Complexity Characteristics and Measurement within Engineering Systems

by

Craig Read

Submitted in partial fulfilment of the requirements

for the award of

A Doctoral Thesis of Loughborough University

November 2008

Abstract:

Complexity is a significant factor in the development of new products and systems; generally speaking, the higher the complexity, the more difficult products and systems are going to be to design and develop. There are a number of different factors that influence complexity within systems, namely: interoperability; upgradability; adaptability; evolving requirements; system size; automation requirements; performance requirements; support requirements; sustainability; reliability; the need for increased product lifespan; and finally, the length of time systems take to develop.

There is, at present, no common language to describe complexity within engineered systems; this language needs to be developed in order to help industry cope with increasing product complexity and thus meet customer demands. This thesis represents a start in the development of that language, and thus an understanding of systems complexity. The thesis offers a framework for complexity analysis within systems, one which identifies some of the key complexity characteristics that need to be taken into consideration, and which embraces complexity problems, definitions, concepts and classifications, origins and coping mechanisms. It has also has been developed in terms of a measurement approach, thereby allowing for a meaningful comparison between products, and an understanding of the complexities within them. This framework was developed using information collected from academic literature and from more specific case studies. Each complexity characteristic was investigated, and the interactions between characteristics were identified; these interactions allow us to understand complexity and help to develop a common language.

The thesis develops a measurement technique that quantifies various complexity characteristics in terms of the framework laid down, thus enabling a quantified understanding of complexity within systems. This new measurement approach was tested on a set of recent case studies, and the complexity characteristics produced by the measurement technique were, in turn, tested against attributes of the system. The framework itself is always evolving - it incorporates new complexity characteristics. Nevertheless, such evolution can only further our understanding of complexity.

Further work, to explore and integrate the approach demonstrated in this thesis into an automated tool, and test its robustness, along with a continual development of other elements of the framework, such as a classification of complexity, is recommended.

Contents

Table of Figures

Table of Tables

Glossary of Terms

ADT	Axiomatic Design Theory
CAS	Complex Adaptive System
CCCS	Complexity Component and Characteristic Store
CIAS	Complex Interface Adaptive Systems
CIEAS	Complex Interface Element Adaptive Systems
CMT	Complexity Measurement Tool
CONOPS	Concept of Operations
COTS	Commercial Off The Shelf
CPT	Complexity Problem Table
CSIL	Computer Systems Integration Laboratory
DPA	Defence Procurement Agency
DPO	Defence Procurement Office
DSM	Design Structure Matrices
ESIL	Electronics Systems Integration Laboratory
FCS	Future Combat System
FRES	Future Rapid Effect System
FSS	Frequency Selective Surfaces
GEC	General Electric Company
GPS	Global Positioning System
IBDF	Interactions Between Different Factors
ICP	Integrated Capability Programme
LCD	Liquid Crystal Display
LM	Lifecycle Management
MoD	Ministry of Defence
MOTS	Military Off The Shelf
N2	N Squared (System Decomposition Tool)
NSI	Number of Skills Involved
PCO	Procurement Contracts Office

PIA	People Involved within Analysis
TECH	Technology
TRL	Technology Readiness Level
UK	United Kingdom
TECH	Technology
NCS	Naval Command System
NCST	Naval Command System Trainer
SEIC	Systems Engineering Innovation Centre
EMDA	East Midlands Development Agency

1 Introduction

This chapter introduces the research that was conducted, outlines the structure of the Thesis, and provides an overview of the work.

1.1 Thesis Overview

"Our platforms and equipment are increasingly complex." (UK Ministry of Defence)
Complexity is a significant factor in the development of new products and systems. Generally speaking, the higher the complexity of these products, the more difficult they are going to be to design and develop.

The production of more complex systems within the defence sector arises from the need for increased product capability due to ever evolving threats (Wilson 2002). This increased capability requirement is the result of new technologies that are able to support greater functionality, a greater speed of operation, increased automation and an improved accuracy, while also reducing support costs, increasing sustainability and system reliability.

With the need for increased product capability, companies must innovate their products through the development of technology and through savings in manufacture costs. However, with this requirement for innovation in products which incorporate new technologies comes higher costs; in addition there is a need to ensure that systems are servicable for longer periods of time. The engineering lifecycles of these products are lengthening, and programmes are frequently started with projected life spans of 40 years or more, significantly longer than projects started 20 or 30 years ago. With longer running programmes come higher costs; moreover, these products are expected to last longer and to be adaptable to an ever evolving capability requirement that follows threat changes. Products are expected to be upgradeable (as it is too expensive to continue replacing them in their entirety), robust, reliable, sustainable, adaptable and increasingly interoperable (in particular new defence systems as they move towards Networked Enabled Capability or NEC (UK Ministry of Defence 2005).

As product functionality, intricacy and interoperability increases, often the intrinsic complexity of these products also increases. This increase in intrinsic complexity leads to difficulties in accurate modelling and accurate fault finding, often lengthening

development times; moreover intrinsic complexity may, potentially, lead to an increase in emergent properties within systems, something which may or may not be desirable.

Induced complexities occur as a result of the development process, and have a potential to increase the effort, and perhaps the cost, required to develop a system. With the emergence of additional, undesirable complexity, namely, complexity that offers no benefit to the development programme or system, there are, again, further effects on modelling accuracy, and accuracy in fault finding; furthermore, emergent properties, along with development times and costs, all become problematic, and all are potentially avoidable. However, not all induced complexity is detrimental to product development, and in some cases enhancing the complexity of a system reduces the workload and development times; this is achieved, perhaps, with the introduction of already proven commercial or military technology, which may have intricacies within it arising from additional and unnecessary functionality (and therefore unnecessary complexity), but which nevertheless provides a high cost benefit and a reduced development time as a result of its implementation.

Such complexity within systems needs to be understood in an engineering context; complexity needs to be classified and quantified in a way that takes into account specific engineering issues, and in a way that provides useful feedback to programmes. Classifications, characteristics, measures and approaches that minimise intrinsic complexity, while simultaneously managing the desirable and undesirable induced complexities within development programmes need to be developed.

Currently, the words *"complexity"* and *"complex"* are often used within industry as 'buzz-words': thus comments such as *"we develop complex products"*; *"these systems are complex"*; *"we're dealing with a lot of complexity"*, etc. These phrases are often used without any concrete meaning, let alone any common understanding of useful definition or utility. An agreed understanding of complexity in engineering systems is needed, one that can be applied consistently across all engineering domains, so that industry is able to say: *"the system is complex because of x"*, and go on to support the statement.

Before this can be done, an understanding of complexity has to be developed. This thesis produces such an understanding, a *"complexity understanding"*. This

"complexity understanding" can be used and applied within engineering in a clear and consistent manner, providing a tangible meaning to the word *"complex"* within systems development in engineering.

The complexity understanding was developed by means of an exploration of the literature of complexity across a number of disciplines (biology, physics, chemistry, psychology, engineering, etc.). This exploration was further subdivided into complexity definitions, origins, measures and classifications; finally, concepts found within each of these areas were related, if possible, to the engineering domain. For example:

- **Complexity Definitions** – Reducible (can the system be made any smaller? Is it bloated?); Intricate/Coupled/Highly Connected (interfaces that are highly coupled generate interactions that dramatically effect system behaviour – if this is not understood, then undesirable effects may result, which add to the risk of system failure or of poor system performance); Difficult to Analyse (the modelling of the system is too difficult to effect accurately – how, then, can we be sure it will work when built?), etc.

- **Complexity Concepts** - Detail (Static Hierarchical System Structures or organisations are generally easy to model because the flow of information is only up or down the structure); Dynamic (Static Non-Hierarchical System Structures or organisations are much more difficult to understand as elements can exchange information laterally *and* up and down the structure); Adaptive (Changing System Structures are constantly shifting, and so are difficult to model, understand or predict).

- **Complexity Measures** – Connectivity (simple connectivity measures of connections/elements); Intricacy of Connections (understanding and quantifying the complexity of interfaces either quantitatively or qualitatively); Complexity of Elements (same as intricacy, but for elements); Commonality (variation in the design - how much is duplicated?).

These aspects form the complexity understanding, and offer and a very simple, bite-sized view of complexity terms. There are hundreds of definitions, but essentially they can be reduced to basic *themes* - the same is true for concepts and measures.

