

'CD'), (')', ')'), ('is', 'VBZ'), ('taken', 'VBN'), ('to', 'TO'), ('be', 'VB'), ('the', 'DT'), ('bottom', 'JJ'), ('left', 'JJ'), ('corner', 'NN'), ('as', 'IN'), ('if', 'IN'), ('a', 'DT'), ('player', 'NN'), ('had', 'VBD'), ('just', 'RB'), ('walked', 'VBN'), ('through', 'IN'), ('the', 'DT'), ('door', 'NN'), ('.', '.')]], [[('4', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('needs', 'VBZ'), ('to', 'TO'), ('be', 'VB'), ('run', 'VBN'), ('multiple', 'JJ'), ('times', 'NNS'), ('with', 'IN'), ('the', 'DT'), ('only', 'JJ'), ('change', 'NN'), ('being', 'VBG'), ('the', 'DT'), ('compound', 'NN'), ('of', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('.', '.')], [('The', 'DT'), ('simulation', 'NN'), ('used', 'VBD'), ('the', 'DT'), ('(', '('), ('X', 'NNP'), (',', ','), ('Y', 'NNP'), (',', ','), ('Z', 'NNP'), (')', ')'), ('Cartesian', 'JJ'), ('notation', 'NN'), ('of', 'IN'), ('three', 'CD'), ('dimensional', 'JJ'), ('space', 'NN'), ('.', '.'), ('(', '('), ('0,0,0', 'CD'), (')', ')'), ('is', 'VBZ'), ('taken', 'VBN'), ('to', 'TO'), ('be', 'VB'), ('the', 'DT'), ('bottom', 'JJ'), ('left', 'JJ'), ('corner', 'NN'), ('as', 'IN'), ('if', 'IN'), ('a', 'DT'), ('player', 'NN'), ('had', 'VBD'), ('just', 'RB'), ('walked', 'VBN'), ('through', 'IN'), ('the', 'DT'), ('door', 'NN'), ('.', '.')]], [[('5', 'CD'), (')', ')'), ('The', 'DT'), ('flight', 'NN'), ('of', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('constrained', 'VBN'), ('by', 'IN'), ('a', 'DT'), ('regulation', 'NN'), ('sized', 'VBN'), ('squash', 'JJ'), ('court', 'NN'), ('.', '.')], [('Any', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('any', 'DT'), ('of', 'IN'), ('the', 'DT'), ('lines', 'NNS'), ('is', 'VBZ'), ('considered', 'VBN'), ('“in”', 'NNS')]], [[('A', 'DT'), (')', ')'), ('The', 'DT'), ('total', 'JJ'), ('run', 'NN'), ('time', 'NN'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('should', 'MD'), ('take', 'VB'), ('less', 'JJR'), ('than', 'IN'), ('five', 'CD'), ('minutes', 'NNS'), ('to', 'TO'), ('fully', 'RB'), ('execute', 'VB'), ('.', '.')], [('The', 'DT'), ('simulation', 'NN'), ('used', 'VBD'), ('the', 'DT'), ('(', '('), ('X', 'NNP'), (',', ','), ('Y', 'NNP'), (',', ','), ('Z', 'NNP'), (')', ')'), ('Cartesian', 'JJ'), ('notation', 'NN'), ('of', 'IN'), ('three', 'CD'), ('dimensional', 'JJ'), ('space', 'NN'), ('.', '.'), ('(', '('), ('0,0,0', 'CD'), (')', ')'), ('is', 'VBZ'), ('taken', 'VBN'), ('to', 'TO'), ('be', 'VB'), ('the', 'DT'), ('bottom', 'JJ'), ('left', 'JJ'), ('corner', 'NN'), ('as', 'IN'), ('if', 'IN'), ('a', 'DT'), ('player', 'NN'), ('had', 'VBD'), ('just', 'RB'), ('walked', 'VBN'), ('through', 'IN'), ('the', 'DT'), ('door', 'NN'), ('.', '.')]], [[('B', 'NNP'), (')', ')'), ('The', 'DT'), ('overall', 'JJ'), ('simulation', 'NN'), ('and', 'CC'), ('analysis', 'NN'), ('should', 'MD'), ('be', 'VB'), ('possible', 'JJ'), ('on', 'IN'), ('a', 'DT'), ('mid-range', 'JJ'), ('laptop', 'NN'), ('with', 'IN'), ('the', 'DT'), ('maximum', 'JJ'), ('capability', 'NN'), ('of', 'IN'), ('8GB', 'CD'), ('of', 'IN'), ('Ram', 'NNP'), (',', ','), ('2.5', 'CD'), ('GHz', 'NNP'), ('quad', 'NN'), ('core', 'NN'), ('Intel', 'NNP'), ('Core', 'NNP'), ('i', 'NN'), ('7', 'CD'), ('processor', 'NN'), (',', ','), ('500GB', 'CD'), ('of', 'IN'), ('hard', 'JJ'), ('drive', 'NN'), ('space', 'NN'), ('.', '.')], [('The', 'DT'), ('simulation', 'NN'), ('used', 'VBD'), ('the', 'DT'), ('(', '('), ('X', 'NNP'), (',', ','), ('Y', 'NNP'), (',', ','), ('Z', 'NNP'), (')', ')'), ('Cartesian', 'JJ'), ('notation', 'NN'), ('of', 'IN'), ('three', 'CD'), ('dimensional', 'JJ'), ('space', 'NN'), ('.', '.'), ('(', '('), ('0,0,0', 'CD'), (')', ')'), ('is', 'VBZ'), ('taken', 'VBN'), ('to', 'TO'), ('be', 'VB'), ('the', 'DT'), ('bottom', 'JJ'), ('left', 'JJ'), ('corner', 'NN'), ('as', 'IN'), ('if', 'IN'), ('a', 'DT'), ('player', 'NN'), ('had', 'VBD'), ('just', 'RB'), ('walked', 'VBN'), ('through', 'IN'), ('the', 'DT'), ('door', 'NN'), ('.', '.')]], [[('D', 'NNP'), (')', ')'), ('The', 'DT'), ('Modelling', 'NNP'), ('software', 'NN'), ('which', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('used', 'VBN'), ('includes', 'VBZ'), (';', ':'), ('Matlab', 'NNP'), (',', ','), ('LabVIEW', 'NNP'), (',', ','), ('C', 'NNP'), ('with', 'IN'), ('standard', 'JJ'), ('libraries', 'NNS'), (',', ','), ('or', 'CC'), ('Python', 'NNP'), ('2', 'CD'), ('with', 'IN'), ('standard', 'JJ'), ('libraries', 'NNS'), ('.', '.')], [('This', 'DT'), ('code', 'NN'), ('was', 'VBD'), ('developed', 'VBN'), ('in', 'IN'), ('LabVIEW', 'NNP'), ('2013', 'CD'),

('32bit', 'CD')]], [[('F', 'NNP'), (')', ')'), ('The', 'DT'), ('output', 'NN'), ('results', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('saved', 'VBN'), ('in', 'IN'), ('a', 'DT'), ('file', 'NN'), ('format', 'NN'), ('that', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('interrogated', 'VBN'), ('at', 'IN'), ('a', 'DT'), ('later', 'JJ'), ('date', 'NN'), ('.', '.')], [('The', 'DT'), ('simulation', 'NN'), ('used', 'VBD'), ('the', 'DT'), ('(', '('), ('X', 'NNP'), (',', ','), ('Y', 'NNP'), (',', ','), ('Z', 'NNP'), (')', ')'), ('Cartesian', 'JJ'), ('notation', 'NN'), ('of', 'IN'), ('three', 'CD'), ('dimensional', 'JJ'), ('space', 'NN'), ('.', '.'), ('(', '('), ('0,0,0', 'CD'), (')', ')'), ('is', 'VBZ'), ('taken', 'VBN'), ('to', 'TO'), ('be', 'VB'), ('the', 'DT'), ('bottom', 'JJ'), ('left', 'JJ'), ('corner', 'NN'), ('as', 'IN'), ('if', 'IN'), ('a', 'DT'), ('player', 'NN'), ('had', 'VBD'), ('just', 'RB'), ('walked', 'VBN'), ('through', 'IN'), ('the', 'DT'), ('door', 'NN'), ('.', '.')]]]

9.6.5 ANALYSIS FOUR

File inputs:

A. Documentation_of_a_Particle_Moving_in_Free_Space.txt

B. Documentation_of_Energy_transfer_model.txt

Outputs From NLP application:

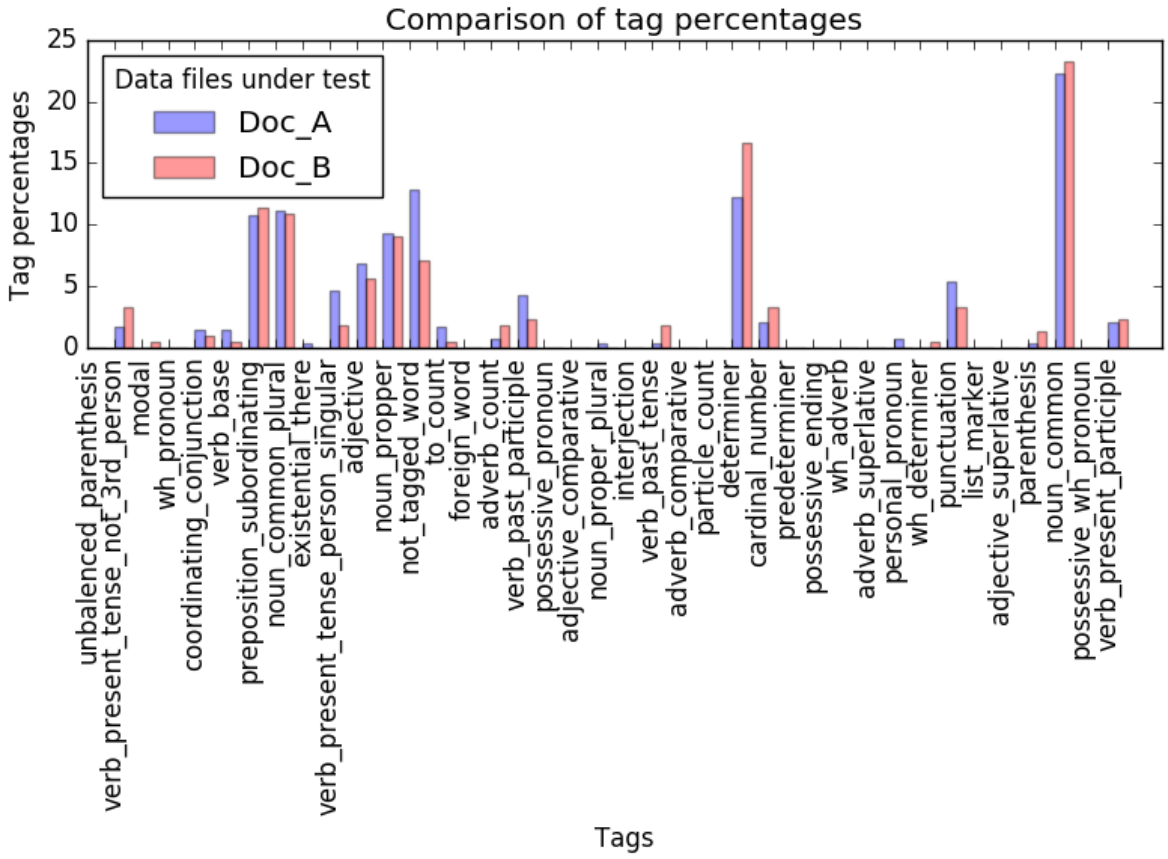

Figure 9.23: The percentage distribution of tags within the two compared documents from case study one analysis four.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 17
common_dev_enviroments = ['LabVIEW']
number of words = 1
common_identified_modeling_tuerms = ['inputs', 'inputs', 'outputs']
number of common_identified_modeling_tuerms = 3
common_identified_project_tuerms = []
number of common_identified_project_tuerms = 3
common_identified_prog_languages = []
number of common_identified_prog_languages = 0
common_identified_file_types = []
```

number of common_identified_file_types = 0

Identified_common_company_words = {
'common_identified_file_types': [], 'number_of_identified_prog_languages': 0,
'number_of_common_identified_project_tuerms': 0,
'number_of_common_dev_enviroments': 1,
'common_identified_prog_languages': [],
'common_identified_modeling_tuerms': ['inputs', 'inputs', 'outputs'],
'common_dev_enviroments': ['LabVIEW'],
'number_of_identified_file_types': 0,
'common_identified_project_tuerms': [],
'number_of_common_identified_modeling_tuerms': 3}

dict_of_words_in_both_docs = {
'adjective_superlative_list': [],
'adverb_comparative_list': [],
'preposition_subordinating_list': ['of', 'throughout', 'from', 'per', 'within', 'in'],
'to_list': ['to'],
'predeterminer_list': [],
'verb_present_tense_person_singular_list': ['is'],
'determiner_list': ['No', 'the', 'all', 'This', 'a', 'The', 'no', 'All'],
'noun_propper_singular_list': ['Written', 'March', 'SI', 'LabVIEW', 'Velocity'],
'wh_adverb_list': [],
'existential_there_list': [],
'interjection_list': [],
'adjective_list': ['common', 'second'],
'adjective_comparative_list': [],
'list_marker_list': [],
'noun_proper_plural_list': [],
'adverb_superlative_list': [],
'personal_pronoun_list': [],
'verb_present_tense_not_3rd_person_list': ['are'],
'verb_base_list': [],
'verb_present_participle_list': [],
'possessive_ending_list': [],
'verb_past_participle_list': ['developed', 'made', 'held'],
'noun_common_list': ['particle', 'Structure', 'space', 'resistance', 'model',
'Creation', 'ball', 'Documentation', 'date', 'project', 'velocity', 'file', 'subVIs',
'code'],
'adverb_list': ['only'],
'unbalenced_parenthesis_list': [],
'noun_propper_list': [],
'foreign_word_list': [],
'punctuation_list': ['.'],
'wh_pronoun_list': [],

'parenthesis_list': [':'],
'cardinal_number_list': ['2015', '7', '32bit', '2013'],
'wh_determiner_list': [],
'noun_common_plural_list': ['Assumptions', 'mps', 'toolkits', 'NLP_test_cases',
'modules', 'Outputs', 'meters', 'inputs', 'outputs', 'units', 'numbers', 'windows'],
'possessive_wh_pronoun_list': [],
'verb_past_tense_list': ['was'],
'coordinating_conjunction_list': ['and'],
'modal_list': [],
'possessive_pronoun_list': [],
'not_tagged_word_list': [',', ')', '('],
'particle_list': []
}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('Documentation', 'NN'), ('of', 'IN'), ('a', 'DT'), ('particle', 'NN'), ('moving', 'VBG'), ('in', 'IN'), ('free', 'JJ'), ('space', 'NN')], [('The', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('particle', 'NN')]], [[('This', 'DT'), ('code', 'NN'), ('was', 'VBD'), ('developed', 'VBN'), ('in', 'IN'), ('LabVIEW', 'NNP'), ('2013', 'CD'), ('32bit', 'CD'), ('.', '.')], [('This', 'DT'), ('code', 'NN'), ('was', 'VBD'), ('developed', 'VBN'), ('in', 'IN'), ('LabVIEW', 'NNP'), ('2013', 'CD'), ('32bit', 'CD')]], [[('No', 'DT'), ('toolkits', 'NNS'), ('needed', 'VBN'), ('.', '.')], [('No', 'DT'), ('toolkits', 'NNS'), ('needed', 'VBD')]], [[('No', 'DT'), ('modules', 'NNS'), ('needed', 'VBN'), ('.', '.')], [('No', 'DT'), ('modules', 'NNS'), ('needed', 'VBD')]], [[('Model', 'NNP'), ('requires', 'VBZ'), ('final', 'JJ'), ('velocity', 'NN'), (',', ','), ('starting', 'VBG'), ('velocity', 'NN'), (',', ','), ('distance', 'NN'), ('and', 'CC'), ('sample', 'NN'), ('rate', 'NN'), ('to', 'TO'), ('be', 'VB'), ('known', 'VBN'), ('and', 'CC'), ('it', 'PRP'), ('calculates', 'VBZ'), ('acceleration', 'NN'), (',', ','), ('time', 'NN'), ('that', 'IN'), ('it', 'PRP'), ('takes', 'VBZ'), ('for', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('to', 'TO'), ('get', 'VB'), ('from', 'IN'), ('one', 'CD'), ('end', 'NN'), ('to', 'TO'), ('the', 'DT'), ('other', 'JJ'), (',', ','), ('time', 'NN'), ('sample', 'NN'), (',', ','), ('displacement', 'JJ'), ('steps', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('time', 'NN'), ('sample', 'NN'), ('and', 'CC'), ('the', 'DT'), ('distance', 'NN'), ('that', 'IN'), ('ball', 'NN'), ('travels', 'NNS'), ('in', 'IN'), ('a', 'DT'), ('single', 'JJ'), ('time', 'NN'), ('sample', 'NN'), ('.', '.')], [('Returning', 'VBG'), ('velocity', 'NN'), ('(', '('), ('mps', 'NNS'), (')', ')')]], [[('First', 'RB'), ('the', 'DT'), ('acceleration', 'NN'), ('of', 'IN'), ('the', 'DT'), ('particle', 'NN'), ('is', 'VBZ'), ('calculated', 'VBN'), ('.', '.')], [('The', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('particle', 'NN')]], [[('Second', 'IN'), ('the', 'DT'), ('time', 'NN'), ('for', 'IN'), ('the', 'DT'), ('particle', 'NN'), ('to', 'TO'), ('complete', 'VB'), ('the', 'DT'), ('distance', 'NN'), ('is', 'VBZ'), ('calculated', 'VBN'), ('.', '.')], [('The', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('particle', 'NN')]], [[('Assumptions', 'NNS'), ('made', 'VBN'), (':', ':')], [('Assumptions', 'NNS'), ('made', 'VBN'), (':', ':')]], [[('Simulation', 'NN'), ('uses', 'VBZ'), ('Newtonian', 'JJ'), ('equations',

'NNS'), ('of', 'IN'), ('motion', 'NN'), ('.', '.')], [('Simulation', 'NNP'), ('only', 'RB'), ('runs', 'VBZ'), ('once', 'RB'), (',', ','), ('no', 'DT'), ('loops', 'NNS'), ('.', '.')]], [[('The', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('considered', 'VBN'), ('to', 'TO'), ('be', 'VB'), ('a', 'DT'), ('particle', 'NN'), ('.', '.')], [('Energy', 'NNP'), ('transfer', 'NN'), ('model', 'NN'), ('of', 'IN'), ('a', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('a', 'DT'), ('surface', 'NN'), ('and', 'CC'), ('losing', 'VBG'), ('energy', 'NN'), ('on', 'IN'), ('impact', 'NN'), ('and', 'CC'), ('rebound', 'NN'), ('.', '.')]], [[('All', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are', 'VBP'), ('known', 'VBN'), ('values', 'NNS'), ('.', '.')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are', 'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS'), ('.', '.'), ('Input', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('metric', 'JJ'), ('.', '.')]], [[('The', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('only', 'RB'), ('moving', 'VBG'), ('in', 'IN'), ('one', 'CD'), ('plane', 'NN'), ('.', '.')], [('Energy', 'NNP'), ('transfer', 'NN'), ('model', 'NN'), ('of', 'IN'), ('a', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('a', 'DT'), ('surface', 'NN'), ('and', 'CC'), ('losing', 'VBG'), ('energy', 'NN'), ('on', 'IN'), ('impact', 'NN'), ('and', 'CC'), ('rebound', 'NN'), ('.', '.')]], [[('SI', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('used', 'VBN'), ('throughout', 'IN'), ('meters', 'NNS'), (',', ','), ('seconds', 'NNS'), (',', ','), ('meters', 'NNS'), ('per', 'IN'), ('second', 'NN'), (',', ','), ('and', 'CC'), ('meters', 'NNS'), ('per', 'IN'), ('second', 'JJ'), ('per', 'IN'), ('second', 'NN')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are', 'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS'), ('.', '.'), ('Input', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('metric', 'JJ'), ('.', '.')]], [[('The', 'DT'), ('following', 'JJ'), ('inputs', 'NNS'), ('are', 'VBP'), ('all', 'DT'), ('floating', 'VBG'), ('points', 'NNS')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are', 'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS'), ('.', '.'), ('Input', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('metric', 'JJ'), ('.', '.')]], [[('Starting', 'VBG'), ('Velocity', 'NNP'), ('U', 'NNP'), ('(', '('), ('mps', 'NNS'), (')', ')')], [('Returning', 'VBG'), ('velocity', 'NN'), ('(', '('), ('mps', 'NNS'), (')', ')')]], [[('The', 'DT'), ('following', 'JJ'), ('outputs', 'NNS'), ('are', 'VBP'), ('single', 'JJ'), ('floating', 'VBG'), ('point', 'NN'), ('numbers', 'NNS'), ('.', '.')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('outputs', 'NNS'), ('are', 'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS')]], [[('All', 'DT'), ('of', 'IN'), ('the', 'DT'), ('subVIs', 'NN'), ('are', 'VBP'), ('held', 'VBN'), ('within', 'IN'), ('a', 'DT'), ('common', 'JJ'), ('project', 'NN'), ('file', 'NN'), ('NLP_test_cases', 'NNS')], [('All', 'DT'), ('of', 'IN'), ('the', 'DT'), ('subVIs', 'NN'), ('are', 'VBP'), ('held', 'VBN'), ('within', 'IN'), ('a', 'DT'), ('common', 'JJ'), ('project', 'NN'), ('file', 'NN'), ('NLP_test_cases', 'NNS')]]]

9.6.6 ANALYSIS FIVE

File inputs:

A. Documentation_of_a_Particle_Moving_in_Free_Space.txt

B. Documentation_of_Squash_Court_in_or_Out_Model.txt

Outputs From NLP application:

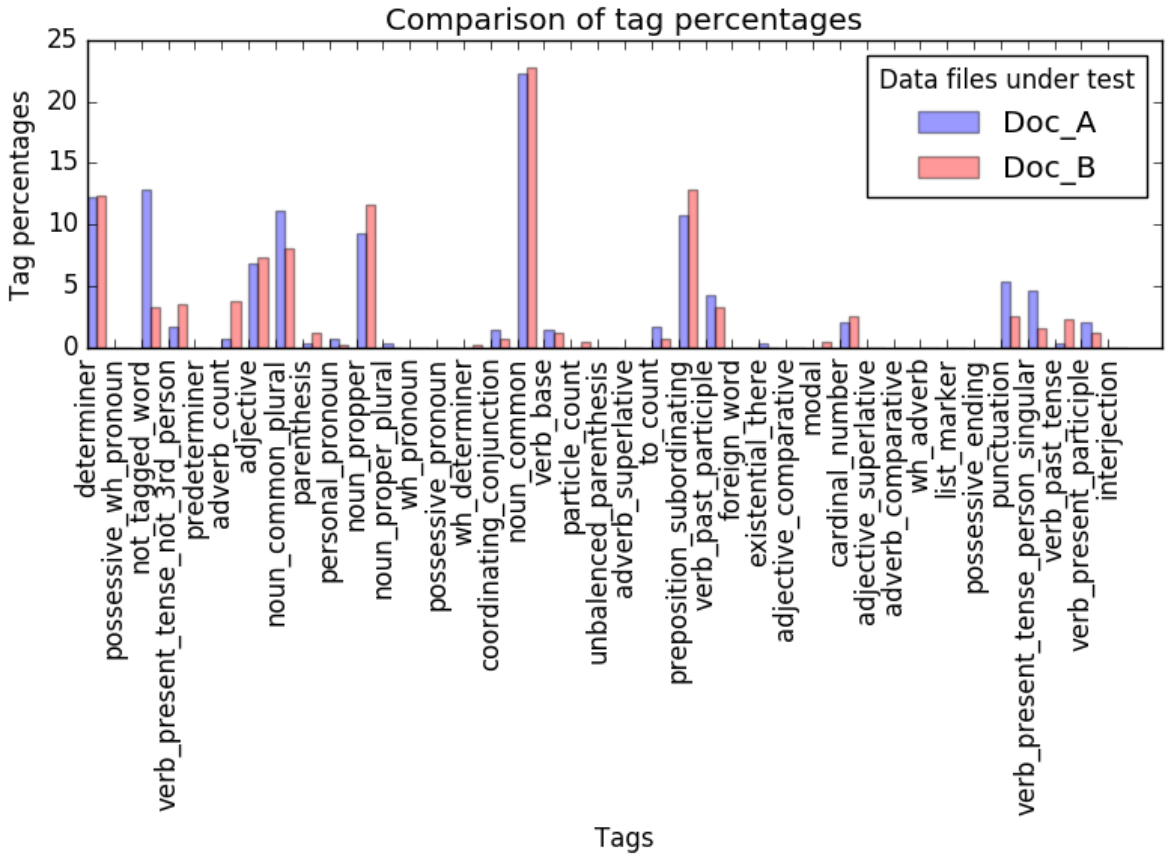

Figure 9.24: The percentage distribution of tags within the two compared documents from case study one analysis five.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 18
common_dev_enviroments = ['LabVIEW']
number of words = 1
common_identified_modeling_tuerms = ['inputs', 'inputs', 'outputs']
number of common_identified_modeling_tuerms = 3
common_identified_project_tuerms = []
number of common_identified_project_tuerms = 3
common_identified_prog_languages = []
number of common_identified_prog_languages = 0
common_identified_file_types = []
```

number of common_identified_file_types = 0

Identified_common_company_words = {
'number_of_common_dev_enviroments': 1,
'common_identified_project_tuerms': [],
'common_identified_prog_languages': [],
'number_of_identified_prog_languages': 0,
'common_dev_enviroments': ['LabVIEW'],
'common_identified_file_types': [],
'number_of_identified_file_types': 0,
'number_of_common_identified_project_tuerms': 0,
'common_identified_modeling_tuerms': ['inputs', 'inputs', 'outputs'],
'number_of_common_identified_modeling_tuerms': 3
}

dict_of_words_in_both_docs = {
'possessive_ending_list': [],
'foreign_word_list': [],
'wh_pronoun_list': [],
'verb_past_tense_list': ['was'],
'interjection_list': [],
'particle_list': [],
'parenthesis_list': [':'],
'wh_adverb_list': [],
'existential_there_list': [],
'determiner_list': ['No', 'no', 'The', 'a', 'the', 'This', 'all', 'All'],
'preposition_subordinating_list': ['for', 'in', 'of', 'from', 'throughout', 'within'],
'noun_proper_plural_list': [],
'noun_common_list': ['code', 'model', 'Creation', 'ball', 'space', 'project', 'file',
'subVIs', 'particle', 'Structure', 'point', 'Documentation', 'date'],
'wh_determiner_list': [],
'modal_list': [],
'adjective_list': ['common'],
'adjective_comparative_list': [],
'verb_base_list': ['be'],
'adverb_superlative_list': [],
'possessive_wh_pronoun_list': [],
'punctuation_list': ['.'],
'adjective_superlative_list': [],
'unbalenced_parenthesis_list': [],
'adverb_list': ['only'],
'adverb_comparative_list': [],
'noun_common_plural_list': ['NLP_test_cases', 'Assumptions', 'modules', 'to
olkits', 'windows', 'units', 'numbers', 'meters', 'outputs', 'inputs', 'Outputs', 'Inputs'],
'not_tagged_word_list': [',', '(', ')'],

'to_list': ['to'],
'verb_present_tense_not_3rd_person_list': ['are'],
'cardinal_number_list': ['2015', '32bit', '2013', '7'],
'predeterminer_list': [],
'possessive_pronoun_list': [],
'verb_present_tense_person_singular_list': ['is'],
'personal_pronoun_list': [],
list_marker_list': [],
'coordinating_conjunction_list': ['and'],
'noun_propper_list': [],
'verb_past_participle_list': ['developed', 'held', 'considered', 'made'],
'noun_propper_singular_list': ['Boolean', 'Written', 'LabVIEW', 'March'],
'verb_present_participle_list': []
}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('Documentation', 'NN'), ('of', 'IN'), ('a', 'DT'), ('particle', 'NN'), ('moving', 'VBG'), ('in', 'IN'), ('free', 'JJ'), ('space', 'NN')], [('The', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('particle', 'NN')]], [[('This', 'DT'), ('code', 'NN'), ('was', 'VBD'), ('developed', 'VBN'), ('in', 'IN'), ('LabVIEW', 'NNP'), ('2013', 'CD'), ('32bit', 'CD'), ('.', '.')], [('This', 'DT'), ('code', 'NN'), ('was', 'VBD'), ('developed', 'VBN'), ('in', 'IN'), ('LabVIEW', 'NNP'), ('2013', 'CD'), ('32bit', 'CD')]], [[('No', 'DT'), ('toolkits', 'NNS'), ('needed', 'VBN'), ('.', '.')], [('No', 'DT'), ('toolkits', 'NNS'), ('needed', 'VBD')]], [[('No', 'DT'), ('modules', 'NNS'), ('needed', 'VBN'), ('.', '.')], [('No', 'DT'), ('modules', 'NNS'), ('needed', 'VBD')]], [[('Model', 'NNP'), ('requires', 'VBZ'), ('final', 'JJ'), ('velocity', 'NN'), (',', ','), ('starting', 'VBG'), ('velocity', 'NN'), (',', ','), ('distance', 'NN'), ('and', 'CC'), ('sample', 'NN'), ('rate', 'NN'), ('to', 'TO'), ('be', 'VB'), ('known', 'VBN'), ('and', 'CC'), ('it', 'PRP'), ('calculates', 'VBZ'), ('acceleration', 'NN'), (',', ','), ('time', 'NN'), ('that', 'IN'), ('it', 'PRP'), ('takes', 'VBZ'), ('for', 'IN'), ('the', 'DT'), ('ball', 'NN'), ('to', 'TO'), ('get', 'VB'), ('from', 'IN'), ('one', 'CD'), ('end', 'NN'), ('to', 'TO'), ('the', 'DT'), ('other', 'JJ'), (',', ','), ('time', 'NN'), ('sample', 'NN'), (',', ','), ('displacement', 'JJ'), ('steps', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('time', 'NN'), ('sample', 'NN'), ('and', 'CC'), ('the', 'DT'), ('distance', 'NN'), ('that', 'IN'), ('ball', 'NN'), ('travels', 'NNS'), ('in', 'IN'), ('a', 'DT'), ('single', 'JJ'), ('time', 'NN'), ('sample', 'NN'), ('.', '.')], [('Any', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('any', 'DT'), ('of', 'IN'), ('the', 'DT'), ('lines', 'NNS'), ('is', 'VBZ'), ('considered', 'VBN'), ('“in”', 'NNS')]], [[('First', 'RB'), ('the', 'DT'), ('acceleration', 'NN'), ('of', 'IN'), ('the', 'DT'), ('particle', 'NN'), ('is', 'VBZ'), ('calculated', 'VBN'), ('.', '.')], [('The', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('particle', 'NN')]], [[('Second', 'IN'), ('the', 'DT'), ('time', 'NN'), ('for', 'IN'), ('the', 'DT'), ('particle', 'NN'), ('to', 'TO'), ('complete', 'VB'), ('the', 'DT'), ('distance', 'NN'), ('is', 'VBZ'), ('calculated', 'VBN'), ('.', '.')], [('The', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('particle', 'NN')]],