These *themes* form the basis of the complexity understanding and can be universally used within industry, and attributed to systems relatively easily:

"System X exhibits a non-hierarchical structure but it is stable, it has few connections (low connectivity) between its elements but these connections are data buses transferring vast amounts of information which can dramatically change the behaviour of the elements they are connected to."

In this way, we are thus able to describe complexity in a common and consistent way, using bite-sized chunks; these chunks are referred to as "Characteristics" of the complexity within systems. Moreover, like whole systems with elements and connections between them, the characteristics (elements) can be connected or linked. Complexity definition characteristics can be linked with measurement characteristics, for example; definition of the type "Intricate/Coupled/Highly Connected" can link up conveniently with measures of connectivity and interface complexity; definition type "Difficult to Analyse" links to systems which are difficult to understand (perhaps in terms of the nature of the information flows within them), and with complexity concepts such as "Non-Hierarchical Static System Structures".

These links between the complexity characteristics form the basis of the *"Complexity Framework"* which is a more comprehensive understanding. This is a framework which includes origins, definitions, concepts and approaches to mitigate or control complexity, allowing the user to identify and characterise the complexity within their system, and by using the framework identify all the other linking characteristics; thus, he may, perhaps, be able to look for them in the system, and identify an appropriate mitigation to reduce or control the complexity of the system.

Case studies were used to explore problems arising in programmes and to see how these problems might be caused or influenced by complexity in some way. Six different business areas were chosen within a single company to form the basis of these case studies. The case studies covered air, sea and land base systems, and in total over 20 different problem issues were identified. From these 20 problems, a set of common problem issues were found and the relationships between them and the complexity characteristics within the framework were identified. These relationships were then fed back into the framework.

The thesis later examines how these characteristics, once identified, can be quantified, so not only is the complexity within the system understood - what it is, the nature of it, where it comes from, what we can do about it, etc - but also how *"big"* it is:

"The system exhibits a predominantly hierarchical structure x% with some non-hierarchical structures contained within it y%"

This approach enables a clear, non-subjective, repeatable, consistent and comprehensive method of understanding complexity within engineered systems, and also an ability to understand its size. This is not a part of the conventional industrial process – this approach prevents the use of 'complexity' as a subjective 'buzz-word'. Complexity can now be defined consistently, from system to system, and then quantified. Systems can be compared and analysed; systems can be better understood. Furthermore, as systems are being developed using architectural type tools, this method can be easily incorporated and automated in order to provide feedback to engineers and project managers within programmes.

Such a toolset as this approach provides allows complexity to be managed, monitored and checked. The intrinsic complexity of the system can be found within the basic functional flow diagrams and early system designs;, and as these components are replaced by real engineered designs, the complexities within them, and within the concept as a whole, can be compared to give an approximation of the induced complexity which has been added by design. If detailed designs differ substantially from conceptual designs in terms of their complexity, then this may well be an issue in future – by using the framework such effects can explored in more detail.

Although this approach does not directly influence the costs of programmes, it does provide an additional layer of control other than the standard cost and time controls. The complexity dimension, added to the common standard programme metrics, mean that programmes can be managed more effectively.

1.2 The Thesis Background

1.2.1 A Brief Overview of Systems Engineering

Systems engineering is the basis for the project lifecycles of most modern engineering, and the systems engineering lifecycle (Haskins 2006) is the backbone of product development. The lifecycle consists of a capability or requirement definition

which follows through an iterative process, thus refining set requirements. After this refinement, conceptual designs for the system and sub-systems are created, and detailed designs then follow. The integration of the sub-systems then begins, and as the integration progresses, testing is carried out at sub-system and system level until the integration is completed. Once this testing is complete, then overall system tests finalise product compliance in terms of the customers' needs.

The systems engineering process is used widely within the defence industry, and also within a number of commercial industries. It supports large scale projects such as Eurofighter, Astute and CVF. The use of this process is expanding, and more investment in research into systems engineering tools, and techniques is going ahead. SysML tools (Weilkiens 2008, SysML Partners, Bock 2005) is but one example of investment in new tools to aid systems engineering in the future.

1.2.2 An Overview of the Challenges of Increased Product Complexity for Systems Engineering

However, the systems engineering process is facing some tough challenges, as systems become larger, more intricate, more interoperable, and in need of greater functionality (IBM 2004, Charles, Philip 2003). Coupled with this is the move from static requirement bases to evolving requirements, and to capability acquisition rather than product acquisition. All of this can increase the difficulties associated with the design, development, modelling, integration and testing exercises within the lifecycle.

The systems engineering process is used on a number of projects. It is considered robust and is particularly strong when applied to projects with a fixed requirement, or with fixed capability definitions. The process attempts to finalise the requirements during the requirements capture phases, and then use these requirements to design sub-systems which are later integrated into the whole. Design and development of sub-systems can be planned, managed and conducted concurrently, as the requirements are fixed, and interfaces between sub-systems are, as a result, fixed and documented in specifications. However, with requirements which do not stay fixed throughout product lifecycles, and which need constant updates, the systems engineering process has to deal with new complexities and challenges. Most modern defence projects are no longer based on a fixed requirement; indeed, the majority now

have an initial capability based requirement, from which the detailed requirement specification is derived.

With the move towards capability based requirements (Taylor 2003), taking the format *'we wish to have an ability to achieve set out goals'*, rather than, *'the system shall do x'*, there is an emphasis on the contractor to translate the capabilities into an agreed requirement specification, which is then used for design. The move from a product-delivery perspective, i.e. *'the system shall do x'*, to a capability-delivery perspective, i.e. *'we wish to have an ability to achieve set out goals'* in procurement - currently evident in many systems engineering domains, including defence - adds an additional process to the systems engineering lifecycle, and one which must be addressed. The process represents the translation of capability needs into functional requirements on which the design is based. Invariably, increasing the scope of the process will increase the complexity and resource requirement of that process.

With the introduction of smart procurement (UK Ministry of Defence 1999), and the integration of the Defence Procurement Agency (DPA) and the Defence Logistics Organisations (DLO) to form a new organisation, Defence Equipment and Support (DE&S), an additional requirement on product development becomes evident, one which sets out to integrate support strategies and products with the main product as one package. As a result product capability requirements include not only design, development and testing, but sustainability and logistic support as well. The systems engineering process currently ends at customer acceptance, but product lifecycles will now be much longer as products will include the logistic support element.

Defence companies are already creating businesses that specialise in the support and logistics element of defence systems. Some companies have anticipated this change in procurement strategy and have created customer support and solutions divisions, dedicated to ensuring that customers not only get good products, but that part of that product is the logistic organisation that supports and maintains those systems. The UK Ministry of Defence as part of its Acquisition Operating Framework (UK Ministry of Defence 1999) employs the lines of development which encompass what is expected from newly procured systems:

1. Training
2. Equipment

3. Personnel
4. Information
5. Concepts and Doctrine
6. Organisation
7. Infrastructure
8. Logistics
9. Interoperability

(UK Ministry of Defence 1999)

The systems engineering lifecycle (Haskins 2006) is built upon a fixed blueprint design strategy: requirements are collected, refined, agreed, and then (in an ideal world) fixed. Once the requirements are fixed, engineering practices can be adopted that accelerate the design and production of the product. The systems engineering approach enables a product to be split into sub-systems with their own requirement and interface specifications; these can then be designed simultaneously, using a concurrent engineering approach. Unfortunately, concurrent engineering practices can be troublesome when the requirements for a sub-system are subject to evolution due to technological changes, and demands from customers for these changes to be included. To combat this, interface specifications are then produced which fix the interfaces between system elements in an attempt to deal with evolution in design; however, this is not always effective, and can restrict the level of evolution within the sub-systems, sometimes even preventing customers' demands from being met. This design evolution, due to time scales and technology progression, coupled with the increased interoperability and increased functionality requirements of system elements, has the potential to increase product complexity. The increase in product complexity may affect costs, time scales, reliability, predictability, and, in some cases, even safety; as a result it is important that industry takes steps to understand complexity within their designs in order to mitigate against risk that could potentially cost a programme. This presents yet another challenge for systems engineering.