[[('Assumptions', 'NNS'), ('made', 'VBN'), (':', ':')], [('Assumptions', 'NNS'), ('made', 'VBN'), (':', ':')]], [[('Simulation', 'NN'), ('uses', 'VBZ'), ('Newtonian', 'JJ'), ('equations', 'NNS'), ('of', 'IN'), ('motion', 'NN'), ('.', '.')], [('Simulation', 'NNP'), ('only', 'RB'), ('runs', 'VBZ'), ('once', 'RB'), (',', ','), ('no', 'DT'), ('loops', 'NNS'), ('.', '.')]], [[('The', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('considered', 'VBN'), ('to', 'TO'), ('be', 'VB'), ('a', 'DT'), ('particle', 'NN'), ('.', '.')], [('Any', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('any', 'DT'), ('of', 'IN'), ('the', 'DT'), ('lines', 'NNS'), ('is', 'VBZ'), ('considered', 'VBN'), ('“in”', 'NNS')]], [[('All', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are', 'VBP'), ('known', 'VBN'), ('values', 'NNS'), ('.', '.')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are', 'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS'), ('.', '.'), ('Input', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('metric', 'JJ'), ('.', '.'), ('Inputs', 'NNS'), ('can', 'MD'), ('be', 'VB'), ('in', 'IN'), ('meters', 'NNS'), ('or', 'CC'), ('centimetres', 'NNS'), ('as', 'RB'), ('long', 'RB'), ('as', 'IN'), ('they', 'PRP'), ('are', 'VBP'), ('consistent', 'JJ'), ('throughout', 'IN'), ('.', '.')]], [[('The', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('only', 'RB'), ('moving', 'VBG'), ('in', 'IN'), ('one', 'CD'), ('plane', 'NN'), ('.', '.')], [('Any', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('any', 'DT'), ('of', 'IN'), ('the', 'DT'), ('lines', 'NNS'), ('is', 'VBZ'), ('considered', 'VBN'), ('“in”', 'NNS')]], [[('SI', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('used', 'VBN'), ('throughout', 'IN'), ('meters', 'NNS'), (',', ','), ('seconds', 'NNS'), (',', ','), ('meters', 'NNS'), ('per', 'IN'), ('second', 'NN'), (',', ','), ('and', 'CC'), ('meters', 'NNS'), ('per', 'IN'), ('second', 'JJ'), ('per', 'IN'), ('second', 'NN')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are', 'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS'), ('.', '.'), ('Input', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('metric', 'JJ'), ('.', '.'), ('Inputs', 'NNS'), ('can', 'MD'), ('be', 'VB'), ('in', 'IN'), ('meters', 'NNS'), ('or', 'CC'), ('centimetres', 'NNS'), ('as', 'RB'), ('long', 'RB'), ('as', 'IN'), ('they', 'PRP'), ('are', 'VBP'), ('consistent', 'JJ'), ('throughout', 'IN'), ('.', '.')]], [[('The', 'DT'), ('following', 'JJ'), ('inputs', 'NNS'), ('are', 'VBP'), ('all', 'DT'), ('floating', 'VBG'), ('points', 'NNS')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are', 'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS'), ('.', '.'), ('Input', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('metric', 'JJ'), ('.', '.'), ('Inputs', 'NNS'), ('can', 'MD'), ('be', 'VB'), ('in', 'IN'), ('meters', 'NNS'), ('or', 'CC'), ('centimetres', 'NNS'), ('as', 'RB'), ('long', 'RB'), ('as', 'IN'), ('they', 'PRP'), ('are', 'VBP'), ('consistent', 'JJ'), ('throughout', 'IN'), ('.', '.')]], [[('The', 'DT'), ('stop', 'NN'), ('loop', 'NN'), ('input', 'NN'), ('is', 'VBZ'), ('type', 'JJ'), ('Boolean', 'NNP')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('outputs', 'NNS'), ('are', 'VBP'), ('both', 'DT'), ('Boolean', 'NNP'), ('and', 'CC'), ('double-precision', 'NN'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS')]], [[('The', 'DT'), ('following', 'JJ'), ('outputs', 'NNS'), ('are', 'VBP'), ('single', 'JJ'), ('floating', 'VBG'), ('point', 'NN'), ('numbers', 'NNS'), ('.', '.')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('outputs', 'NNS'), ('are', 'VBP'), ('both', 'DT'), ('Boolean', 'NNP'), ('and', 'CC'), ('double-precision', 'NN'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS')]], [[('The', 'DT'), ('displacement', 'JJ'), ('step', 'NN'), ('St', 'NNP'), ('(', '('), ('meters', 'NNS'), (')', ')'), ('is', 'VBZ'), ('an', 'DT'), ('Array', 'NN')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are',

'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS'), ('.', '.'), ('Input', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('metric', 'JJ'), ('.', '.'), ('Inputs', 'NNS'), ('can', 'MD'), ('be', 'VB'), ('in', 'IN'), ('meters', 'NNS'), ('or', 'CC'), ('centimetres', 'NNS'), ('as', 'RB'), ('long', 'RB'), ('as', 'IN'), ('they', 'PRP'), ('are', 'VBP'), ('consistent', 'JJ'), ('throughout', 'IN'), ('.', '.')]], [[('All', 'DT'), ('of', 'IN'), ('the', 'DT'), ('subVIs', 'NN'), ('are', 'VBP'), ('held', 'VBN'), ('within', 'IN'), ('a', 'DT'), ('common', 'JJ'), ('project', 'NN'), ('file', 'NN'), ('NLP_test_cases', 'NNS')], [('All', 'DT'), ('of', 'IN'), ('the', 'DT'), ('subVIs', 'NN'), ('are', 'VBP'), ('held', 'VBN'), ('within', 'IN'), ('a', 'DT'), ('common', 'JJ'), ('project', 'NN'), ('file', 'NN'), ('NLP_test_cases', 'NNS')]]]

9.6.7 ANALYSIS SIX

File inputs:

A. Documentation_of_Energy_transfer_model.txt

B. Documentation_of_Squash_Court_in_or_Out_Model.txt

Outputs From NLP application:

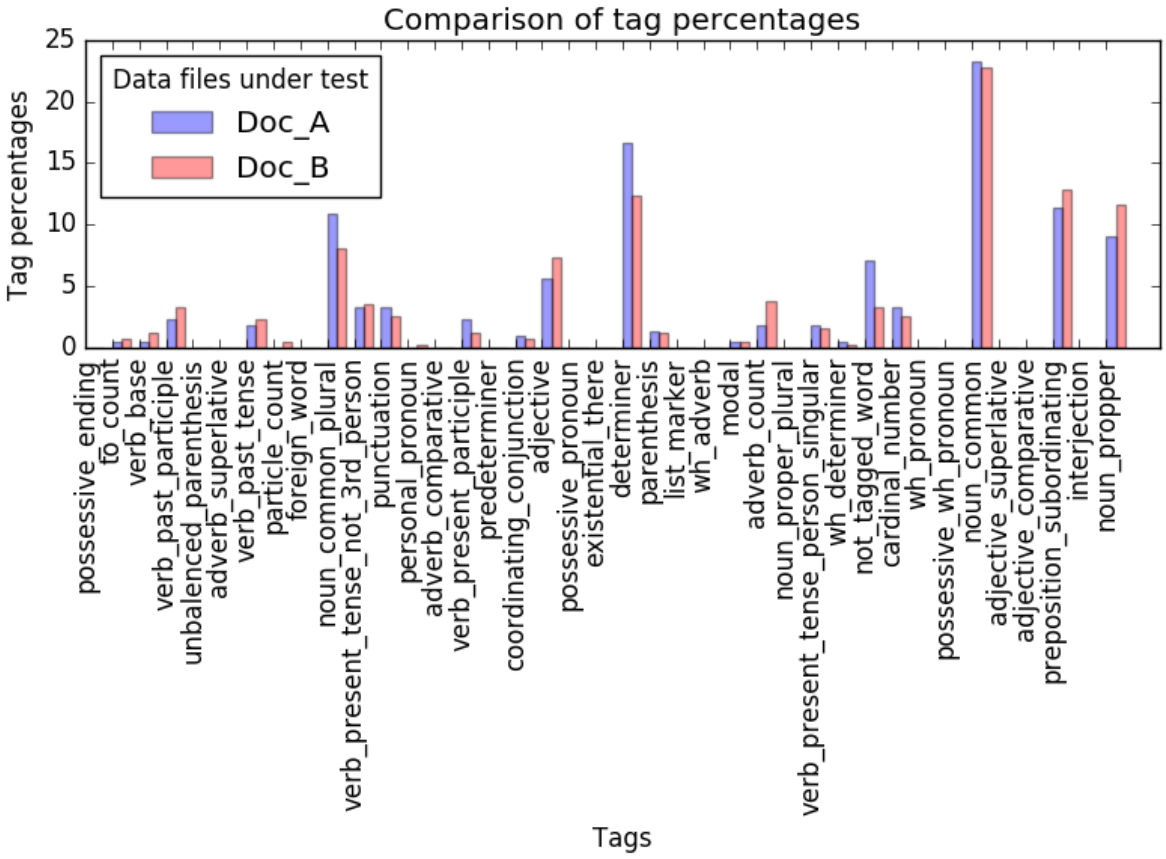

Figure 9.25: The percentage distribution of tags within the two compared documents from case study one analysis six.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 15
common_dev_enviroments = ['LabVIEW']
number of words = 1
common_identified_modeling_tuerms = ['inputs', 'outputs']
number of common_identified_modeling_tuerms = 2
common_identified_project_tuerms = []
number of common_identified_project_tuerms = 2
common_identified_prog_languages = []
number of common_identified_prog_languages = 0
common_identified_file_types = []
```

number of common_identified_file_types = 0

Identified_common_company_words = {
'number_of_identified_prog_languages': 0,
'common_identified_file_types': [],
'common_identified_project_tuerms': [],
'common_identified_prog_languages': [],
'number_of_identified_file_types': 0,
'common_identified_modeling_tuerms': ['inputs', 'outputs'],
'number_of_common_identified_modeling_tuerms': 2,
'number_of_common_dev_enviroments': 1,
'common_dev_enviroments': ['LabVIEW'],
'number_of_common_identified_project_tuerms': 0
}

dict_of_words_in_both_docs = {
'adjective_comparative_list': [],
'adjective_superlative_list': [],
'adverb_comparative_list': [],
'possessive_ending_list': [],
'modal_list': ['can'],
'to_list': ['to'],
'parenthesis_list': [':'],
'verb_past_tense_list': ['needed', 'was'],
'list_marker_list': [],
'possessive_pronoun_list': [],
'wh_pronoun_list': [],
'adjective_list': ['metric', 'common', 'dimensional', 'smooth', 'floating-point',
'double-precision', 'main'],
'foreign_word_list': [],
'possessive_wh_pronoun_list': [],
'verb_present_tense_not_3rd_person_list': ['are'],
'verb_base_list': ['move'],
'noun_propper_singular_list': ['Written','VI', 'Input', 'March', 'LabVIEW', 'VIs',
'Simulation'],
'noun_common_list': ['ball', 'simulation', 'project', 'particle', 'file', 'model', 'data',
'space', 'subVIs', 'list', 'Structure', 'code', 'Creation', 'sub', 'date',
'Documentation'],
'particle_list': [],
'noun_common_plural_list': ['outputs', 'protrusions', 'numbers', 'meters', 'surfaces',
'Assumptions', 'units', 'inputs', 'loops', 'toolkits', 'windows', 'NLP_test_cases',
'modules', 'Outputs', 'types'],
'punctuation_list': ['.'],
'existential_there_list': [],
'adverb_list': ['only', 'completely','once', 'freely'],

'personal_pronoun_list': [],
'preposition_subordinating_list': ['from', 'with', 'by', 'within', 'of', 'throughout', 'as', 'in'],
'interjection_list': [],
'unbalenced_parenthesis_list': [],
'verb_present_participle_list':['hitting', 'following'],
'verb_past_participle_list': ['called', 'developed', 'held', 'made'],
'coordinating_conjunction_list': ['and'],
'verb_present_tense_person_singular_list': ['has', 'runs', 'is'],
'noun_proper_plural_list': [],
'wh_determiner_list': ['which'],
'determiner_list': ['the', 'All', 'This', 'all', 'no', 'No', 'a', 'The', 'A'],
'predeterminer_list': [],
'cardinal_number_list': ['7', '32bit', 'three', '2015', '2013'],
'adverb_superlative_list': [],
'wh_adverb_list': [],
'noun_propper_list': [],
'not_tagged_word_list': ['(', ')', ',']
}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('This', 'DT'), ('code', 'NN'), ('was', 'VBD'), ('developed', 'VBN'), ('in', 'IN'), ('LabVIEW', 'NNP'), ('2013', 'CD'), ('32bit', 'CD')], [('This', 'DT'), ('code', 'NN'), ('was', 'VBD'), ('developed', 'VBN'), ('in', 'IN'), ('LabVIEW', 'NNP'), ('2013', 'CD'), ('32bit', 'CD')]], [[('No', 'DT'), ('toolkits', 'NNS'), ('needed', 'VBD')], [('No', 'DT'), ('toolkits', 'NNS'), ('needed', 'VBD')]], [[('No', 'DT'), ('modules', 'NNS'), ('needed', 'VBD')], [('No', 'DT'), ('modules', 'NNS'), ('needed', 'VBD')]], [[('Energy', 'NNP'), ('transfer', 'NN'), ('model', 'NN'), ('of', 'IN'), ('a', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('a', 'DT'), ('surface', 'NN'), ('and', 'CC'), ('losing', 'VBG'), ('energy', 'NN'), ('on', 'IN'), ('impact', 'NN'), ('and', 'CC'), ('rebound', 'NN'), ('.', '.')], [('Outputs', 'NNS'), ('to', 'TO'), ('the', 'DT'), ('model', 'NN'), ('are', 'VBP'), (':', ':')]], [[('Simulation', 'NNP'), ('only', 'RB'), ('runs', 'VBZ'), ('once', 'RB'), (',', ','), ('no', 'DT'), ('loops', 'NNS'), ('.', '.')], [('Simulation', 'NNP'), ('only', 'RB'), ('runs', 'VBZ'), ('once', 'RB'), (',', ','), ('no', 'DT'), ('loops', 'NNS'), ('.', '.')]], [[('Assumptions', 'NNS'), ('made', 'VBN'), (':', ':')], [('Assumptions', 'NNS'), ('made', 'VBN'), (':', ':')]], [[('The', 'DT'), ('surfaces', 'NNS'), ('are', 'VBP'), ('completely', 'RB'), ('smooth', 'JJ'), ('with', 'IN'), ('no', 'DT'), ('protrusions', 'NNS')], [('The', 'DT'), ('surfaces', 'NNS'), ('are', 'VBP'), ('completely', 'RB'), ('smooth', 'JJ'), ('with', 'IN'), ('no', 'DT'), ('protrusions', 'NNS')]], [[('The', 'DT'), ('ball', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('particle', 'NN')], [('Any', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('any', 'DT'), ('of', 'IN'), ('the', 'DT'), ('lines', 'NNS'), ('is', 'VBZ'), ('considered', 'VBN'), ('“in”', 'NNS')]], [[('The', 'DT'), ('ball', 'NN'), ('can', 'MD'),

('move', 'VB'), ('freely', 'RB'), ('within', 'IN'), ('the', 'DT'), ('three', 'CD'), ('dimensional', 'JJ'), ('space', 'NN')], [('Any', 'DT'), ('ball', 'NN'), ('hitting', 'VBG'), ('any', 'DT'), ('of', 'IN'), ('the', 'DT'), ('lines', 'NNS'), ('is', 'VBZ'), ('considered', 'VBN'), ('“in”', 'NNS')]], [[('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are', 'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS'), ('.', '.'), ('Input', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('metric', 'JJ'), ('.', '.')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are', 'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS'), ('.', '.'), ('Input', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('metric', 'JJ'), ('.', '.'), ('Inputs', 'NNS'), ('can', 'MD'), ('be', 'VB'), ('in', 'IN'), ('meters', 'NNS'), ('or', 'CC'), ('centimetres', 'NNS'), ('as', 'RB'), ('long', 'RB'), ('as', 'IN'), ('they', 'PRP'), ('are', 'VBP'), ('consistent', 'JJ'), ('throughout', 'IN'), ('.', '.')]], [[('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('outputs', 'NNS'), ('are', 'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS')], [('The', 'DT'), ('data', 'NN'), ('types', 'NNS'), ('of', 'IN'), ('all', 'DT'), ('of', 'IN'), ('the', 'DT'), ('inputs', 'NNS'), ('are', 'VBP'), ('double-precision', 'JJ'), (',', ','), ('floating-point', 'JJ'), ('numbers', 'NNS'), ('.', '.'), ('Input', 'NNP'), ('units', 'NNS'), ('are', 'VBP'), ('metric', 'JJ'), ('.', '.'), ('Inputs', 'NNS'), ('can', 'MD'), ('be', 'VB'), ('in', 'IN'), ('meters', 'NNS'), ('or', 'CC'), ('centimetres', 'NNS'), ('as', 'RB'), ('long', 'RB'), ('as', 'IN'), ('they', 'PRP'), ('are', 'VBP'), ('consistent', 'JJ'), ('throughout', 'IN'), ('.', '.')]], [[('Outputs', 'NNS'), ('to', 'TO'), ('the', 'DT'), ('model', 'NN'), ('are', 'VBP'), (':', ':')], [('Outputs', 'NNS'), ('to', 'TO'), ('the', 'DT'), ('model', 'NN'), ('are', 'VBP'), (':', ':')]], [[('All', 'DT'), ('of', 'IN'), ('the', 'DT'), ('subVIs', 'NN'), ('are', 'VBP'), ('held', 'VBN'), ('within', 'IN'), ('a', 'DT'), ('common', 'JJ'), ('project', 'NN'), ('file', 'NN'), ('NLP_test_cases', 'NNS')], [('All', 'DT'), ('of', 'IN'), ('the', 'DT'), ('subVIs', 'NN'), ('are', 'VBP'), ('held', 'VBN'), ('within', 'IN'), ('a', 'DT'), ('common', 'JJ'), ('project', 'NN'), ('file', 'NN'), ('NLP_test_cases', 'NNS')]], [[('The', 'DT'), ('model', 'NN'), ('has', 'VBZ'), ('3', 'CD'), ('sub', 'NN'), ('VIs', 'NNP'), ('which', 'WDT'), ('are', 'VBP'), ('called', 'VBN'), ('within', 'IN'), ('the', 'DT'), ('main', 'JJ'), ('VI', 'NNP'), ('.', '.')], [('Outputs', 'NNS'), ('to', 'TO'), ('the', 'DT'), ('model', 'NN'), ('are', 'VBP'), (':', ':')]], [[('The', 'DT'), ('following', 'VBG'), ('3', 'CD'), ('sub', 'JJ'), ('VIs', 'NNP'), ('use', 'NN'), ('linear', 'JJ'), ('mathematics', 'NNS'), ('.', '.')], [('The', 'DT'), ('model', 'NN'), ('has', 'VBZ'), ('15', 'CD'), ('sub', 'NN'), ('VIs', 'NNP'), ('which', 'WDT'), ('are', 'VBP'), ('called', 'VBN'), ('within', 'IN'), ('the', 'DT'), ('main', 'JJ'), ('VI', 'NNP'), ('.', '.')]]]

9.7 DOCUMENTS USED FOR CASE STUDY ONE

The text documents that were used for testing as part of case study one are detailed in this section. The documents were read into the application as .txt files.

9.7.1 Requirements For Case Study One Squash Ball Moving Around A Court

1) The simulation is to capture the behaviour of a squash ball in flight.

2) The simulation is to capture the behaviour of a squash ball bouncing off the surface of a wall.

3) The simulation is to capture the behaviour of a ball interacting with a player's racket.

4) The simulation needs to be run multiple times with the only change being the compound of the ball.

5) The flight of the ball is constrained by a regulation sized squash court.

A) The total run time of the simulation should take less than five minutes to fully execute.

B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.

C) No specific computational hardware or peripherals are to be used.

D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.

E) The resolution of the time steps across the models is to be at 0.001 seconds.

F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.

9.7.2 DOCUMENTATION OF A PARTICLE MOVING IN FREE SPACE

This code was developed in LabVIEW 2013 32bit.

No toolkits needed.

No modules needed.

Written in windows 7

Creation date March 2015

Model requires final velocity, starting velocity, distance and sample rate to be known and it calculates acceleration, time that it takes for the ball to get from one end to the other, time sample, displacement steps in the time sample and the distance that ball travels in a single time sample.

First the acceleration of the particle is calculated.

Second the time for the particle to complete the distance is calculated.

Third the time sample step is calculated.

Forth the displacement steps in meters.

Assumptions made:

Simulation uses Newtonian equations of motion.

The ball is considered to be a particle.

All of the inputs are known values.

There is no wind resistance.

The ball is only moving in one plane.

No obstacles in the path of movement.

SI units are used throughout meters, seconds, meters per second, and meters per second per second

Inputs

The following inputs are all floating points

Starting Velocity U (mps)

Distance (S) (m)

Final Velocity (V) (mps)

Sample Rate (NB samples officer Time)

The stop loop input is type Boolean

Stop Loop

Outputs

The following outputs are single floating point numbers.

Loop counts

Time (T) (Seconds)

Acceleration (A) (meters per second per second)

Time Sample (Ts) (seconds)

Displacement step St (meters)

The displacement step St (meters) is an Array

Structure of the model

All of the subVIs are held within a common project file NLP_test_cases

UTA_to_find_S_of_a_particle

UVA_to_find_T_of_a_particle

UVST_to_find_A_of_a_particle

Set_Ts_from_T

9.7.3 DOCUMENTATION OF ENERGY TRANSFER MODEL

This code was developed in LabVIEW 2013 32bit

No toolkits needed

No modules needed

Written in windows 7

Creation date March 2015

Energy transfer model of a ball hitting a surface and losing energy on impact and rebound.

All of the units throughout the simulation use SI units throughout.

Simulation only runs once, no loops.

Assumptions made:

The energy loss from the impact is specified by the user as a percentage

No loss of energy from air resistance

The surfaces are completely smooth with no protrusions

The ball is a particle

The ball can move freely within the three dimensional space

The data types of all of the inputs are double-precision, floating-point numbers. Input units are metric.

A list of all inputs:

Mass (Kg)

Velocity (meters per second mps)

Energy transfer efficiency (%)

The data types of the outputs are double-precision, floating-point numbers

Outputs to the model are:

Remaining energy

Kinetic energy (Jules)

Energy lost (Jules)

Returning velocity (mps)

Structure of the model

All of the subVIs are held within a common project file NLP_test_cases

The model has 3 sub VIs which are called within the main VI.

The following 3 sub VIs use linear mathematics.

Calculating_Velocity_from_Ke.vi

Energy_transfer_losses_model.vi

Kenetic_energy.vi

9.7.4 DOCUMENTATION OF SQUASH COURT IN OR OUT MODEL

This code was developed in LabVIEW 2013 32bit

No toolkits needed

No modules needed

Written in windows 7

Creation date March 2015

Physical body being modelled is based of the specifications laid out in the document World Squash Federation (WSF) Recommended Standards Approved by the WSF January 2013.

All physical dimensions of the squash are taken from WSF document.

The simulation used the (X,Y,Z) Cartesian notation of three dimensional space. (0,0,0) is taken to be the bottom left corner as if a player had just walked through the door.

All calculations are done using numerical representation, double-precision, floating-point number

Simulation only runs once, no loops.

Assumptions made:

Any ball hitting any of the lines is considered "in"

The dimensions of the ball are not considered

The ball is a particle

The Tin does not stand out of the wall

The ball can move freely within the three dimensional space

The surfaces are completely smooth with no protrusions

The players are not considered

The data types of all of the inputs are double-precision, floating-point numbers. Input units are metric. Inputs can be in meters or centimetres as long as they are consistent throughout.

A list of all inputs:

Total Length

Total With

Total Height

Height of Tin

Front Wall Bottom of Service Line

Front wall top of Service Line

Length to short line

With of service box

With of back Quarter

Depth of service box

Height of back wall line

X Coordinate

Y Coordinate

Z Coordinate

The data types of the outputs are both Boolean and double-precision, floating-point numbers

Outputs to the model are:

Boolean outputs:

Front Half

Left Service box

Right Service box

Left Quarter

Right Quarter

Back half

Top wall

Front wall

Bottom wall

Tin

Out Bounds side wall left

In bounds side wall left

Out bounds side wall right

In bounds side wall right

Out bound height

In bound height

Back wall out bound

Back wall in bound

Double-precision, floating-point numbers Outputs:

Height of line at point x

Structure of the model

All of the subVIs are held within a common project file NLP_test_cases

The model has 15 sub VIs which are called within the main VI.

The following 14 use comparative logic.

Above_top_survice_line

Below_Tin_Line

Compair_Above_or_below_Bound_Line

Compair_Above_or_below_Bound_Line_Left_Wall

Compair_Above_or_below_Bound_Line_Right_Wall

Compair_Ball_Above_or_Below_Back_Wall_Line

Compair_Ball_After_Short_Line_Left_Quarter

Compair_Ball_After_Short_Line_Left_Quarter_Survice_box

Compair_Ball_After_Short_Line_Right_Quarter

Compair_Ball_After_Short_Line_Right_Quarter_Survice_box

Compair_Ball_befrore_Short_Line

Compair_Between_Survice_Lines_of_Front_wall

Compair_Between_Survice_Lines_of_Front_wall_back half

Compair_Between_Tin_and_Survice_Line_Front_wall

The VI below used the y=mx+c formula for calculating the gradient of the line a the side of the court.

Gradent_of_the_side_wall_line

9.8 CASE STUDY TWO AUTOMOTIVE CASE STUDY

The first analysis is the comparison between the potential model documentation and the simulation requirements. There are six potential simulation components that were identified therefore there are six sets of data. Unlike the first case study there is not a reference point to compare the output of the application to assess the validity of the output.

9.8.1 ANALYSIS ONE

File inputs:

A. Requirements_for_a_Combined_Braking_and_Steering_System.txt

B. Modeling_an_Anti_Lock_Braking_System_Matlab_Documentation.txt

Outputs From NLP application:

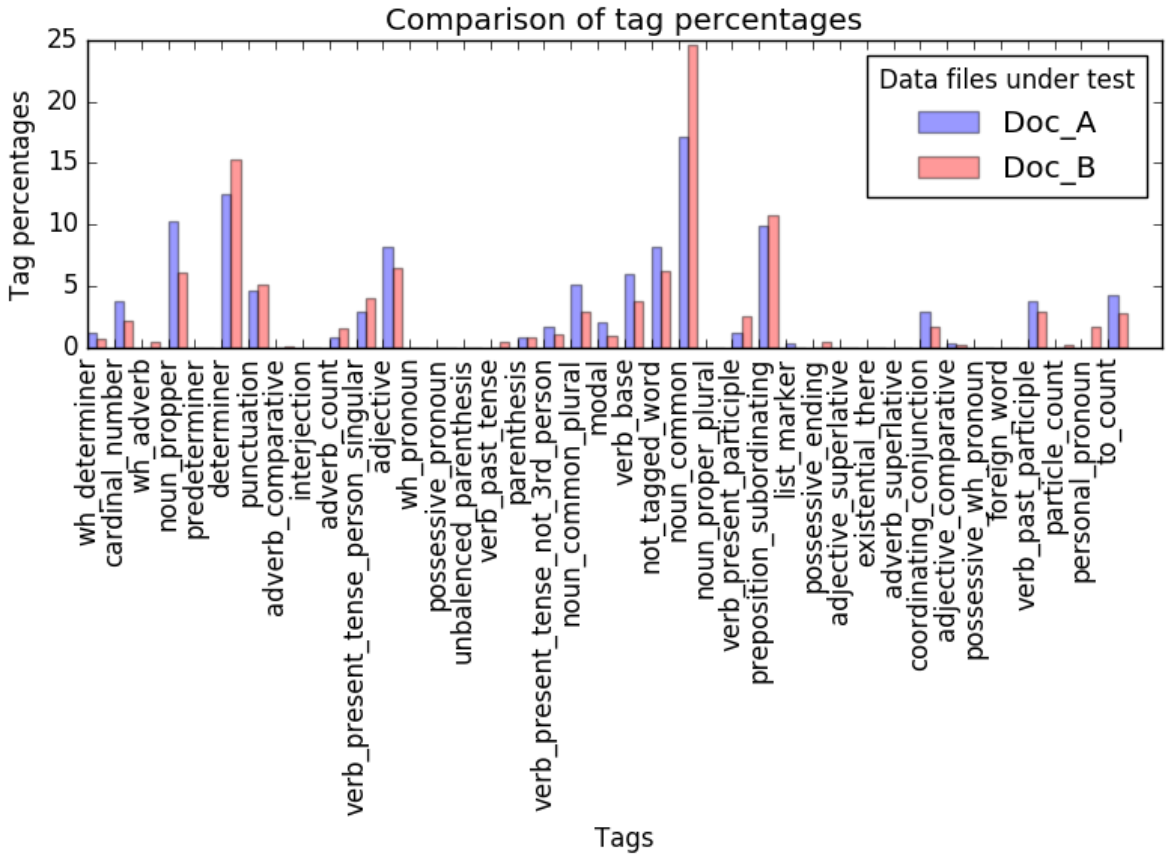

Figure 9.26: The percentage distribution of tags within the two compared documents from case study two analysis one.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 12
common_dev_enviroments = ['Matlab']
 number of words = 1
common_identified_modeling_tuerms = []
 number of common_identified_modeling_tuerms = 0
common_identified_project_tuerms = []
 number of common_identified_project_tuerms = 0
common_identified_prog_languages = ['C', 'C']
 number of common_identified_prog_languages = 2
common_identified_file_types = ['C', 'C']
```

number of common_identified_file_types = 2

Identified_common_company_words = {
'common_identified_prog_languages': ['C', 'C'],
'common_dev_enviroments': ['Matlab'],
'common_identified_file_types': ['C', 'C'],
'common_identified_project_tuerms': [],
'number_of_common_identified_project_tuerms': 0,
'common_identified_modeling_tuerms': [],
'number_of_identified_prog_languages': 2,
'number_of_identified_file_types': 2,
'number_of_common_dev_enviroments': 1,
'number_of_common_identified_modeling_tuerms': 0
}

dict_of_words_in_both_docs = {
'verb_present_participle_list': [],
'adverb_comparative_list': [],
'noun_proper_plural_list': [],
'possessive_wh_pronoun_list': [],
'adverb_list': [],
'interjection_list': [],
'not_tagged_word_list': [',', ')'],
'personal_pronoun_list': [],
'noun_common_plural_list': ['speeds', 'times', 'results'],
'preposition_subordinating_list': ['on', 'from', 'than', 'of', 'If', 'in', 'at', 'for', 'with'],
'possessive_pronoun_list': [],
'adjective_superlative_list': [],
'particle_list': [],
'determiner_list': ['a', 'The', 'A', 'the'],
'existential_there_list': [],
'verb_base_list': ['be'],
'modal_list': ['can', 'may'],
'list_marker_list': [],
'coordinating_conjunction_list': ['and', 'or'],
'verb_present_tense_person_singular_list': ['has', 'is'],
'verb_past_tense_list': [],
'possessive_ending_list': [],
'unbalenced_parenthesis_list': [],
'verb_past_participle_list': ['used'],
'noun_propper_singular_list': ['C', 'Braking', 'System', 'Matlab', 'ABS'],
'foreign_word_list': [],
'punctuation_list': ['.'],
'to_list': ['to'],
'adverb_superlative_list': [],

'parenthesis_list': [';'],
'verb_present_tense_not_3rd_person_list': ['are'],
'noun_propper_list': [],
'cardinal_number_list': ['4', '5', '1', '2'],
'predeterminer_list': [],
'wh_adverb_list': [],
'noun_common_list': ['component', 'simulation', 'hardware', 'model', 'system', 'vehicle', 'speed', 'time'],
'wh_determiner_list': ['that', 'which'],
'wh_pronoun_list': [],
'adjective_comparative_list': ['less'],
'adjective_list': ['specific', 'single', 'hard', 'maximum']
}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('1', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('vehicle', 'NN'), ('which', 'WDT'), ('has', 'VBZ'), ('a', 'DT'), ('combined', 'VBN'), ('ABS', 'NNP'), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('It', 'PRP'), ('is', 'VBZ'), ('used', 'VBN'), ('in', 'IN'), ('this', 'DT'), ('example', 'NN'), ('to', 'TO'), ('illustrate', 'VB'), ('the', 'DT'), ('conceptual', 'JJ'), ('construction', 'NN'), ('of', 'IN'), ('such', 'PDT'), ('a', 'DT'), ('simulation', 'NN'), ('model', 'NN'), ('.', '.')]], [[('2', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('with', 'IN'), ('a', 'DT'), ('sinusoidal', 'JJ'), ('steering', 'NN'), ('input', 'NN'), ('.', '.')], [('It', 'PRP'), ('is', 'VBZ'), ('used', 'VBN'), ('in', 'IN'), ('this', 'DT'), ('example', 'NN'), ('to', 'TO'), ('illustrate', 'VB'), ('the', 'DT'), ('conceptual', 'JJ'), ('construction', 'NN'), ('of', 'IN'), ('such', 'PDT'), ('a', 'DT'), ('simulation', 'NN'), ('model', 'NN'), ('.', '.')]], [[('3', 'LS'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('needs', 'VBZ'), ('to', 'TO'), ('be', 'VB'), ('run', 'VBN'), ('multiple', 'JJ'), ('times', 'NNS'), ('with', 'IN'), ('the', 'DT'), ('speed', 'NN'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('changing', 'VBG'), ('across', 'IN'), ('operational', 'JJ'), ('speeds', 'NNS'), ('from', 'IN'), ('10KH-1', 'JJ'), ('to', 'TO'), ('115KH-1', 'JJ')], [('It', 'PRP'), ('is', 'VBZ'), ('used', 'VBN'), ('in', 'IN'), ('this', 'DT'), ('example', 'NN'), ('to', 'TO'), ('illustrate', 'VB'), ('the', 'DT'), ('conceptual', 'JJ'), ('construction', 'NN'), ('of', 'IN'), ('such', 'PDT'), ('a', 'DT'), ('simulation', 'NN'), ('model', 'NN'), ('.', '.')]], [[('4', 'CD'), (')', ')'), ('The', 'DT'), ('model', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('contain', 'VB'), ('models', 'NNS'), (';', ':'), ('Driver', 'NNP'), ('input', 'NN'), (',', ','), ('ABS', 'NNP'), ('System', 'NNP'), (',', ','), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('This', 'DT'), ('example', 'NN'), ('shows', 'VBZ'), ('how', 'WRB'), ('to', 'TO'), ('model', 'VB'), ('a', 'DT'), ('simple', 'JJ'), ('model', 'NN'), ('for', 'IN'), ('an', 'DT'), ('Anti-Lock', 'NNP'), ('Braking', 'NNP'), ('System', 'NNP'), ('(', '('), ('ABS', 'NNP'), (')', ')'), ('.', '.')]], [[('5', 'CD'), (')', ')'), ('The',