1.2.3 The Problem Overview

One of the challenges facing systems engineering in the future is the need to better understand complexity within systems. At present industry does not have a mechanism for recognising, quantifying or classifying complexity in design - there are no tools, processes or approaches that characterise or measure complexity effectively. The tools that are available, although not specifically directed towards complexity, attempt to characterise or measure complexity and then use their measures to determine a level of risk. COSYSMO (Valerdi, Boehm et al. 2003) is an example of this, one which uses a composite complexity measure as a factor in budget estimation. Industry does not understand complexity – the origins of complexity, its effects, complexity measurements or metrics, concepts of complexity and approaches to complexity, all remain elusive and unaddressed.

1.3 The Thesis Objectives in Detail

The understanding of complexity in design and development is of prime importance when tackling systems with high levels of interoperability, such as are becoming more and more common. Knowledge of the origins, definitions, measures, coping mechanisms, approaches and problem issues or effects related to complexity in systems are part of this understanding; such knowledge increases awareness of associated risk in products (lack of functionality, not meeting requirements, emergent properties, inability to model effectively, etc.) and their development lifecycles (resource requirements, budgets, personnel, time). The more complex a system is, the more it contains a high number of intricate and coupled components (Evans 1987), the higher the associated risk.

The understanding of complexity in design means developing tools and processes that enable organisations to reduce, understand or better assess and quantify the complexity in products, and the risk, effort and potential costs associated with it. This thesis is a step towards developing that understanding of complexity in engineering systems, by providing a framework for the understanding of complexity characteristics in systems and developing a method of quantifying them.

The Aims of this thesis are:

1. **To produce a framework for further research into complexity within engineering.**
2. **To provide valuable input into the complexity theme within the SEIC.**

The Objectives of this thesis are:

1. **To develop an analysis technique that can be used to create a common understanding, appreciation and concept of complexity within systems engineering in an industrial context.**
2. **To determine complexity characteristics within real life systems using metrics and analysis techniques.**
3. **To provide a system wide view of complexity within the interfaces and sub-systems independently as well as a system whole.**
4. **To validate the tool using real systems and conceptual systems.**

Unfortunately, due to time constraints it has not been possible to test the analysis technique or tool on a real system in the conceptual phase. Defence has been the primary focus of research, and all major engineering programmes in this industry run in excess of the 3 years it has taken to conduct this research. It would be much better to validate the tool output following a real development programme, but in this case it was impractical.

This research work was conducted closely with a defence company as a part of the complexity theme within the SEIC (Systems Engineering Innovation Centre (SEIC)) which is a collaborative effort between BAE Systems , Loughborough University and the East Midlands Development Agency (emda). As a result the work carried out is highly business and engineering orientated, and is focused primarily on defence systems. The approach and analysis technique, however, remains applicable outside of the defence industry, where complex systems are developed.

1.4 Methodology

The problem of complexity spans a number of distinct areas that are all key to improving industrial performance when developing complex systems. In order to

develop a toolset that can cope with and address complexity in design and organisation of all kinds, a number of questions need to be answered.

1. What are the origins of complexity?
2. What are the effects of complexity?
3. Is complexity within design always a bad thing?
4. How can complexity be measured and understood?
5. What are the best approaches to complex problems?
6. How can emergence be quantified or measured?
7. How can intrinsic and induced complexity be reduced?

The answer to these questions will provide an understanding of complexity, and one that can then be used to develop tools, analysis techniques, models and new engineering or organisational approaches. These will then improve the ability of industry to develop complex systems with a reduced risk.

The area of research set out by the questions above is obviously too large for a single research project; in fact some of the questions on their own are too great for a single research project (question 5 for example). As a result, a focus for this research project was required, and this focus was one of measurement, and an understanding of complexity characteristics in systems – that is to say that this research produces an approach to complexity measurement in systems, and an understanding of the complexity characteristics that form that measurement. This was chosen as it seemed the most useful to the projects industrial sponsor.

The following bullet points outline a summary of the method used to develop this measurement approach in the research project. Initially, of course, the focus was not one of measurement, but one of understanding complexity and all its facets from a systems engineering viewpoint; in order to do this the methodology was as follows:

1. The initial step when conducting the research was to gather information surrounding the subject in order to form a comprehensive literature review of the subject area (chapter 2). This was achieved by dividing the information collected into categories, namely: complexity origins; effects; measures; and approaches and classifications from a number of different disciplines (biology,

engineering, mathematics, etc.). This method ensured that the breadth of complexity concepts across disciplines was covered, and specific themes for each category could be explored; these could later be related to engineering issues. The literature did not address specifically developmental issues within programmes, but rather the origins of complexity within development programmes; consequently, the literature review needed to be bolstered by case studies from within industry that identified typical industrial problems. Once collected, each category identified themes within the literature. These themes (referred to as 'types' within the framework) were focused on a commonality between various characteristics identified within each category under scrutiny. The definition of each category of complexity contained the following themes: reducibility; difficulty in modelling difficulty in understanding, etc.

2. Information was collected regarding problems within industrial programmes. Case studies were used, and an interview technique was utilised, in order to gather data, and any accounts of the difficulties that exist or have existed within engineering programmes, along with any solutions that were implemented. The issues that arose from the case studies were then later explored in more detail, linking them to phases of the development lifecycle and attributes of the programmes, such as timescales, budget, and resources. Although each interviewee was given a directive as to the focus of the research, there was a distinct risk that some problem issues identified and documented would not be a result of complexity within the system or design process.

3. The problem issues identified in each case study were linked to complexity themes or characteristics found within each category, as explored in the literature review. These links enabled an identification of the common complexity characteristics as they exist between different problem issues; only the most common characteristics of complexity were linked, along with problems that have similar complexity characteristic mappings. The difficulty in linking problem issues with complexity characteristics is that, in some cases, there may not be a clear relationship between the issue or issues and the complexity characteristics, as no clear distinction has been made between

issues resulting from complexity and those which have not. In addition, some complexity characteristics are more easily mapped than others.

4. A framework was developed, based upon the links between complexity characteristics; this included definitions, origins, problem issues, measures, concepts and classifications. The development of this framework was achieved by means of an analysis of any linkages between complexity themes within the categories, along with the previous analysis of linkages between problem issues within industry. These linkages form the basis of the complexity understanding, and thus form the Complexity Framework.

This was the point at which the research moved towards the production of a concept of complexity measurement, a quantifying strategy for complexity in systems. This production was built upon the understanding gathered from the work described above. The next stage in the methodology was to create the analysis approach for complexity in systems, and to test this approach. This was achieved in the following way:

5. A measurement tool was built, together with an approach to complexity within engineering. This was achieved by utilising a typical product development lifecycle consisting of requirements capture, design, development, testing and validation.

    a. Discussions were conducted which aimed to gather requirements from the industrial sponsor. These discussions helped to highlight issues or concerns raised by the sponsor concerning the development or future use of the tool or approach in development.

    b. The development phase was iterative, with the initial tool developed and tested using real system data. The results generated by the tool were then analysed, and improvements made to the tool; the data was revisited if necessary.

    c. The testing and validation of the approach, and of the tool developed, was difficult under the circumstances encountered. In order to validate the tool outputs there needs to be a datum to validate against; currently, the tool has been tested against already existing systems.

An overall weakness in the methodology is that of not testing the tool within a complete lifecycle of a product. Unfortunately the sponsors' typical programmes, within which they plan to utilise such a tool, frequently run to in excess of 5 years.

Figure 1 shows the research plan, beginning in October 2004 and finishing in October 2007 with a write up period extending to April 2008. The activities shown within the gant chart are linked to the methodology described above.