'DT'), ('outputs', 'NNS'), ('off', 'IN'), ('the', 'DT'), ('component', 'NN'), ('systems', 'NNS'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('recorded', 'VBN'), ('.', '.')], [('This', 'DT'), ('component', 'NN'), ('is', 'VBZ'), ('then', 'RB'), ('referenced', 'VBN'), ('using', 'VBG'), ('a', 'DT'), ('"Model", 'NN'), ("'", 'POS'), ('block', 'NN'), ('.', '.')]], [[('A', 'DT'), (')', ')'), ('The', 'DT'), ('total', 'JJ'), ('run', 'NN'), ('time', 'NN'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('should', 'MD'), ('take', 'VB'), ('less', 'JJR'), ('than', 'IN'), ('five', 'CD'), ('minutes', 'NNS'), ('to', 'TO'), ('fully', 'RB'), ('execute', 'VB'), ('.', '.')], [('This', 'DT'), ('significantly', 'RB'), ('reduces', 'VBZ'), ('the', 'DT'), ('time', 'NN'), ('needed', 'VBN'), ('to', 'TO'), ('prove', 'VB'), ('new', 'JJ'), ('ideas', 'NNS'), ('by', 'IN'), ('enabling', 'VBG'), ('actual', 'JJ'), ('testing', 'VBG'), ('early', 'JJ'), ('in', 'IN'), ('the', 'DT'), ('development', 'NN'), ('cycle', 'NN'), ('.', '.')]], [[('B', 'NNP'), (')', ')'), ('The', 'DT'), ('overall', 'JJ'), ('simulation', 'NN'), ('and', 'CC'), ('analysis', 'NN'), ('should', 'MD'), ('be', 'VB'), ('possible', 'JJ'), ('on', 'IN'), ('a', 'DT'), ('mid-range', 'JJ'), ('laptop', 'NN'), ('with', 'IN'), ('the', 'DT'), ('maximum', 'JJ'), ('capability', 'NN'), ('of', 'IN'), ('8GB', 'CD'), ('of', 'IN'), ('Ram', 'NNP'), (',', ','), ('2.5', 'CD'), ('GHz', 'NNP'), ('quad', 'NN'), ('core', 'NN'), ('Intel', 'NNP'), ('Core', 'NNP'), ('i7', 'NN'), ('processor', 'NN'), (',', ','), ('500GB', 'CD'), ('of', 'IN'), ('hard', 'JJ'), ('drive', 'NN'), ('space', 'NN'), ('.', '.')], [('It', 'PRP'), ('is', 'VBZ'), ('used', 'VBN'), ('in', 'IN'), ('this', 'DT'), ('example', 'NN'), ('to', 'TO'), ('illustrate', 'VB'), ('the', 'DT'), ('conceptual', 'JJ'), ('construction', 'NN'), ('of', 'IN'), ('such', 'PDT'), ('a', 'DT'), ('simulation', 'NN'), ('model', 'NN'), ('.', '.')]], [[('C', 'NNP'), (')', ')'), ('No', 'NNP'), ('specific', 'JJ'), ('computational', 'JJ'), ('hardware', 'NN'), ('or', 'CC'), ('peripherals', 'NNS'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('used', 'VBN'), ('.', '.')], [('C', 'NNP'), ('code', 'NN'), ('is', 'VBZ'), ('generated', 'VBN'), ('and', 'CC'), ('compiled', 'VBN'), ('for', 'IN'), ('the', 'DT'), ('controller', 'NN'), ('hardware', 'NN'), ('to', 'TO'), ('test', 'VB'), ('the', 'DT'), ('concept', 'NN'), ('in', 'IN'), ('a', 'DT'), ('vehicle', 'NN'), ('.', '.')]], [[('D', 'NNP'), (')', ')'), ('The', 'DT'), ('Modelling', 'NNP'), ('software', 'NN'), ('which', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('used', 'VBN'), ('includes', 'VBZ'), (';', ':'), ('Matlab', 'NNP'), (',', ','), ('LabVIEW', 'NNP'), (',', ','), ('C', 'NNP'), ('with', 'IN'), ('standard', 'JJ'), ('libraries', 'NNS'), (',', ','), ('or', 'CC'), ('Python', 'NNP'), ('2', 'CD'), ('with', 'IN'), ('standard', 'JJ'), ('libraries', 'NNS'), ('.', '.')], [('Modeling', 'VBG'), ('an', 'DT'), ('Anti-Lock', 'NNP'), ('Braking', 'NNP'), ('System', 'NNP'), ('Matlab', 'NNP'), ('Documentation', 'NNP')]], [[('E', 'NN'), (')', ')'), ('If', 'IN'), ('LabVIEW', 'NNP'), ('or', 'CC'), ('Matlab', 'NNP'), ('is', 'VBZ'), ('used', 'VBN'), ('only', 'RB'), ('a', 'DT'), ('single', 'JJ'), ('license', 'NN'), ('may', 'MD'), ('be', 'VB'), ('used', 'VBN'), ('.', '.')], [('Modeling', 'VBG'), ('an', 'DT'), ('Anti-Lock', 'NNP'), ('Braking', 'NNP'), ('System', 'NNP'), ('Matlab', 'NNP'), ('Documentation', 'NNP')]], [[('F', 'NNP'), (')', ')'), ('The', 'DT'), ('output', 'NN'), ('results', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('saved', 'VBN'), ('in', 'IN'), ('a', 'DT'), ('file', 'NN'), ('format', 'NN'), ('that', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('interrogated', 'VBN'), ('at', 'IN'), ('a', 'DT'), ('later', 'JJ'), ('date', 'NN'), ('.', '.')], [('Figure', 'NN'), ('3', 'CD'), ('visualizes', 'VBZ'), ('the', 'DT'), ('ABS', 'NNP'), ('simulation', 'NN'), ('results', 'NNS'), ('(', '('), ('for', 'IN'), ('default', 'NN'), ('parameters', 'NNS'), (')', ')'), ('.', '.')]], [[('G', 'NNP'), (')', ')'), ('All', 'DT'), ('component', 'NN'), ('parts', 'NNS'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('in', 'IN'), ('the', 'DT'), ('public', 'JJ'), ('domain', 'NN'), ('.', '.')],

[('This', 'DT'), ('component', 'NN'), ('is', 'VBZ'), ('then', 'RB'), ('referenced', 'VBN'), ('using', 'VBG'), ('a', 'DT'), ("'Model", 'NN'), ("'", 'POS'), ('block', 'NN'), ('.', '.')]]]

9.8.2 ANALYSIS TWO

File inputs:

A. Requirements_for_a_Combined_Braking_and_Steering_System.txt

B. Vehicle_Body.txt

Outputs From NLP application:

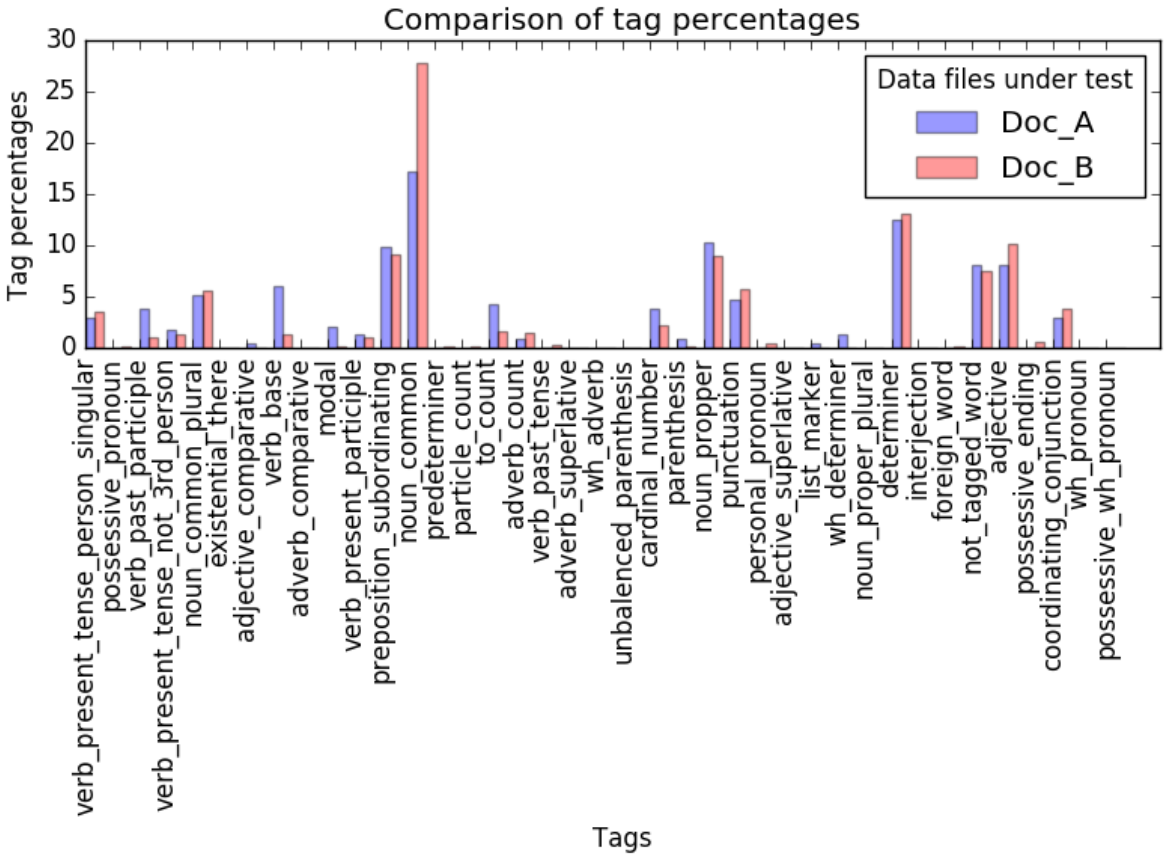

Figure 9.27: The percentage distribution of tags within the two compared documents from case study two analysis two.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 5
common_dev_enviroments = []
number of words = 0
common_identified_modeling_tuerms = ['output', 'output', 'output', 'output']
number of common_identified_modeling_tuerms = 4
common_identified_project_tuerms = []
number of common_identified_project_tuerms = 4
common_identified_prog_languages = []
number of common_identified_prog_languages = 0
common_identified_file_types = []
```

number of common_identified_file_types = 0
Identified_common_company_words = {
'common_identified_project_tuerms': [],
'number_of_common_identified_project_tuerms': 0,
'common_identified_modeling_tuerms': ['output', 'output', 'output', 'output'],
'number_of_common_dev_enviroments': 0,
'number_of_common_identified_modeling_tuerms': 4,
'common_identified_prog_languages': [],
'common_identified_file_types': [],
'number_of_identified_prog_languages': 0,
'number_of_identified_file_types': 0,
'common_dev_enviroments': []
}

dict_of_words_in_both_docs = {
'punctuation_list': ['.'],
'possessive_pronoun_list': [],
'adverb_superlative_list': [],
'adverb_list': ['only'],
'modal_list': ['can'],
'possessive_wh_pronoun_list': [],
'noun_proper_plural_list': [],
'foreign_word_list': [],
'predeterminer_list': [],
'unbalenced_parenthesis_list': [],
'particle_list': [],
'personal_pronoun_list': [],
'wh_determiner_list': [],
'wh_pronoun_list': [],
'determiner_list': ['The', 'A', 'the', 'a'],
'preposition_subordinating_list': ['from', 'in', 'for', 'If', 'with', 'at', 'on', 'of'],
'noun_common_list': ['output', 'model', 'vehicle', 'input', 'speed'],
'noun_common_plural_list': ['models'],
'verb_present_tense_person_singular_list': ['is'],
'adjective_list': [],
'cardinal_number_list': ['2'],
'existential_there_list': [],
'verb_past_participle_list': [],
'not_tagged_word_list': [')', ','],
'noun_propper_singular_list': [],
'verb_past_tense_list': [],
'verb_base_list': ['be'],
'wh_adverb_list': [],
'coordinating_conjunction_list': ['and', 'or'],
'list_marker_list': [],

'adjective_comparative_list': [],
'possessive_ending_list': [],
'interjection_list': [],
'to_list': ['to'],
'noun_propper_list': [],
'verb_present_tense_not_3rd_person_list': ['are'],
'parenthesis_list': [],
'verb_present_participle_list': [],
'adjective_superlative_list': [],
'adverb_comparative_list': []
}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('1', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('vehicle', 'NN'), ('which', 'WDT'), ('has', 'VBZ'), ('a', 'DT'), ('combined', 'VBN'), ('ABS', 'NNP'), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('The', 'DT'), ('vehicle', 'NN'), ('wheels', 'NNS'), ('are', 'VBP'), ('assumed', 'JJ'), ('identical', 'JJ'), ('in', 'IN'), ('size', 'NN'), ('.', '.')]], [[('2', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('with', 'IN'), ('a', 'DT'), ('sinusoidal', 'JJ'), ('steering', 'NN'), ('input', 'NN'), ('.', '.')], [('The', 'DT'), ('vehicle', 'NN'), ('wheels', 'NNS'), ('are', 'VBP'), ('assumed', 'JJ'), ('identical', 'JJ'), ('in', 'IN'), ('size', 'NN'), ('.', '.')]], [[('3', 'LS'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('needs', 'VBZ'), ('to', 'TO'), ('be', 'VB'), ('run', 'VBN'), ('multiple', 'JJ'), ('times', 'NNS'), ('with', 'IN'), ('the', 'DT'), ('speed', 'NN'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('changing', 'VBG'), ('across', 'IN'), ('operational', 'JJ'), ('speeds', 'NNS'), ('from', 'IN'), ('10KH-1', 'JJ'), ('to', 'TO'), ('115KH-1', 'JJ')], [('Physical', 'JJ'), ('signal', 'NN'), ('input', 'NN'), ('ports', 'NNS'), ('W', 'NNP'), ('and', 'CC'), ('beta', 'VB'), ('provide', 'VBP'), ('the', 'DT'), ('means', 'NNS'), ('to', 'TO'), ('specify', 'VB'), ('the', 'DT'), ('headwind', 'NN'), ('speed', 'NN'), ('and', 'CC'), ('road', 'NN'), ('incline', 'NN'), ('angle', 'NN'), ('.', '.')]], [[('4', 'CD'), (')', ')'), ('The', 'DT'), ('model', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('contain', 'VB'), ('models', 'NNS'), (';', ':'), ('Driver', 'NNP'), ('input', 'NN'), (',', ','), ('ABS', 'NNP'), ('System', 'NNP'), (',', ','), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('Physical', 'JJ'), ('signal', 'NN'), ('input', 'NN'), ('ports', 'NNS'), ('W', 'NNP'), ('and', 'CC'), ('beta', 'VB'), ('provide', 'VBP'), ('the', 'DT'), ('means', 'NNS'), ('to', 'TO'), ('specify', 'VB'), ('the', 'DT'), ('headwind', 'NN'), ('speed', 'NN'), ('and', 'CC'), ('road', 'NN'), ('incline', 'NN'), ('angle', 'NN'), ('.', '.')]], [[('F', 'NNP'), (')', ')'), ('The', 'DT'), ('output', 'NN'), ('results', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('saved', 'VBN'), ('in', 'IN'), ('a', 'DT'), ('file', 'NN'), ('format', 'NN'), ('that', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('interrogated', 'VBN'), ('at', 'IN'), ('a', 'DT'), ('later', 'JJ'), ('date', 'NN'), ('.', '.')], [('Physical', 'JJ'), ('signal', 'NN'), ('output', 'NN'), ('ports', 'NNS'), ('V', 'NNP'), (',', ','),

('NF', 'NNP'), (',', ','), ('and', 'CC'), ('NR', 'NNP'), ('provide', 'VBP'), ('the', 'DT'), ('measurements', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('longitudinal', 'JJ'), ('velocity', 'NN'), (',', ','), ('front-axle', 'JJ'), ('normal', 'JJ'), ('force', 'NN'), (',', ','), ('and', 'CC'), ('rear-axle', 'JJ'), ('normal', 'JJ'), ('force', 'NN'), ('.', '.')]]]

9.8.3 ANALYSIS THREE

File inputs:

A. Requirements_for_a_Combined_Braking_and_Steering_System.txt

B. Power_Assisted_Steering_Mechanism.txt

Outputs From NLP application:

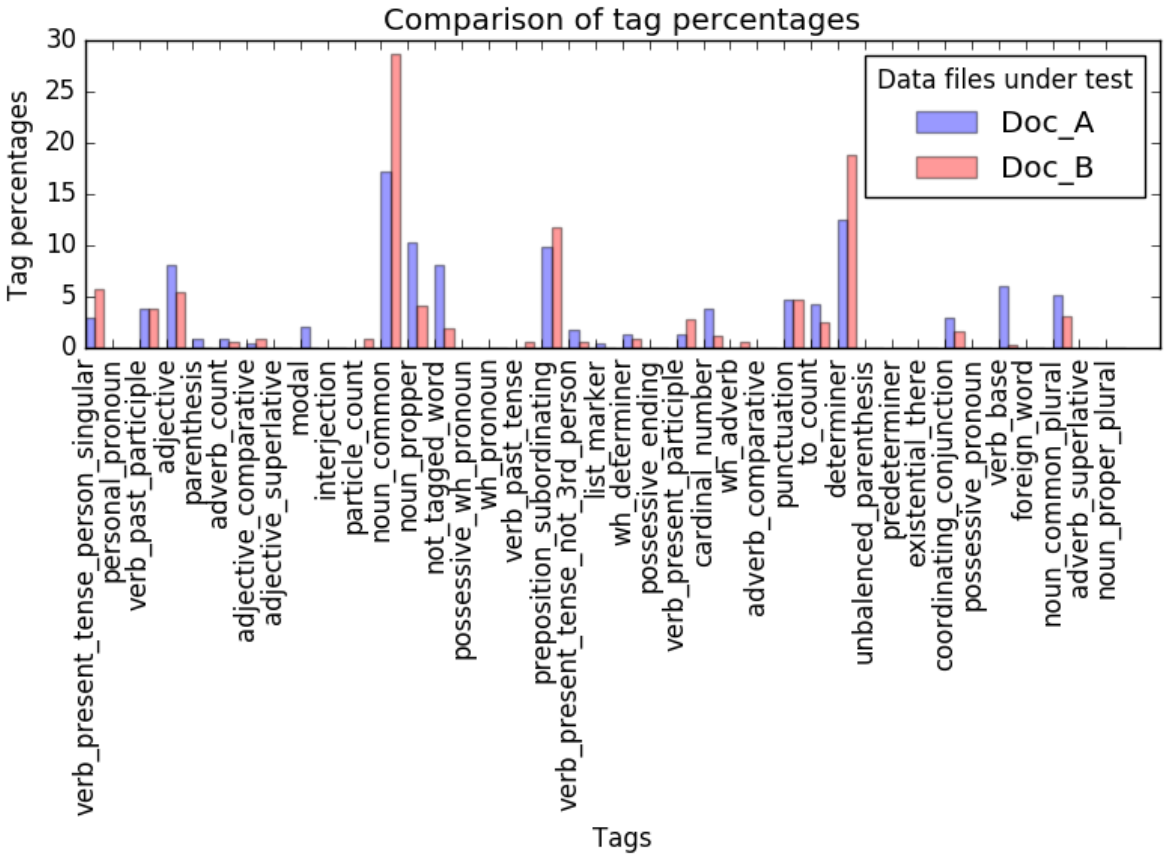

Figure 9.28: The percentage distribution of tags within the two compared documents from case study two analysis three.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 3
common_dev_enviroments = []
number of words = 0
common_identified_modeling_tuerms = []
number of common_identified_modeling_tuerms = 0
common_identified_project_tuerms = []
number of common_identified_project_tuerms = 0
common_identified_prog_languages = []
number of common_identified_prog_languages = 0
common_identified_file_types = []
```