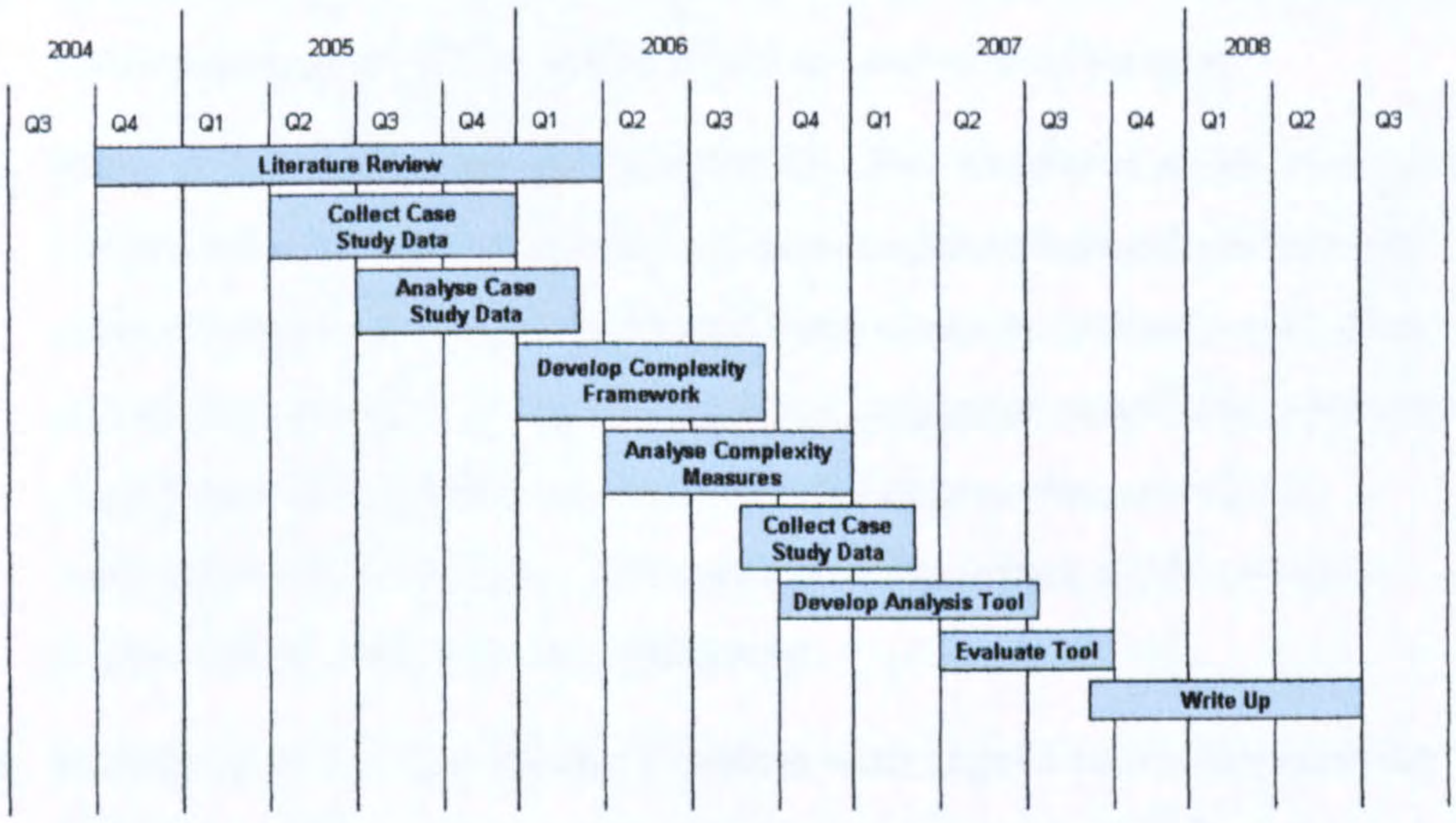


Figure 1 - Research plan.

Table 1 below shows the links between the methodology approach and the activities outlined within the schedule in Figure 1.

	1	2	3	4	5		
					a	b	c
Literature Review	■						
Collect Case Study Data		■					
Analyse Case Study Data		■	■				
Develop Complexity Framework			■	■			
Analyse Complexity Measures					■		
Collect Case Study Data						■	
Develop Analysis Tool					■	■	■
Evaluate Tool							■

Table 1 - Mapping of activities to methodology.

1.5 Introduction of the Roadmap

This section is an overview of the thesis structure in detail – it provides a roadmap that will act as a guide throughout, and will be offered again as key milestones in the thesis are reached.

The thesis is structured as follows:

1. **Introduction –** *This chapter introduces the research that was done, outlines the structure of the thesis and provides an overview of the work.*
2. **State of the Art Review on Complexity -** *This chapter is a literature review which serves to generate a comprehensive and detailed understanding of various complexity components or characteristics in systems engineering: definitions; examples of system; causes or origins of complexity; concepts or classifications of complexity; measures and approaches, or coping mechanisms of complexity. This understanding is then used to create a framework of complexity in engineering.*
3. **Statement of the Complexity Problem with regard to the Engineering Lifecycle -** *This chapter uses the literature review (chapter 2) to determine what the potential complexity issues are within engineering systems. This section evaluates the changes in the defence sector and procurement strategies and outlines the problem to be tackled by the research. This is then elaborated by comparison with 'real world' problems investigated within chapter 3.*
4. **Initial Complexity Case** Studies **-** *The following section analyses a set of case studies which identify typical industrial problems and also how they may be complexity issues, or related to complexity issues. It expands on the theoretical nature of chapter 2, and the problems identified within chapter 3.*
5. **Introduction to Complexity Characteristics and Mapping of Characteristics to Complexity Problem Issues -** *This chapter identifies some of the key and common aspects of complexity problems in engineering systems using information gathered from the literature within chapter 2 and also from the initial case studies from chapter 3. The case studies within chapter 4 are then mapped onto these complexity characteristics, which leads onto the Complexity Framework which is described in chapter 6.*

6. **The Characteristics of Complexity and their Inter-relations: A Complexity Framework** - *This chapter examines the relationships between various complexity characteristics. These characteristics were determined from the literature search in chapter 2, the case studies and the interactions within the case studies of complexity components in chapter 4. The output is a framework of complexity which can be used as an aid to creating various measurements, coping mechanisms or management approaches for complexity in engineering'*

7. **Measurement of Complexity in Engineering** - *This chapter narrows the scope of the thesis to focus on measuring complexity characteristics. It contains a summary of the complexity measures that were found in the literature search within section 2.7. These are then down selected against what is considered useful for engineering: the background to complexity problems in engineering from chapter 3, the problems and issues within engineering that may need addressing in chapter 4 and the understanding of their relationships to different complexity characteristics in chapter 5; and assessments of how easy to apply the measures are (available data etc.). This information then filters into creating the complexity measurement tool in chapter 8.*

8. **Industrial Case Study Data and the Complexity Measurement Tool** - *This chapter takes the information regarding complexity issues and their mappings to complexity characteristics in chapters 4 and 5, and then using the measures that are useful from chapter 7, creates an analysis tool for complexity in systems (including the sub-systems and interfaces).*

9. **Results Analysis** - *This chapter details the results that were generated from the second set of case studies outlined within section 8.2 using the complexity analysis tool outlined within section 8.3. Once the results are collected and shown they are then compared and conclusions made regarding the tool and data.*

10. **Changes to the Tool** -*This chapter details the changes that were made to the complexity measurement tool as a result of the first set of results and the analysis within chapter 9. The conclusions formed from those results lead to*

the introduction of some other metrics, the removal of some metrics. The new results from the new metrics have been included here, and these metrics are; maturity, increased statistics and spreads for interface types and commonality.

11. **Discussion -** *The following chapter reviews all the work so far, the initial development of potential complexity problems from case studies (chapter 4), the mapping of those case studies onto complexity characteristics and the creation of the Complexity Framework (chapter 5), the understanding of the relationships between different complexity characteristics (chapter 6), the understanding of measurement of complexity and how it relates to complexity characteristics (chapter 7), the development of a complexity measurement tool and strategy (chapter 8), the analysis of the results from that measurement strategy (chapter 9) and the improvements made to the tool as a result of evaluation (chapter 10). Finally a reflection back on the work, what could be improved, what work is required and an assessment of how well the work met the set out objectives.*

12. **Conclusion -** *The following chapter details the conclusions of this thesis. The conclusions refer back to the discussion within chapter 11 against the aims set out within section 1.3.*

A flowchart has been developed as a guide through the thesis, indicating sections in the work, and the relationships between these sections. This roadmap and summary is first presented here, and from this point will appear at the beginning of every chapter with a short introduction and summary.

1.6 Outputs of the Thesis

The following is a diagrammatical view of the layout of the thesis including the different outputs, and the flow of work.

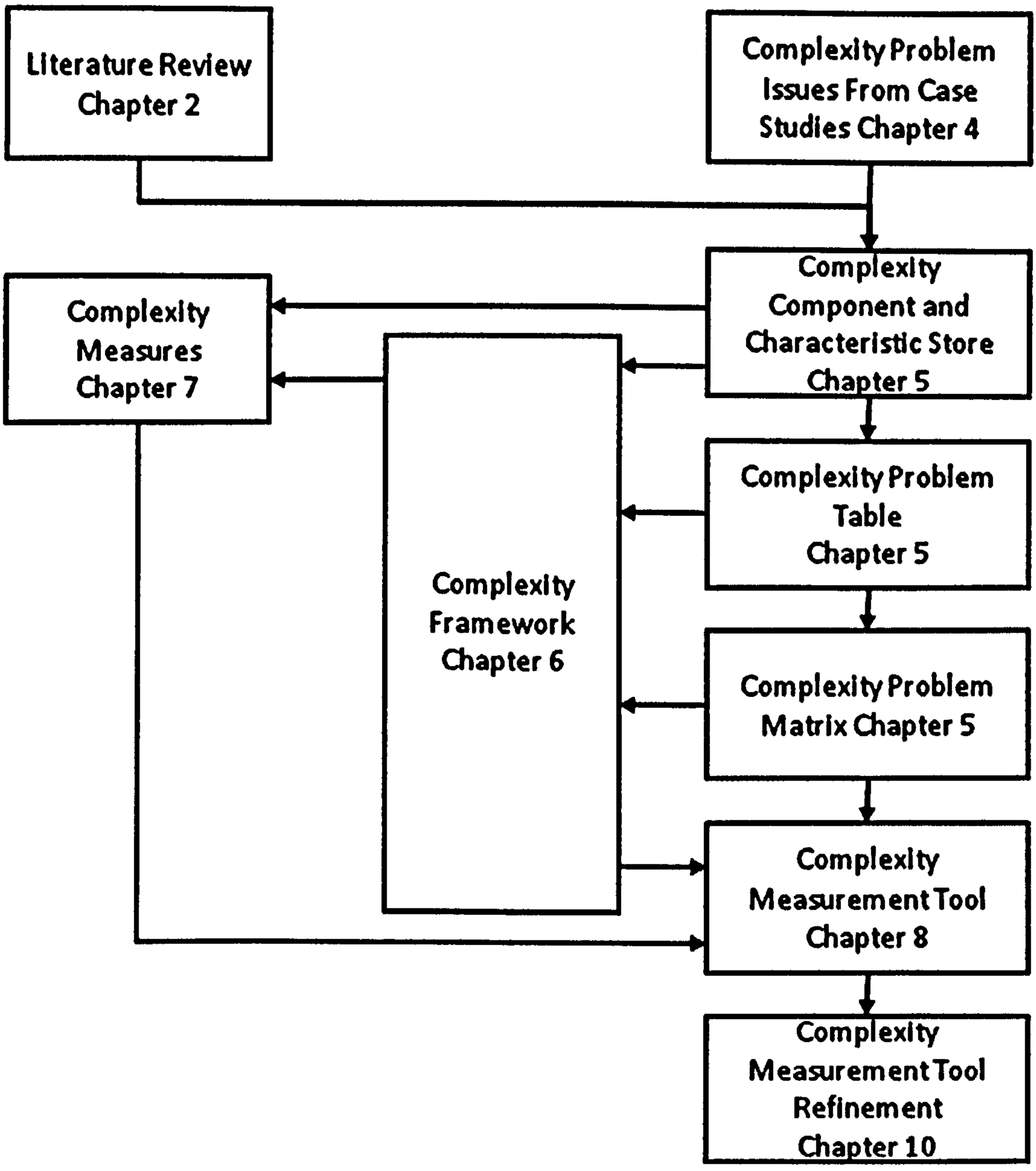


Figure 2 - The layout of the work and the thesis outputs roadmap.

The following are the elements within the flow diagram shown within Figure 2:

Complexity Component and Characteristic Store (CCCS) – A product of the literature review, and problem analysis, the store contains different complexity attributes and problem issues.

Complexity Problem Table (CPT) – A table which describes how the various aspects of the problems inter-relate. It also contains details regarding any coping mechanisms or approaches that were implemented. These are recorded and related, in order to improve the understanding of the problem issues within engineering.

Complexity Framework – The Complexity Framework or Complexity Framework shows the various attributes and characteristics of complexity, and how they interact and affect each other.

Complexity Problem Matrix (CPM) –The mapping of complexity characteristics from within the CCCS to the complexity problems identified within the case studies.

Complexity Measurements – A detailed look at complexity measurement, using data gathered from the literature review and the CCCS.

Complexity Measurement Tool (CMT) – The complexity measurement tool, and approach is the primary output of the thesis.

2 State of the Art Review on Complexity

This chapter is a literature review which serves to generate a comprehensive and detailed understanding of various complexity components or characteristics in systems engineering; definitions, examples of systems, causes or origins, concepts or classifications, measures and approaches or coping mechanisms of complexity. This understanding is then used to create a framework of complexity in engineering.

2.1 Introduction

This section of the thesis is an outline of the literature review undertaken to collect information concerning complexity from various disciplines. The review is centred on information regarding the following areas of complexity, and in the following order, with an introduction to complexity in engineering:

- **Definitions of Complexity** and their appropriate domains, providing different views for how complexity is described, what constitutes complex systems, and what attributes are common to complex systems. This also enables common themes within definitions to be established.
- **Complex System Examples,** from both engineering and from nature, providing insight into what constitutes a complex system, and how these systems compare with definitions.
- **The Causes of Complexity,** the origins and effects of complexity in engineering and other disciplines, helping to understand how complexity affects engineering design processes, products or systems.
- **Complexity Concepts and Classifications,** the conceptual ideas surrounding complexity in various domains, and the types of complexities that can exist also form a valuable part of the complexity understanding that the literature review is trying to develop.
- **Complexity Measurement,** the various different measures and their approaches are of great importance to the industrial sponsor, as measurement of complexity is one method of showing improvement in engineering processes at reducing complexity in systems that is costly.

- **Complexity Coping Mechanisms and Approaches**, help to control the unnecessary complexity in industrial products or systems. Understanding what the available mechanisms or approaches are is of key interest to the industrial sponsor and is very useful when considering the complexity understanding.

As complexity is such a widely researched topic, the categories were chosen for an exploration of complexity in a variety of different disciplines, not merely engineering; some mathematical and biological sources are also included. The inclusion of a number of different disciplines allowed cross fertilisation of ideas and concepts; however, engineering is the focus of the work.

Problem areas were excluded from the literature review as they were case study specific, and specific to the industrial sponsor. Problem issues were identified from these case studies (see chapter 4) and consequently do not form a part of the literature review.

2.1.1 Complexity Research Programmes in the UK, Europe and Worldwide

There are a number of complexity research programmes within the UK, Europe and worldwide. A large investor in complexity research in the UK and Europe is the Engineering and Physical Sciences Research Council (EPSRC). The EPSRC has a number of research and teaching strands within the complexity domain, including:

- £12 million of research funded in novel computation: coping with complexity
- £0.6 million for taught courses in complexity and complex systems
- £10 million investment in a 5-year centre in Large Scale Complex IT Systems
- £8 million over five years for two Doctoral Training Centres in complexity science
- £2.5m for Fundamentals of Complexity Science 2007

The EPSRC also co-ordinates the Complexity-NET organisation, which aims to link complexity research and training activities carried out at different institutions throughout Europe using the ERANET scheme for Complexity Science.

The Santa Fe Institute, founded in 1984, is a private, non-profit based independent research and education centre, specialising in multidisciplinary collaborations in the physical, biological, computational, and social sciences. The institute specialises in the complexity domain and considers the understanding of adaptive complexity to be critical when addressing key environmental, social, technological, economic and political challenges. The research strands or "topics" are founded on these concepts and are detailed within Table 2 below.