number of common_identified_file_types = 0

Identified_common_company_words = {
'number_of_identified_prog_languages': 0,
'number_of_common_identified_project_tuerms': 0,
'common_dev_enviroments': [],
'number_of_common_dev_enviroments': 0,
'number_of_common_identified_modeling_tuerms': 0,
'common_identified_modeling_tuerms': [],
'common_identified_file_types': [],
'common_identified_prog_languages': [],
'number_of_identified_file_types': 0,
'common_identified_project_tuerms': []
}

dict_of_words_in_both_docs = {
'possessive_ending_list': [],
'coordinating_conjunction_list': ['and', 'or'],
'wh_determiner_list': ['which', 'that'],
'noun_propper_list': [],
'noun_propper_singular_list': ['Steering'],
'cardinal_number_list': ['4'],
'verb_present_participle_list': ['steering'],
'foreign_word_list': [],
'list_marker_list': [],
'noun_proper_plural_list': [],
'adjective_comparative_list': [],
'wh_pronoun_list': [],
'to_list': ['to'],
'interjection_list': [],
'verb_present_tense_person_singular_list': ['is', 'includes'],
'possessive_pronoun_list': [],
'personal_pronoun_list': [],
'adverb_list': [],
'adverb_comparative_list': [],
'verb_present_tense_not_3rd_person_list': [],
'preposition_subordinating_list': ['from', 'If', 'on', 'in', 'for', 'with', 'of'],
'adjective_list': ['hard'],
'punctuation_list': ['.'],
'not_tagged_word_list': [','],
'parenthesis_list': [],
'noun_common_plural_list': [],
'particle_list': [],
'determiner_list': ['the', 'a', 'The'],
'verb_past_participle_list': [],

'adjective_superlative_list': [],
'unbalenced_parenthesis_list': [],
'verb_past_tense_list': [],
'verb_base_list': [],
'adverb_superlative_list': [],
'modal_list': [],
'existential_there_list': [],
'possessive_wh_pronoun_list': [],
'predeterminer_list': [],
'wh_adverb_list': [],
'noun_common_list': ['system', 'steering']
}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('1', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('vehicle', 'NN'), ('which', 'WDT'), ('has', 'VBZ'), ('a', 'DT'), ('combined', 'VBN'), ('ABS', 'NNP'), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('The', 'DT'), ('hydraulic', 'JJ'), ('actuation', 'NN'), ('system', 'NN'), ('includes', 'VBZ'), ('a', 'DT'), ('double-acting', 'JJ'), ('hydraulic', 'JJ'), ('cylinder', 'NN'), (',', ','), ('4-way', 'JJ'), ('valve', 'NN'), (',', ','), ('fixed-displacement', 'JJ'), ('pump', 'NN'), (',', ','), ('and', 'CC'), ('a', 'DT'), ('pressure-relief', 'JJ'), ('valve', 'NN'), ('.', '.')]], [[('2', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('with', 'IN'), ('a', 'DT'), ('sinusoidal', 'JJ'), ('steering', 'NN'), ('input', 'NN'), ('.', '.')], [('This', 'DT'), ('example', 'NN'), ('shows', 'VBZ'), ('a', 'DT'), ('simplified', 'JJ'), ('version', 'NN'), ('of', 'IN'), ('a', 'DT'), ('power-assisted', 'JJ'), ('steering', 'NN'), ('mechanism', 'NN'), ('.', '.')]], [[('4', 'CD'), (')', ')'), ('The', 'DT'), ('model', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('contain', 'VB'), ('models', 'NNS'), (';', ':'), ('Driver', 'NNP'), ('input', 'NN'), (',', ','), ('ABS', 'NNP'), ('System', 'NNP'), (',', ','), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('The', 'DT'), ('hydraulic', 'JJ'), ('actuation', 'NN'), ('system', 'NN'), ('includes', 'VBZ'), ('a', 'DT'), ('double-acting', 'JJ'), ('hydraulic', 'JJ'), ('cylinder', 'NN'), (',', ','), ('4-way', 'JJ'), ('valve', 'NN'), (',', ','), ('fixed-displacement', 'JJ'), ('pump', 'NN'), (',', ','), ('and', 'CC'), ('a', 'DT'), ('pressure-relief', 'JJ'), ('valve', 'NN'), ('.', '.')]]]

9.8.4 ANALYSIS FOUR

File inputs:

A. Requirements_for_a_Combined_Braking_and_Steering_System.txt

B. Simple_2D_kinematic_vehicle_steering_model_and_animation.txt

Outputs From NLP application:

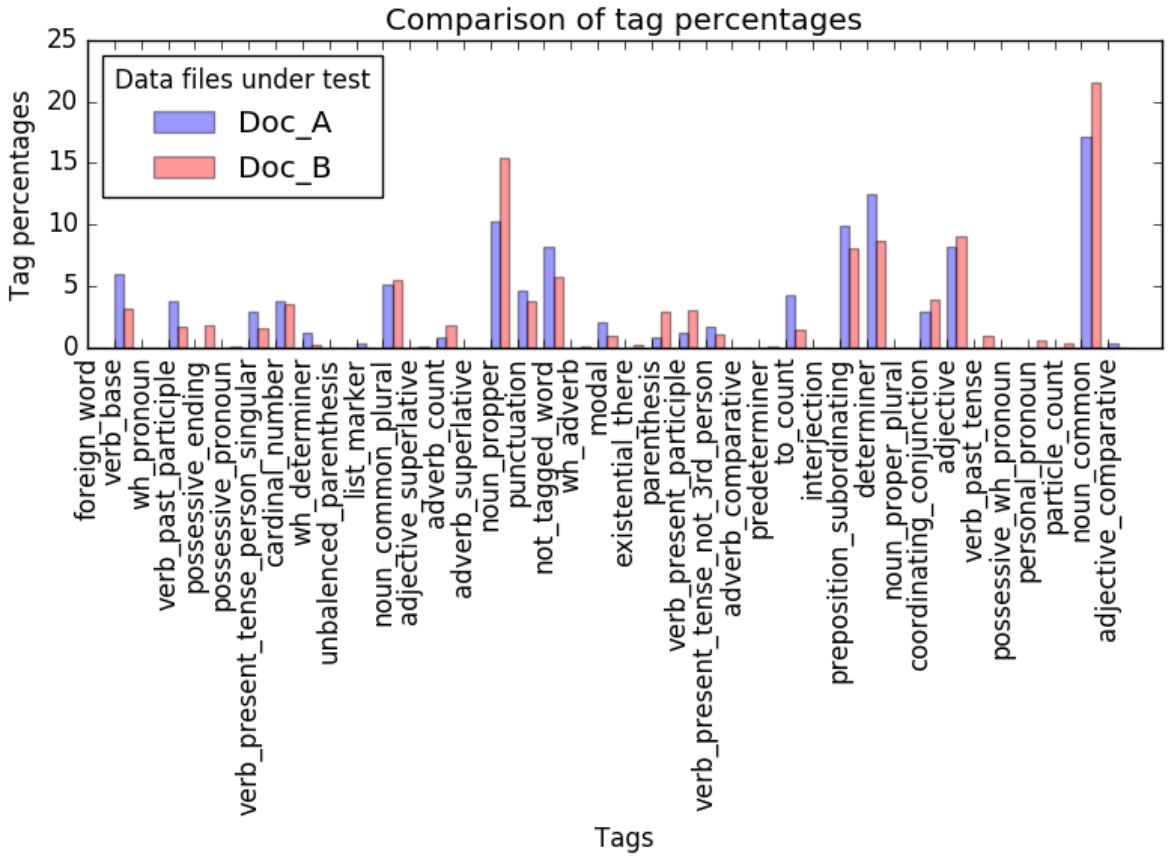

Figure 9.29: The percentage distribution of tags within the two compared documents from case study two analysis four.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 9
common_dev_enviroments = ['Matlab', 'Matlab', 'Matlab', 'Matlab', 'Matlab']
number of words = 5
common_identified_modeling_tuerms = []
number of common_identified_modeling_tuerms = 0
common_identified_project_tuerms = []
number of common_identified_project_tuerms = 0
common_identified_prog_languages = []
number of common_identified_prog_languages = 0
common_identified_file_types = []
```

number of common_identified_file_types = 0

Identified_common_company_words = {
'number_of_identified_prog_languages': 0,
'common_identified_file_types': [],
'number_of_common_identified_modeling_tuerms':0,
'common_identified_modeling_tuerms': [],
'common_identified_project_tuerms':[],
'number_of_common_identified_project_tuerms': 0,
'number_of_identified_file_types': 0,
'common_dev_enviroments': ['Matlab', 'Matlab', 'Matlab', 'Matlab', 'Matlab'],
'number_of_common_dev_enviroments': 5,
'common_identified_prog_languages': []
}

dict_of_words_in_both_docs = {
'verb_present_tense_person_singular_list': ['is'],
'particle_list': [],
'possessive_wh_pronoun_list': [],
'preposition_subordinating_list': ['for', 'in', 'with', 'from', 'of', 'on', 'at'],
'to_list': ['to'],
'adjective_list': [],
'interjection_list': [],
'verb_past_tense_list': [],
'adverb_list': [],
'personal_pronoun_list': [],
'determiner_list': ['The', 'a', 'the'],
'noun_propper_singular_list': ['Matlab', 'Steering'],
'noun_common_list': ['simulation', 'vehicle', 'steering', 'file', 'speed', 'time', 'run',
'model'],
'modal_list': ['can', 'should'],
'existential_there_list': [],
'noun_propper_list': [],
'verb_present_tense_not_3rd_person_list': ['are'],
'adverb_comparative_list': [],
'parenthesis_list': [';'],
'wh_adverb_list': [],
'wh_determiner_list': ['that'],
'noun_proper_plural_list': [],
'predeterminer_list': [],
'coordinating_conjunction_list': ['and', 'or'],
'noun_common_plural_list': ['libraries', 'results'],
'possessive_ending_list': [],
'cardinal_number_list': ['1', '2'],
'foreign_word_list': [],

'not_tagged_word_list': [',', ')'],
'unbalenced_parenthesis_list': [],
'wh_pronoun_list': [],
'verb_present_participle_list': ['steering', 'changing'],
'possessive_pronoun_list': [],
'punctuation_list': ['.'],
'adjective_comparative_list': [],
'adjective_superlative_list': [],
'list_marker_list': [],
'adverb_superlative_list': [],
'verb_base_list': ['execute'],
'verb_past_participle_list': []
}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('1', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('vehicle', 'NN'), ('which', 'WDT'), ('has', 'VBZ'), ('a', 'DT'), ('combined', 'VBN'), ('ABS', 'NNP'), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('The', 'DT'), ('Simulink', 'NNP'), ('model', 'NN'), ('"s", 'POS'), ('base', 'NN'), ('simulation', 'NN'), ('timestep', 'NN'), ('is', 'VBZ'), ('h_fixed=0.05', 'JJ'), ('(', '('), ('s', 'NN'), (')', ')'), (',', ','), ('or', 'CC'), ('50ms', 'CD'), ('.', '.')]], [[('2', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('with', 'IN'), ('a', 'DT'), ('sinusoidal', 'JJ'), ('steering', 'NN'), ('input', 'NN'), ('.', '.')], [('The', 'DT'), ('Simulink', 'NNP'), ('model', 'NN'), ('"s", 'POS'), ('base', 'NN'), ('simulation', 'NN'), ('timestep', 'NN'), ('is', 'VBZ'), ('h_fixed=0.05', 'JJ'), ('(', '('), ('s', 'NN'), (')', ')'), (',', ','), ('or', 'CC'), ('50ms', 'CD'), ('.', '.')]], [[('3', 'LS'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('needs', 'VBZ'), ('to', 'TO'), ('be', 'VB'), ('run', 'VBN'), ('multiple', 'JJ'), ('times', 'NNS'), ('with', 'IN'), ('the', 'DT'), ('speed', 'NN'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('changing', 'VBG'), ('across', 'IN'), ('operational', 'JJ'), ('speeds', 'NNS'), ('from', 'IN'), ('10KH-1', 'JJ'), ('to', 'TO'), ('115KH-1', 'JJ')], [('The', 'DT'), ('Simulink', 'NNP'), ('model', 'NN'), ('"s", 'POS'), ('base', 'NN'), ('simulation', 'NN'), ('timestep', 'NN'), ('is', 'VBZ'), ('h_fixed=0.05', 'JJ'), ('(', '('), ('s', 'NN'), (')', ')'), (',', ','), ('or', 'CC'), ('50ms', 'CD'), ('.', '.')]], [[('4', 'CD'), (')', ')'), ('The', 'DT'), ('model', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('contain', 'VB'), ('models', 'NNS'), (';', ':'), ('Driver', 'NNP'), ('input', 'NN'), (',', ','), ('ABS', 'NNP'), ('System', 'NNP'), (',', ','), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('Simple', 'JJ'), ('2D', 'CD'), ('kinematic', 'JJ'), ('vehicle', 'NN'), ('steering', 'VBG'), ('model', 'NN'), ('and', 'CC'), ('animation', 'NN')]], [[('A', 'DT'), (')', ')'), ('The', 'DT'), ('total', 'JJ'), ('run', 'NN'), ('time', 'NN'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('should', 'MD'), ('take', 'VB'), ('less', 'JJR'), ('than', 'IN'), ('five', 'CD'), ('minutes', 'NNS'), ('to', 'TO'), ('fully', 'RB'), ('execute', 'VB'),

('.', '.')], [('[', 'JJ'), ('setup.m', 'NN'), (']', 'NNP'), ('-', ':'), ('run', 'NN'), ('this', 'DT'), ('first', 'JJ'), (',', ','), ('it', 'PRP'), ('will', 'MD'), ('bring', 'VB'), ('up', 'RP'), ('the', 'DT'), ('Simulink', 'NNP'), (',', ','), ('then', 'RB'), ('press', 'NN'), ('play', 'NN'), ('to', 'TO'), ('simulate', 'VB'), ('the', 'DT'), ('vehicle', 'NN')]], [[('B', 'NNP'), (')', ')'), ('The', 'DT'), ('overall', 'JJ'), ('simulation', 'NN'), ('and', 'CC'), ('analysis', 'NN'), ('should', 'MD'), ('be', 'VB'), ('possible', 'JJ'), ('on', 'IN'), ('a', 'DT'), ('mid-range', 'JJ'), ('laptop', 'NN'), ('with', 'IN'), ('the', 'DT'), ('maximum', 'JJ'), ('capability', 'NN'), ('of', 'IN'), ('8GB', 'CD'), ('of', 'IN'), ('Ram', 'NNP'), (',', ','), ('2.5', 'CD'), ('GHz', 'NNP'), ('quad', 'NN'), ('core', 'NN'), ('Intel', 'NNP'), ('Core', 'NNP'), ('i7', 'NN'), ('processor', 'NN'), (',', ','), ('500GB', 'CD'), ('of', 'IN'), ('hard', 'JJ'), ('drive', 'NN'), ('space', 'NN'), ('.', '.')], [('The', 'DT'), ('Simulink', 'NNP'), ('model', 'NN'), ("'s", 'POS'), ('base', 'NN'), ('simulation', 'NN'), ('timestep', 'NN'), ('is', 'VBZ'), ('h_fixed=0.05', 'JJ'), ('(', '('), ('s', 'NN'), (')', ')'), (',', ','), ('or', 'CC'), ('50ms', 'CD'), ('.', '.')]], [[('D', 'NNP'), (')', ')'), ('The', 'DT'), ('Modelling', 'NNP'), ('software', 'NN'), ('which', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('used', 'VBN'), ('includes', 'VBZ'), (':', ':'), ('Matlab', 'NNP'), (',', ','), ('LabVIEW', 'NNP'), (',', ','), ('C', 'NNP'), ('with', 'IN'), ('standard', 'JJ'), ('libraries', 'NNS'), (',', ','), ('or', 'CC'), ('Python', 'NNP'), ('2', 'CD'), ('with', 'IN'), ('standard', 'JJ'), ('libraries', 'NNS'), ('.', '.')], [('This', 'DT'), ('is', 'VBZ'), ('the', 'DT'), ('Readme', 'NNP'), ('file', 'NN'), ('for', 'IN'), ('a', 'DT'), ('simple', 'JJ'), ('2D', 'CD'), ('kinematic', 'JJ'), ('vehicle', 'NN'), ("'s", 'POS'), ('steering', 'VBG'), ('motion', 'NN'), ('and', 'CC'), ('visualzation', 'NN'), ('implemented', 'VBN'), ('in', 'IN'), ('Matlab', 'NNP'), ("'s", 'POS'), ('Simulink', 'NNP'), ('.', '.')]], [[('E', 'NN'), (')', ')'), ('If', 'IN'), ('LabVIEW', 'NNP'), ('or', 'CC'), ('Matlab', 'NNP'), ('is', 'VBZ'), ('used', 'VBN'), ('only', 'RB'), ('a', 'DT'), ('single', 'JJ'), ('license', 'NN'), ('may', 'MD'), ('be', 'VB'), ('used', 'VBN'), ('.', '.')], [('This', 'DT'), ('is', 'VBZ'), ('the', 'DT'), ('Readme', 'NNP'), ('file', 'NN'), ('for', 'IN'), ('a', 'DT'), ('simple', 'JJ'), ('2D', 'CD'), ('kinematic', 'JJ'), ('vehicle', 'NN'), ("'s", 'POS'), ('steering', 'VBG'), ('motion', 'NN'), ('and', 'CC'), ('visualzation', 'NN'), ('implemented', 'VBN'), ('in', 'IN'), ('Matlab', 'NNP'), ("'s", 'POS'), ('Simulink', 'NNP'), ('.', '.')]], [[('F', 'NNP'), (')', ')'), ('The', 'DT'), ('output', 'NN'), ('results', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('saved', 'VBN'), ('in', 'IN'), ('a', 'DT'), ('file', 'NN'), ('format', 'NN'), ('that', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('interrogated', 'VBN'), ('at', 'IN'), ('a', 'DT'), ('later', 'JJ'), ('date', 'NN'), ('.', '.')], [('The', 'DT'), ('Simulink', 'NNP'), ('model', 'NN'), ("'s", 'POS'), ('base', 'NN'), ('simulation', 'NN'), ('timestep', 'NN'), ('is', 'VBZ'), ('h_fixed=0.05', 'JJ'), ('(', '('), ('s', 'NN'), (')', ')'), (',', ','), ('or', 'CC'), ('50ms', 'CD'), ('.', '.')]]]

9.8.5 ANALYSIS FIVE

File inputs:

A. Requirements_for_a_Combined_Braking_and_Steering_System.txt

B. Tyre_Simple.txt

Outputs From NLP application:

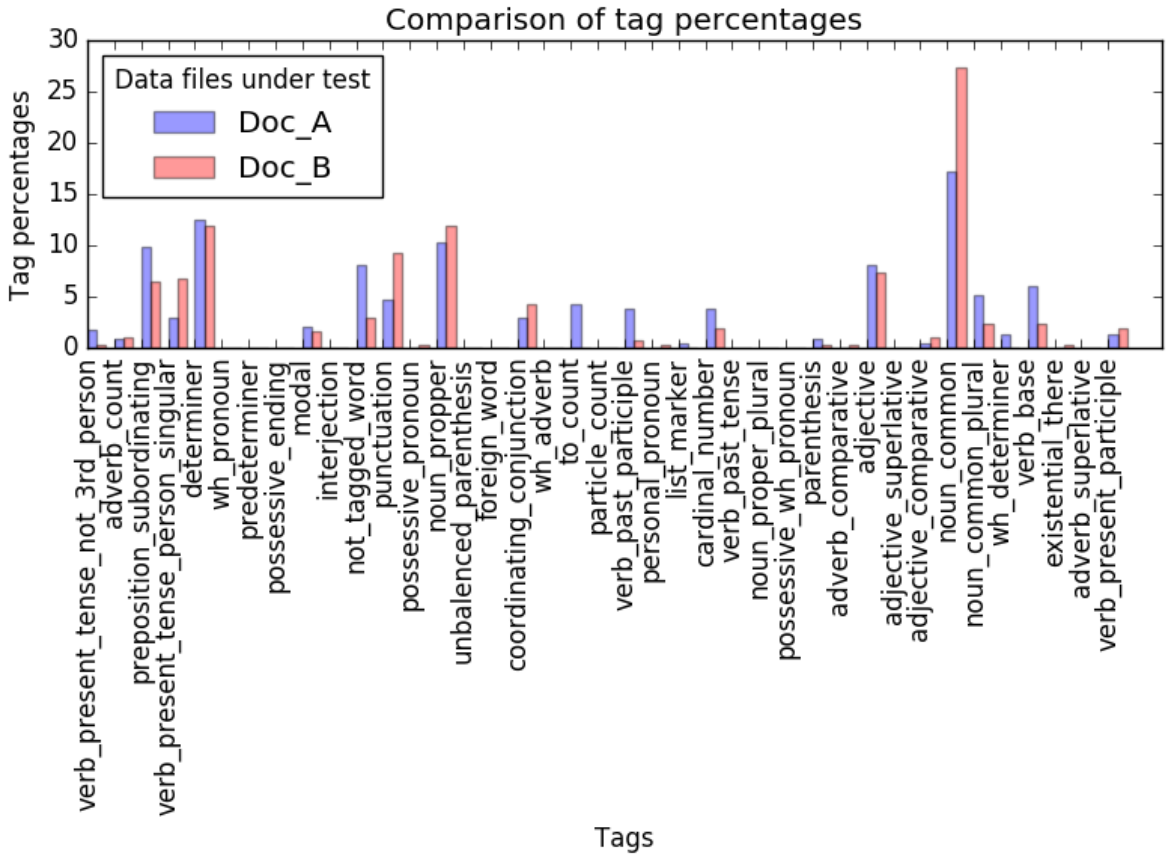

Figure 9.30: The percentage distribution of tags within the two compared documents from case study two analysis five.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 7
common_dev_enviroments = []
number of words = 0
common_identified_modeling_tuerms = []
number of common_identified_modeling_tuerms = 0
common_identified_project_tuerms = []
number of common_identified_project_tuerms = 0
common_identified_prog_languages = []
number of common_identified_prog_languages = 0
common_identified_file_types = []
```

number of common_identified_file_types = 0

Identified_common_company_words ={
'number_of_common_identified_modeling_tuerms': 0,
'common_identified_project_tuerms': [],
'common_identified_modeling_tuerms': [],
'number_of_identified_prog_languages': 0,
'common_identified_file_types':[],
'number_of_common_identified_project_tuerms': 0,
'number_of_identified_file_types': 0,
'common_dev_enviroments': [],
'common_identified_prog_languages': [],
'number_of_common_dev_enviroments': 0
}

dict_of_words_in_both_docs = {
'cardinal_number_list': ['1'],
'adjective_list': [],
'adverb_comparative_list': [],
'modal_list': ['can'],
'personal_pronoun_list': [],
'possessive_ending_list': [],
'not_tagged_word_list': [',', ')'],
'noun_propper_singular_list': [],
'coordinating_conjunction_list': ['and', 'or'],
'possessive_wh_pronoun_list': [],
'predeterminer_list': [],
'adverb_list': [],
'noun_common_plural_list': [],
'existential_there_list': [],
'noun_propper_list': [],
'possessive_pronoun_list': [],
'parenthesis_list': [],
'wh_pronoun_list': [],
'interjection_list': [],
'adjective_superlative_list': [],
'verb_past_participle_list': [],
'wh_adverb_list': [],
'to_list': [],
'adjective_comparative_list': [],
'determiner_list': ['the', 'a', 'The', 'A'],
'foreign_word_list': [],
'particle_list': [],
'verb_present_tense_person_singular_list': ['is', 'includes'],
'adverb_superlative_list': [],

'wh_determiner_list': [],
'noun_common_list': ['capability', 'model', 'simulation'],
'punctuation_list': ['.'],
'verb_base_list': ['be'],
'verb_past_tense_list': [],
'list_marker_list': [],
'noun_proper_plural_list': [],
'unbalenced_parenthesis_list': [],
'verb_present_participle_list': [],
'preposition_subordinating_list': ['of', 'with', 'for', 'than'],
'verb_present_tense_not_3rd_person_list': []
}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('1', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('vehicle', 'NN'), ('which', 'WDT'), ('has', 'VBZ'), ('a', 'DT'), ('combined', 'VBN'), ('ABS', 'NNP'), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('These', 'DT'), ('dynamics', 'NNS'), ('require', 'VBP'), ('additional', 'JJ'), ('computation', 'NN'), ('and', 'CC'), ('might', 'MD'), ('make', 'VB'), ('the', 'DT'), ('model', 'NN'), ('less', 'RBR'), ('suitable', 'JJ'), ('for', 'IN'), ('real-time', 'JJ'), ('simulation', 'NN'), ('.', '.')]], [[('2', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('with', 'IN'), ('a', 'DT'), ('sinusoidal', 'JJ'), ('steering', 'NN'), ('input', 'NN'), ('.', '.')], [('These', 'DT'), ('dynamics', 'NNS'), ('require', 'VBP'), ('additional', 'JJ'), ('computation', 'NN'), ('and', 'CC'), ('might', 'MD'), ('make', 'VB'), ('the', 'DT'), ('model', 'NN'), ('less', 'RBR'), ('suitable', 'JJ'), ('for', 'IN'), ('real-time', 'JJ'), ('simulation', 'NN'), ('.', '.')]], [[('3', 'LS'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('needs', 'VBZ'), ('to', 'TO'), ('be', 'VB'), ('run', 'VBN'), ('multiple', 'JJ'), ('times', 'NNS'), ('with', 'IN'), ('the', 'DT'), ('speed', 'NN'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('changing', 'VBG'), ('across', 'IN'), ('operational', 'JJ'), ('speeds', 'NNS'), ('from', 'IN'), ('10KH-1', 'JJ'), ('to', 'TO'), ('115KH-1', 'JJ')], [('These', 'DT'), ('dynamics', 'NNS'), ('require', 'VBP'), ('additional', 'JJ'), ('computation', 'NN'), ('and', 'CC'), ('might', 'MD'), ('make', 'VB'), ('the', 'DT'), ('model', 'NN'), ('less', 'RBR'), ('suitable', 'JJ'), ('for', 'IN'), ('real-time', 'JJ'), ('simulation', 'NN'), ('.', '.')]], [[('4', 'CD'), (')', ')'), ('The', 'DT'), ('model', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('contain', 'VB'), ('models', 'NNS'), (';', ':'), ('Driver', 'NNP'), ('input', 'NN'), (',', ','), ('ABS', 'NNP'), ('System', 'NNP'), (',', ','), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('The', 'DT'), ('block', 'NN'), ('represents', 'VBZ'), ('a', 'DT'), ('simple', 'JJ'), (',', ','), ('no-slip', 'JJ'), ('model', 'NN'), ('of', 'IN'), ('a', 'DT'), ('tyre', 'NN'), ('parameterized', 'VBN'), ('by', 'IN'), ('its', 'PRP$'), ('radius', 'NN'), ('.', '.')]], [[('A', 'DT'), (')', ')'), ('The', 'DT'), ('total', 'JJ'), ('run', 'NN'), ('time', 'NN'), ('of', 'IN'),

('the', 'DT'), ('simulation', 'NN'), ('should', 'MD'), ('take', 'VB'), ('less', 'JJR'), ('than', 'IN'), ('five', 'CD'), ('minutes', 'NNS'), ('to', 'TO'), ('fully', 'RB'), ('execute', 'VB'), ('.', '.')], [('These', 'DT'), ('dynamics', 'NNS'), ('require', 'VBP'), ('additional', 'JJ'), ('computation', 'NN'), ('and', 'CC'), ('might', 'MD'), ('make', 'VB'), ('the', 'DT'), ('model', 'NN'), ('less', 'RBR'), ('suitable', 'JJ'), ('for', 'IN'), ('real-time', 'JJ'), ('simulation', 'NN'), ('.', '.')]], [[('B', 'NNP'), (')', ')'), ('The', 'DT'), ('overall', 'JJ'), ('simulation', 'NN'), ('and', 'CC'), ('analysis', 'NN'), ('should', 'MD'), ('be', 'VB'), ('possible', 'JJ'), ('on', 'IN'), ('a', 'DT'), ('mid-range', 'JJ'), ('laptop', 'NN'), ('with', 'IN'), ('the', 'DT'), ('maximum', 'JJ'), ('capability', 'NN'), ('of', 'IN'), ('8GB', 'CD'), ('of', 'IN'), ('Ram', 'NNP'), (',', ','), ('2.5', 'CD'), ('GHz', 'NNP'), ('quad', 'NN'), ('core', 'NN'), ('Intel', 'NNP'), ('Core', 'NNP'), ('i7', 'NN'), ('processor', 'NN'), (',', ','), ('500GB', 'CD'), ('of', 'IN'), ('hard', 'JJ'), ('drive', 'NN'), ('space', 'NN'), ('.', '.')], [('These', 'DT'), ('dynamics', 'NNS'), ('require', 'VBP'), ('additional', 'JJ'), ('computation', 'NN'), ('and', 'CC'), ('might', 'MD'), ('make', 'VB'), ('the', 'DT'), ('model', 'NN'), ('less', 'RBR'), ('suitable', 'JJ'), ('for', 'IN'), ('real-time', 'JJ'), ('simulation', 'NN'), ('.', '.')]], [[('F', 'NNP'), (')', ')'), ('The', 'DT'), ('output', 'NN'), ('results', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('saved', 'VBN'), ('in', 'IN'), ('a', 'DT'), ('file', 'NN'), ('format', 'NN'), ('that', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('interrogated', 'VBN'), ('at', 'IN'), ('a', 'DT'), ('later', 'JJ'), ('date', 'NN'), ('.', '.')], [('These', 'DT'), ('dynamics', 'NNS'), ('require', 'VBP'), ('additional', 'JJ'), ('computation', 'NN'), ('and', 'CC'), ('might', 'MD'), ('make', 'VB'), ('the', 'DT'), ('model', 'NN'), ('less', 'RBR'), ('suitable', 'JJ'), ('for', 'IN'), ('real-time', 'JJ'), ('simulation', 'NN'), ('.', '.')]]]