Santa Fe Institute Topics	Programmes within Topic
Physics of Complex Systems	Non-equilibrium Statistical Physics and Self-Organization Quantum Information and Decoherence Networks: Social, Biological, and Technological Scaling, Universality, and Quantitative Laws of Life
Emergence, Innovation & Robustness in Evolutionary Systems	Innovation in Biological Systems Innovation in Technological Systems Innovation in Markets Robustness in Biological and Social Systems
Information Processing & Computation in Complex Systems	Quantum Algorithms and Cryptography Phase transitions in NP-Hard Constraint Satisfaction Problems Pattern Discovery in Time Series and Spatiotemporal Data Evolving and Understanding Computation in Cellular Arrays Statistical Order and Robust Information Processing Biologically Inspired Solutions to Computational Problems
Dynamics & Quantitative Studies of Human Behaviour	Financial Markets as an Empirical Laboratory The Cost of Money as a Public Good Emergence and Robustness of Community Structures Geographic Structure, Demographic History, and Approximate Bayesian Computation War-time Sexual Violence Agent-based Modelling in the Social Sciences The Dynamics of Civilizations The Evolution of Human Behaviours & Institutions The Evolution of Human Languages
Emergence, Organization & Dynamics of Living Systems	Origin, Synthesis & Form of Life The Emerging Ecology of Living Systems The Emergence of Social Ecosystems HIV Propagation and Treatment

Table 2 - Santa Fe Institute topics, programmes and research activities.

The Santa Fe Institute attracts scientists and researchers from universities and governments globally, and also offers educational courses that reside with the topics above. The institute is involved with other projects and businesses in its research activities.

The Institute for Complexity Sciences (ICC) supports research within the complexity domain, contributing to workshops, conferences and seminars. The ICC aims to:

a) *Promote interdisciplinary collaboration in scientific domains where the notion of complexity has an essential role, such as:*

   a. *Net structure and dynamics*

   b. *Languages, computation and system simulation;*

   c. *Socio-economic system dynamics*

   d. *Natural computation and new computational techniques*

   e. *Auto-organization: organisms and aggregates*

   f. *Management and development of resources*

b) *Support the development of a common language between mathematicians, physicians, economists, sociologists, biologists, linguists and computation scientists, as well as to maintain a forum, opened to researchers in the above domains, allowing crossed fertilization of their disciplines;*

c) *Provide services to the community;*

d) *Cooperate with public and private entities, concerning mutual interests in the domain of research, education and services to community;*

e) *Promote contacts and cooperation with foreign universities and research centers and international entities*

f) *Organize advanced education courses*

g) *To attain its goals, ICC must develop the following activities:*

h) *Research projects*

i) *Seminars, conferences and similar activities*

j) *Publications*

k) Advanced education courses and research direction

(ICC 2008)

The New England Complex Systems Institute (NECSI) is an independent academic research and educational institution. In addition to its own faculties, NECSI also has co-faculties, students and affiliates from MIT, Harvard, and other universities both nationally and internationally (NECSI 2008). The areas of study of NECSI include networks, agent-based modelling, multi-scale analysis and complexity, chaos and predictability, evolution, ecology, biodiversity, systems biology, cellular response, health care, systems engineering, negotiation, military conflict, ethnic violence, and international development. NECSI also offers practical courses and a variety of literature to support the educational strand of the organisation.

2.1.2 Complex Systems Societies

The Complexity Society is based in Manchester, within Manchester business school and institute of innovation and research. The society was formed in 2002, as a result of conversations between Peter Allen of Cranfield University and Elizabeth McMillan of the Open University. These conversations resulted in the invitation to attend a meeting at the Open University to present their ideas.

The aims of the society are as follows:

"The Society seeks to CONNECT, DISSEMINATE, INFORM, TAKE ACTION.

- *CONNECTING individuals and groups enthusiastic about complexity and providing a 'home' for complexity in the UK*
- *DISSEMINATING, communicating and spreading ideas and insights using public seminars, conferences, Internet, publications*
- *INFORMING and educating individuals, organisations, government and policy makers at local and national levels*
- *TAKING ACTION, practising the values we espouse and moving Complexity into the mainstream of UK and European life"*

(Complexity Society 2008)

Today it has a growing membership and has produced a number of papers on various subjects including Organisational Design (Elizabeth McMillan 2001) and agent based modelling (Volterra Consulting Ltd 2003).

The Centre for Social Complexity (CSS) of George Mason University aims *"to advance the knowledge frontiers of pure and applied social science, by using and developing computational and interdisciplinary approaches that yield new insights into the fundamental nature of social phenomena at all levels of social complexity, from cognitive networks to the world system."* The centre *"aspires to contribute as a scholarly collaboratory of excellence, discovery, and invention, pursuing the highest standards, and functioning as an active, cutting-edge leader and participant in the emerging international computational social science community."*

The Complexity Science Research Centre is an Open University based research centre which was established in spring 2001. The Complexity Research Centre has four key objectives which are set out within four distinct themes - exploration, design, innovation and implementation. The objectives are as follows:

- *"Exploration - To carry out, encourage and assist interdisciplinary research and collaborative explorations on complexity science within the University, the UK and internationally. "*
- *"Design - To create imaginative and robust models using the latest concepts and techniques in such areas as organisations and management, computer simulations, traffic systems and environmental systems."*
- *"Innovation - To encourage creative and highly innovative research while working closely with other research groups, interested individuals and UK and international organisations. "*
- *"Implementation - To promote and apply the theoretical and practical applications of complexity science with seminars, conferences, networking, publications and consultancy."*

(Complexity Science Research Centre 2008)

This centre maintains links with the Open University Complexity Society, and has the following overall objective *"to establish the Open University as an international centre of excellence and expertise in complexity science research."*

The Centre for the Study of Complex Systems or CSCS, is a group of researchers from a variety of disciplines including biology, cognitive science, computer science, political science, psychology, information, physics and political science. Established in 1999, it has been driven by the following goals:

- *To catalyze and encourage research in complex adaptive systems at the University of Michigan*
- *To expand and coordinate educational opportunities in complex adaptive systems at UM*
- *To explore the boundaries and overlaps between the complex systems approach and more traditional approaches within the University and business communities*
- *To form a community of complex systems researchers and students-both at UM and throughout southeast Michigan*
- *To enhance the University of Michigan's world-wide reputation in complex systems research and education*
- *To raise funds through government and foundation grants, private and corporate donations to support CSCS activities.*

The centre is conducting a number of different research activities:

- *A weekly Complex Systems seminar series*
- *An annual UM-Santa Fe Institute Workshop*
- *Complex Systems Graduate Certificate Program*
- *An annual Nobel Symposium*
- *Regular co-hosting of conferences on complex systems with other research groups on campus*
- *Regular workshops on complex systems techniques*
- *A complex systems computer laboratory for teaching and research on agent-based models*
- *A CSCS web site (www.cscs.umich.edu)*

- *Complex Systems Advanced Academic Workshop (CSAAW)*
- *Complex Systems Reading Group (CSRG)*
- *Support of interdisciplinary faculty research projects*
- *Coordinated hiring of complex systems faculty in departments across UM*
- *Development of new initiatives and proposals for external funding from government agencies, private foundations and corporate partners.*

(Center for the Study of Complex Systems 2006)

These activities contribute to the goals of the centre, covering a range of different disciplines, and providing links between the centre and other institutions that are also working on similar projects.

There are also various complexity research centres within Warwick University and Hertfordshire University which provide contributions to the area of complexity science.

2.1.3 Centres of Excellence in Complexity Systems, Design and Engineering

There are a number of centres of excellence concerned with complexity systems, design and engineering throughout Europe and the rest of the world. The Systems Engineering Innovation Centre (SEIC) is one such centre, a collaboration between Loughborough University, BAE Systems and the East Midlands Development Agency. The SEIC offers both training and research facilities, working closely with industrial partners to develop effective toolsets, processes and expertise, in order to improve overall industrial effectiveness, and was set up to:

- To promote and enhance Systems Engineering as a strategic discipline
- To create multi-disciplinary engineering
- To focus on the core competencies underpinning profitability and growth, namely Systems Engineering and Project Management
- To address the Systems Engineering challenges associated with increased complexity, degree of integration, novelty and risk.

- To encourage a collaborative hub with other strategic academic and industrial partners.
- To obtain cross -fertilisation of ideas through involvement with partners in other sectors and applications
- To apply the benefits of integrated research-led teaching to the commercial world.

The research was divided into themed areas, and each theme contains its own strands of research; one of these themes is complexity, focusing on the difficulties in developing large complex systems.

Systems Engineering and Integrated Systems for Defence - Autonomous and Semi-autonomous Vehicles Defence Technology Centres (SEAS-DTC) are virtual centres of excellence established in broad technology areas that are of significant importance to the delivery of UK defence capabilities. It consists of a consortium of industrial partners including BAE Systems, MBDA, Selex, AOS, Rolls-Royce, Roke and CAE. Their strategic aim is to provide more rapid pull-through of low maturity research into the UK MOD's defence equipment programme.

The Cambridge Engineering Design Centre (EDC) conducts research activities with the aim of generating knowledge, and improving understanding, methods and tools that contribute to the design process (EDC 2008). The Cambridge University EDC conducts its activities within research themes:

- Knowledge Management
- Process Management
- Change Management
- Computational Design
- Healthcare Design
- Inclusion Design
- Design Practice
- Service Design

These themes, along with a number of collaborative projects working closely with industrial partners enable this EDC to innovate and develop new tools, ways of working, modelling approaches, knowledge and education within the complexity domain which is of utility for engineering systems.

The Newcastle University EDC has specific research and educational strands within the complexity domain, namely the "*Complex process integration and coordination*" theme (Newcastle University EDC 2008).

It has continued to develop its reputation within industry for delivering original and practical research, developing methodologies for tackling, designing, sustaining, modelling, supporting and optimising complexity within engineered systems. The EDC works closely with the EPSRC to ensure its activities provide benefit to industrial stakeholders.

The Santa Fe Institute is another centre which provides education and research material within the complexity domain (see Table 2). Also linked with industry working in a number of different disciplines, the Santa Fe Institute contributes heavily to the complexity domain as a centre of knowledge, research and industrial collaboration.

2.2 What is Complexity in Engineering?

Systems complexity in engineering is one of the challenges faced by systems engineering (IBM 2004, Charles, Philip 2003, Bar-Yam 2003).

Complexity in systems is not limited to design. Indeed, it occurs in a number of ways within systems; the requirements capture phases, design phases, integration phases, or in some cases capability capture phase. In other words, it has a presence throughout the whole lifecycle. Complexity exists within design; within new technologies (perhaps immature technologies); within the interfaces between system elements; within the size of the system; and within hardware and software. There is complexity in the organisation that develops the system; in the internal policies between business units, mergers and acquisitions; and in policies between prime and sub-contractors. There are complexities that are associated with manufacturing the system; the supply chain to manufacture, the manufacture process, and lack of commonality in manufacture procedures. There are complexities in the operation of the systems; and

in the emergent properties within new systems in an operational environment that were not predicted.

In order to improve the understanding of complexity, various different definitions were examined and the relevance of these definitions to complexity issues within engineering was identified.

2.2.1 Early Research into Complexity and Systems Design and Engineering

Kemal A. Delic and Ralph Dum (Kemal A. Delic, Ralph Dum 2006) discuss the *"Historical Prologue to Present Research"* within their paper titled *"On The Emerging Future of Complexity Sciences"*, found on the ubiquity website (Ubiquity 2008). This brief account uses their article to describe how complexity science evolved into what it is today. In their paper they discuss the beginnings of era of complexity science; they write that *"approximately as the war-related work on large scale system optimizations and intensive simulations in nuclear research. Practical needs and problems evolved into academic work engaging some of the most brilliant scientist of that time. "*

Their paper discusses the emergence of Complexity Science and the evolution of that science over the years, identifying Weaver's paper (Weaver 1948) as potentially one of the first significant works within the field of complexity science. The paper points to different research stages which span 350 years, starting with *'simplicity'*, to *'disorganised'* and moving toward *'organised complexity'*. *'Simplicity'* represents a stage of low dimensional problems with perhaps two variables; *'disorganised complexity'* involves a much higher number of dimensions or variables (billions); while *'organised complexity'* is a space within which both living and man-made systems reside.

Traditionally, the scientific approach to the understanding of systems was to vary single variables within systems, and study the effects of this change. This approach would work for small, simple systems, and an understanding of their behaviour could be formed by this method; however, with systems that are complex, varying a single value would not necessarily produce results that were readily interpretable. Understanding these systems (economic systems, large human organisations, weather, etc), and working with them required a new approach. Cybernetics, *"the science of*

control and communication, in the animal and the machine" (Wiener 1965), was a key stage within the evolution of complexity science as it offered an approach to dealing with complexity; and among the pioneers of this area of research were Ashby and Weiner (Wiener 1965, Ashby 1976).

In his work, Ashby offered a law of pre-requisite variety, a law that states that *"the complexity of a control system must be equal to or greater than the complexity of the system it controls."* This means that simple systems only require simple controllers, whereas complex systems require complex controllers.

The concept of complexity was later expanded upon by the Nobel Laureate for economics, Herb Simon (Simon 1962). Kemal A. Delic and Ralph Dum discusses, in his paper *'The Architecture of Complexity'*, *"hierarchy as a distinctive structure feature of complex systems and at the property of "near decomposability" simplifying the description of complex systems."* This idea was later enhanced by a paper, 'More is Different', by P. W. Anderson (Anderson 1972), which focuses on the idea that complex systems are more than the sum of their parts; in a sense, complex systems represent a Gestalt, one which exists as a result of various connections between elements within systems that exhibit emergent properties that otherwise could not be predicted. Anderson, Gell-Mann and Kenneth Arrow later went on to found the Santa Fe Institute in 1984. Kemal A. Delic and Ralph Dum's account of the history describes this as a "milestone in the development of a science of complexity".

2.2.2 The Development of Measures of Complexity

Cosma Rohilla Shalizi's, in his brief notes (Shalizi 2008) which cover the development and early history of complexity measurement, states that, usually, the first complexity measure is traced back to Kolmogorov complexity (later renamed "algorithmic information"), a measure of the shortest computer program capable of producing a given output (string of characters). Kolmogorov complexity is not easily computable, and does not specifically calculate the complexity of a string; rather, it describes how random the information contained within that string is. It is, therefore, a measure that is very difficult to apply, and has described as *"solemnly taken out, exhibited, and solemnly put away as useless for any practical application"*. Cosma Rohilla Shalizi goes on to say that "*generally speaking, complexity measures either take after Kolmogorov complexity, and involve finding some computer or abstract*

automaton which will produce the pattern of interest, or they take after information theory and produce something like the entropy" (Shalizi 2008).

Those measures that have been developed as a consequence of Kolmogorov include Bennett's logical depth (Bennett 1988), other information theory measures (Shannon 1948), and computational complexities. Since these initial ideas, complexity measures have been under development within a number of different domains, including engineering. Connections with network theory have enabled complexity measurements to be applied to architectures and system designs, along with biological ecosystems; and the field of research continues to expand.

2.2.3 The Development of Network Theory

Marta Gonzalez describes networks as *"a set of items, called nodes or vertices, with some connections between them called links or edges."* (Gonzalez 2006). Systems that take the form of networks exist all around us - financial systems, the internet, social networks (facebook, MySpace), infrastructure (roads, communications, power distribution), information networks in defence systems (Network Enabled Capability, NEC); even within the biological world networks exist, such as food chains or ecological networks.

While all very different, these networks all contain nodes and links, and can be analysed using graph theory (Reinhard 2005). Graph theory can be used to understand networks, and includes concepts of network theory such as spanning trees (Irvine 1996), where links between nodes have "length" and direction, so information flow throughout the network can be modelled. Network theory can be applied when modelling complex systems, determining the potential paths for information between nodes (links to Senge's (1994) concepts of dynamic and detail complexity, see section 2.6).

Research into the social aspect of network theory has included the "strength of weak ties" within social networks (Tesson 2006). Mark Granovetter conducted research which looked at where job applicants heard about the jobs they were applying for (Granovetter 1974). This research determined that the strong links within the applicants' social network (i.e. friends, family, immediate work colleagues) were often not responsible for providing information which ultimately leads to finding a job; instead, it was the weaker links – acquaintances, for example - that provided this

information. Granovetter suggested that these weak links acted as *"network bridges"* between two networks with strong connections that would otherwise remain unconnected.

The "small world" concept of network theory developed by Watts and Strogatz, further expanded on the idea of the strength of weak links by seeking a mathematical explanation. The model developed by Watts and Strogatz showed that these "small world" networks were neither random nor regularly organised, but were, in fact, a structure that sat between the two. As with Granovetter's concept of the strength of weak links, Watts and Strogatz determined that small networks contain long length links within the structure that are able to join together two highly connected nodes or hubs.