9.8.6 ANALYSIS SIX

File inputs:

A. Requirements_for_a_Combined_Braking_and_Steering_System.txt

B. Tyre_Magic_Formula.txt

Outputs From NLP application:

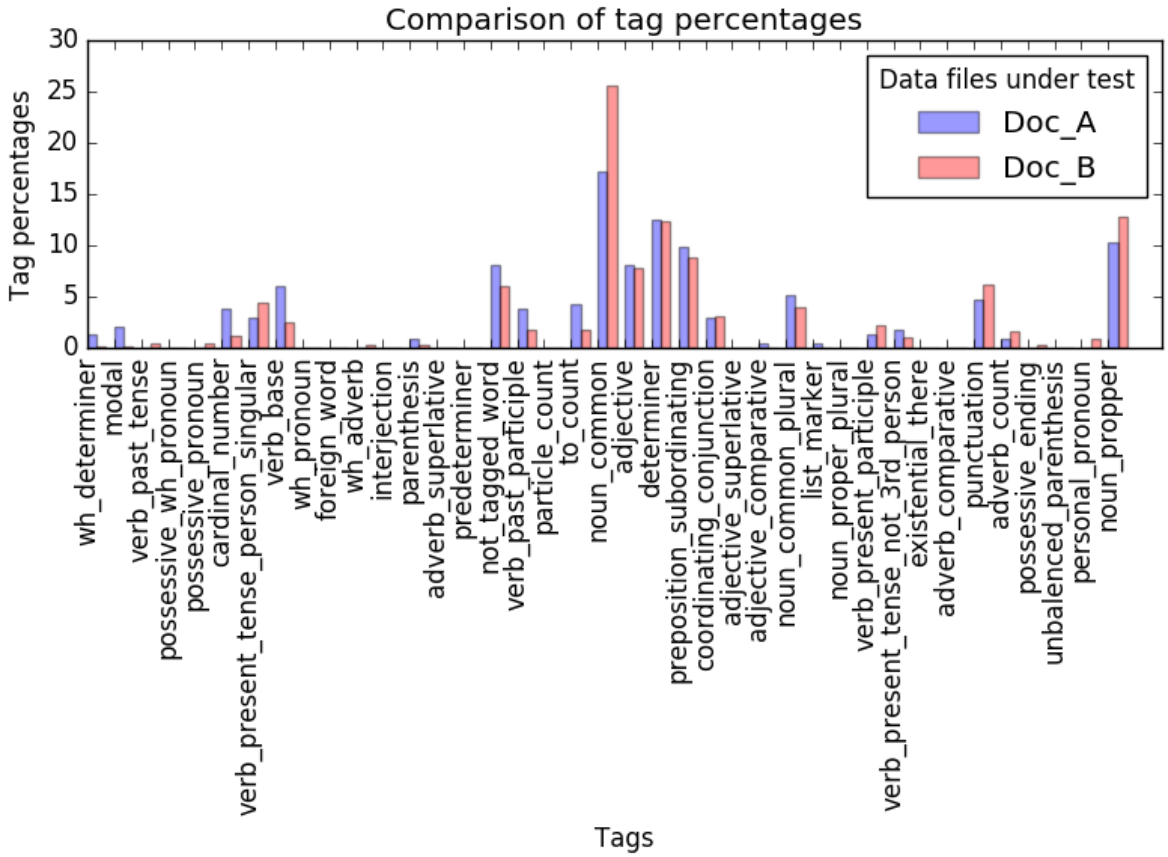

Figure 9.31: The percentage distribution of tags within the two compared documents from case study two analysis six.

Python Console output:

```
Noun-verb Sentences analysis completed and results saved to file
Number of sentences that contain the same Nouns and Verbs = 9
common_dev_enviroments = []
number of words = 0
common_identified_modeling_tuerms = ['outputs', 'output']
number of common_identified_modeling_tuerms = 2
common_identified_project_tuerms = []
number of common_identified_project_tuerms = 2
common_identified_prog_languages = []
number of common_identified_prog_languages = 0
common_identified_file_types = []
```

number of common_identified_file_types = 0

Identified_common_company_words = {
'common_identified_file_types': [],
'common_identified_prog_languages': [],
'number_of_identified_prog_languages': 0,
'common_identified_modeling_tuerms': ['outputs', 'output'],
'common_identified_project_tuerms': [],
'number_of_identified_file_types': 0,
'common_dev_enviroments': [],
'number_of_common_identified_modeling_tuerms': 2,
'number_of_common_identified_project_tuerms': 0,
'number_of_common_dev_enviroments': 0
}

dict_of_words_in_both_docs = {
'noun_proper_plural_list': [],
'wh_pronoun_list':[],
'foreign_word_list': [],
'adverb_list': ['only'],
'coordinating_conjunction_list': ['and', 'or'],
'noun_propper_singular_list': [],
'verb_past_tense_list':[],
'wh_adverb_list': [],
'unbalenced_parenthesis_list': [],
'particle_list': [],
'adjective_superlative_list': [],
'to_list': ['to'],
'verb_base_list': ['be'],
'noun_propper_list': [],
'verb_past_participle_list': [],
'verb_present_participle_list': [],
'noun_common_plural_list': ['speeds', 'models'],
'wh_determiner_list': ['which', 'that'],
'verb_present_tense_person_singular_list': ['includes', 'has', 'is'],
'existential_there_list': [],
'modal_list': ['can'],
'not_tagged_word_list': [')', ','],
'adverb_comparative_list': [],
'punctuation_list': ['.'],
'noun_common_list': ['input', 'vehicle', 'simulation', 'model', 'time', 'component',
'output'],
'cardinal_number_list': ['1'],
'adverb_superlative_list': [],
'possessive_pronoun_list': [],

'adjective_list': [],
'preposition_subordinating_list': ['from', 'of', 'in', 'on', 'If', 'with', 'for', 'at'],
'personal_pronoun_list': [],
'list_marker_list': [],
'parenthesis_list': [';'],
'determiner_list': ['the', 'A', 'The', 'a'],
'possessive_ending_list': [],
'adjective_comparative_list': [],
'predeterminer_list': [],
'verb_present_tense_not_3rd_person_list': ['are'],
'interjection_list': [],
'possessive_wh_pronoun_list': []
}

Program End
Press any key to continue . . .

Data captured in text file:

[[[('1', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('a', 'DT'), ('vehicle', 'NN'), ('which', 'WDT'), ('has', 'VBZ'), ('a', 'DT'), ('combined', 'VBN'), ('ABS', 'NNP'), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('Port', 'NNP'), ('S', 'NNP'), ('outputs', 'VBZ'), ('a', 'DT'), ('physical', 'JJ'), ('signal', 'NN'), ('with', 'IN'), ('the', 'DT'), ('tyre', 'NN'), ('slip', 'NN'), ('measured', 'VBD'), ('during', 'IN'), ('simulation', 'NN'), ('.', '.')]], [[('2', 'CD'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('capture', 'VB'), ('the', 'DT'), ('behaviour', 'NN'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('with', 'IN'), ('a', 'DT'), ('sinusoidal', 'JJ'), ('steering', 'NN'), ('input', 'NN'), ('.', '.')], [('Port', 'NNP'), ('S', 'NNP'), ('outputs', 'VBZ'), ('a', 'DT'), ('physical', 'JJ'), ('signal', 'NN'), ('with', 'IN'), ('the', 'DT'), ('tyre', 'NN'), ('slip', 'NN'), ('measured', 'VBD'), ('during', 'IN'), ('simulation', 'NN'), ('.', '.')]], [[('3', 'LS'), (')', ')'), ('The', 'DT'), ('simulation', 'NN'), ('needs', 'VBZ'), ('to', 'TO'), ('be', 'VB'), ('run', 'VBN'), ('multiple', 'JJ'), ('times', 'NNS'), ('with', 'IN'), ('the', 'DT'), ('speed', 'NN'), ('of', 'IN'), ('the', 'DT'), ('vehicle', 'NN'), ('changing', 'VBG'), ('across', 'IN'), ('operational', 'JJ'), ('speeds', 'NNS'), ('from', 'IN'), ('10KH-1', 'JJ'), ('to', 'TO'), ('115KH-1', 'JJ')], [('Port', 'NNP'), ('S', 'NNP'), ('outputs', 'VBZ'), ('a', 'DT'), ('physical', 'JJ'), ('signal', 'NN'), ('with', 'IN'), ('the', 'DT'), ('tyre', 'NN'), ('slip', 'NN'), ('measured', 'VBD'), ('during', 'IN'), ('simulation', 'NN'), ('.', '.')]], [[('4', 'CD'), (')', ')'), ('The', 'DT'), ('model', 'NN'), ('is', 'VBZ'), ('to', 'TO'), ('contain', 'VB'), ('models', 'NNS'), (';', ':'), ('Driver', 'NNP'), ('input', 'NN'), (',', ','), ('ABS', 'NNP'), ('System', 'NNP'), (',', ','), ('and', 'CC'), ('steering', 'VBG'), ('system', 'NN'), ('.', '.')], [('To', 'TO'), ('increase', 'VB'), ('the', 'DT'), ('fidelity', 'NN'), ('of', 'IN'), ('the', 'DT'), ('tyre', 'NN'), ('model', 'NN'), (',', ','), ('the', 'DT'), ('block', 'NN'), ('enables', 'VBZ'), ('you', 'PRP'), ('to', 'TO'), ('specify', 'VB'), ('properties', 'NNS'), ('such', 'JJ'), ('as', 'IN'), ('tyre', 'NN'), ('compliance', 'NN'), (',', ','), ('inertia', 'NN'), (',', ','), ('and', 'CC'), ('rolling', 'VBG'), ('resistance', 'NN'), ('.', '.')]], [[('5', 'CD'), (')', ')'),

('The', 'DT'), ('outputs', 'NNS'), ('off', 'IN'), ('the', 'DT'), ('component', 'NN'), ('systems', 'NNS'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('recorded', 'VBN'), ('.', '.')], [('This', 'DT'), ('block', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('structural', 'JJ'), ('component', 'NN'), ('based', 'VBN'), ('on', 'IN'), ('the', 'DT'), ('Tyre-Road', 'JJ'), ('Interaction', 'NNP'), ('(', '('), ('Magic', 'NNP'), ('Formula', 'NNP'), (')', ')'), ('block', 'NN'), ('.', '.')]], [[('A', 'DT'), (')', ')'), ('The', 'DT'), ('total', 'JJ'), ('run', 'NN'), ('time', 'NN'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('should', 'MD'), ('take', 'VB'), ('less', 'JJR'), ('than', 'IN'), ('five', 'CD'), ('minutes', 'NNS'), ('to', 'TO'), ('fully', 'RB'), ('execute', 'VB'), ('.', '.')], [('Consider', 'VB'), ('ignoring', 'VBG'), ('tyre', 'NN'), ('compliance', 'NN'), ('and', 'CC'), ('inertia', 'NN'), ('if', 'IN'), ('simulating', 'VBG'), ('the', 'DT'), ('model', 'NN'), ('in', 'IN'), ('real', 'JJ'), ('time', 'NN'), ('or', 'CC'), ('if', 'IN'), ('preparing', 'VBG'), ('the', 'DT'), ('model', 'NN'), ('for', 'IN'), ('Hardware-in-Loop', 'NNP'), ('(', '('), ('HIL', 'NNP'), (')', ')'), ('simulation', 'NN'), ('.', '.')]], [[('B', 'NNP'), (')', ')'), ('The', 'DT'), ('overall', 'JJ'), ('simulation', 'NN'), ('and', 'CC'), ('analysis', 'NN'), ('should', 'MD'), ('be', 'VB'), ('possible', 'JJ'), ('on', 'IN'), ('a', 'DT'), ('mid-range', 'JJ'), ('laptop', 'NN'), ('with', 'IN'), ('the', 'DT'), ('maximum', 'JJ'), ('capability', 'NN'), ('of', 'IN'), ('8GB', 'CD'), ('of', 'IN'), ('Ram', 'NNP'), (',', ','), ('2.5', 'CD'), ('GHz', 'NNP'), ('quad', 'NN'), ('core', 'NN'), ('Intel', 'NNP'), ('Core', 'NNP'), ('i7', 'NN'), ('processor', 'NN'), (',', ','), ('500GB', 'CD'), ('of', 'IN'), ('hard', 'JJ'), ('drive', 'NN'), ('space', 'NN'), ('.', '.')], [('Port', 'NNP'), ('S', 'NNP'), ('outputs', 'VBZ'), ('a', 'DT'), ('physical', 'JJ'), ('signal', 'NN'), ('with', 'IN'), ('the', 'DT'), ('tyre', 'NN'), ('slip', 'NN'), ('measured', 'VBD'), ('during', 'IN'), ('simulation', 'NN'), ('.', '.')]], [[('F', 'NNP'), (')', ')'), ('The', 'DT'), ('output', 'NN'), ('results', 'NNS'), ('of', 'IN'), ('the', 'DT'), ('simulation', 'NN'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('saved', 'VBN'), ('in', 'IN'), ('a', 'DT'), ('file', 'NN'), ('format', 'NN'), ('that', 'WDT'), ('can', 'MD'), ('be', 'VB'), ('interrogated', 'VBN'), ('at', 'IN'), ('a', 'DT'), ('later', 'JJ'), ('date', 'NN'), ('.', '.')], [('S', 'NNP'), ('Physical', 'NNP'), ('signal', 'NN'), ('output', 'NN'), ('port', 'NN'), ('associated', 'VBN'), ('with', 'IN'), ('the', 'DT'), ('relative', 'JJ'), ('slip', 'NN'), ('between', 'IN'), ('the', 'DT'), ('tyre', 'NN'), ('and', 'CC'), ('road', 'NN')]], [[('G', 'NNP'), (')', ')'), ('All', 'DT'), ('component', 'NN'), ('parts', 'NNS'), ('are', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('in', 'IN'), ('the', 'DT'), ('public', 'JJ'), ('domain', 'NN'), ('.', '.')], [('This', 'DT'), ('block', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('structural', 'JJ'), ('component', 'NN'), ('based', 'VBN'), ('on', 'IN'), ('the', 'DT'), ('Tyre-Road', 'JJ'), ('Interaction', 'NNP'), ('(', '('), ('Magic', 'NNP'), ('Formula', 'NNP'), (')', ')'), ('block', 'NN'), ('.', '.')]]]

9.9 DOCUMENTS USED FOR CASE STUDY TWO

The text documents that were used for testing as part of case study two are detailed in this section. The documents were read into the application as .txt files.

Due to copy write issues the text files in their raw forms have not been duplicated in this work. The text files can be found in Matlab 2016a help files by searching the name of the model.

9.9.1 REQUIREMENTS FOR A COMBINED BRAKING AND STEERING SYSTEM

1) The simulation is to capture the behaviour of a vehicle which has a combined ABS and steering system.

2) The simulation is to capture the behaviour of the vehicle with a sinusoidal steering input.

3) The simulation needs to be run multiple times with the speed of the vehicle changing across operational speeds from 10KH-1 to 115KH-1

4) The model is to contain models; Driver input, ABS System, and steering system.

5) The outputs off the component systems are to be recorded.

A) The total run time of the simulation should take less than five minutes to fully execute.

B) The overall simulation and analysis should be possible on a mid-range laptop with the maximum capability of 8GB of Ram, 2.5 GHz quad core Intel Core i7 processor, 500GB of hard drive space.

C) No specific computational hardware or peripherals are to be used.

D) The Modelling software which can be used includes; Matlab, LabVIEW, C with standard libraries, or Python 2 with standard libraries.

E) If LabVIEW or Matlab is used only a single license may be used.

F) The output results of the simulation are to be saved in a file format that can be interrogated at a later date.

G) All component parts are to be in the public domain.