The concept of hubs has been further expanded by Albert-Laszlo Barabasi (2002), who argued that there are some nodes which are often more highly connected than others, and these are known as "hubs" within networks. Ideas such as this can be applied to failure analysis within systems, whereby one system component is highly connected to other system components, and consequently is a leading cause of overall system failure as a result of being on many information flow paths within the network.

Network theory, combined with graph theory, continues to be a driving force in the development of concepts to support, understand, quantify and model complex systems, be they social, engineered, or mathematical in nature.

2.2.4 Agent-Based Modelling

An agent-based model (ABM) is a computational model for simulating the actions and interactions of autonomous individuals in a network, with a view to assessing their effects on the system as a whole. Each element or agent within the network acts in its own interests, with individual goals or objectives; simulating the interactions of these agents enables the identification of the emergent behaviour, or complex phenomena that results from them. ABM combines elements of game theory (game of life), complex systems, emergence, computational sociology, multi-agent systems, and evolutionary programming along with using Monte Carlo Methods to introduce randomness within simulations. The simulation technique is powerful, and suggests a

variety of applications in a number of different disciplines (engineering, commerce, AI).

The Multi-Agent Systems Lab (MAS Labs 2008) is part of the Computer Science Department at the University of Massachusetts at Amherst. It is concerned with developing and analysing sophisticated artificial intelligence problem solving and control architectures for single and multiple agent-based systems. Organisational design, performance and adaptation research projects within KB-ORG (an automated knowledge based organisation designer for multi-agent systems) were created for developing organisations with different requirements and environment expectations (MAS Lab 2008).

Engineering projects have included the Autonomous Negotiating Teams (ANTS) project. This project analyses the coordination of constrained resources in an uncertain real-time domain. Within this project the coordination of radar sensors was explored when tracking targets, thus developing negotiating strategies dependant on the demands of the environment on the system at any particular time (MAS Labs 2008).

ABM can be used to model organisational structures where agents may be business units, individuals, or groups within an engineering development team. ABMs may also be applied to engineering designs, such as autonomous sub-systems which exchange information with the individual sub-systems, making decisions based on this information, such as autonomous reconnaissance systems.

2.2.5 Dynamical Systems and Chaos

Dynamics is the study of change, and a Dynamical System is a system containing variables which interact and change in time. Examples of dynamical systems are the stock market and the economy, climate, population, ecosystems, and also mathematical systems (Spiegelman 1997).

Lorenz studied mathematical dynamical systems and developed the concept of strange attractors within those systems, which show how the trajectory plot of a dynamical system moving through time in 3D space, in some cases, tends towards specific points - attractors.

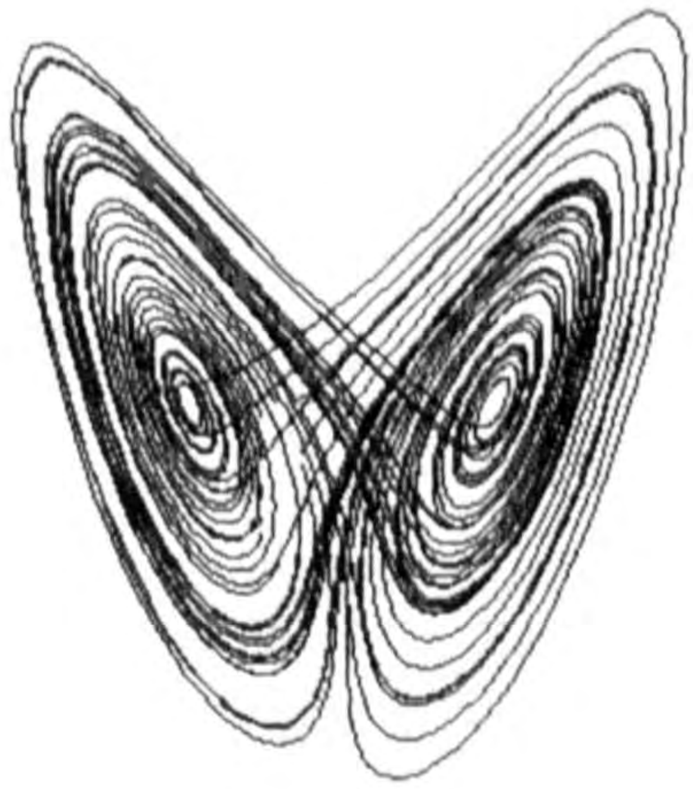


Figure 3 - Lorenz attractor.

Figure 3 (Filip Larsen 2001) shows the Lorenz Attractor. The system never reaches a steady state, but remains chaotic, shifting from attractor to attractor, and is highly sensitive to initial conditions.

The notion of the "butterfly effect" illustrates how small changes in the initial conditions can lead to dramatic changes in later system behaviour. Complex systems sometimes exhibit these characteristics, in that small changes in inputs to the system can lead to effects which totally change the behaviour of the system, and perhaps lead on to unexpected changes in system behaviour (emergent properties).

An interview with Professor Robert MacKay (Warwick University 2008) includes an illustration of a dynamical model of infection spreading between individuals. As the Professor notes in the interview, although the model exhibits the appearance of random chaos, sometimes the infection dies out within the model, and at other times it spreads. The Professor comments that the behaviour and overall result of the model is highly dependent on the initial conditions, and that this is typical of dynamic systems at the edge of chaos.

2.2.6 Scaling and Power Laws

Scaling behaviour and power laws are often found within highly complex dynamical systems. Power laws enable relationships to be found between variables within dynamical systems; examples of power laws are metabolic rate and mass, optimum cruise speed, and size of flying objects.

Changes in the relationship between two variables in some cases are independent of scale, an example of which is the optimum cruise speed and size of flying objects. This relationship has been followed through from objects such as insects right through to 747 civil airliners.

Scaling and power laws enable a much richer understanding of the relationships which can exist within dynamical system variables. These relationships that can be established within systems might otherwise have been overlooked or missed, and may be crucial to understanding how the system works.

In the field of complexity, relationships between variables is extremely important - increasing the ability to understand variable relationships means that networks of interaction within systems can be greatly increased. This improved knowledge and increase in the understanding of the relationships within systems means that science is able to determine the reason for the relationships existence and improve knowledge of that system.

2.2.7 Cellular Automata and the Edge of Chaos

Cellular automata is name given to the relationship between a cell within a network and its neighbours. Examples of such a system is the "game of life" (Conway 2000) in which a cell is either lit or kept in darkness, and will change colour depending on the current state of neighbouring cells on the next clock cycle, depending on a user-provided rule. In this example with a grid of perhaps 50 by 50 cells, with a randomised initial state for each cell (lit or dark), processed clock by clock, the cells change their state depending on the user-defined rule. Initially, the behaviour of the grid seems to be random - cellular states flicker with little or no pattern; however, as the process continues regular patterns emerge, in some cases rapidly. The point at which a system (like the game of life grid of Conway), moving from a state of disorder or randomness (a chaotic state) to one of order and regularity with little to no transition period, is known as the *"edge of chaos"*.

Jeffery Johnson's article, entitled "Can complexity help us understand risk?" (Johnson 2006), describes the history of the development of cellular automata research, beginning with Conway's game of life.

Wolfram (1994) studied cellular automata and was surprised to find that the outputs were not simple regular patterns; rather, what emerged from running these grids of

cells with a set rule was a *"pattern that seems extremely irregular and complex."* Wolfram later determined four classes of output from such a system:

1. All cells become alive or all the cells die, irrespective of the initial conditions.
2. The final state consists of simple structures that may repeat.
3. The behaviour is more complicated with triangle and small structures appearing at random.
4. A mixture of order and randomness; localised structures are produced which on their own are simple but interact with other structures in complicated ways.

(Wolfram 1994)

Langton suggested a value lambda to investigate the behaviour of cellular automata; as this value increased the behaviour of the system was found to progress through Wolframs classes in the order 1, 2, 4 and finally 3. Langton discovered there was a critical value at which class 4 existed, between ordered, structured, and random behaviour. This was later called the "edge of chaos", a point at which the behaviour is a mixture of structured and chaotic behaviour.

2.2.8 Simulation

Edmonds' paper titled "Simulation and Complexity" (Edmonds 2005) begins with the question *"Do simulation models really help us understand complex phenomena?"* In some domains the system is sufficiently complex that standard equation-based and statistical models are often impractical or even impossible to apply (Edmonds 2005), posing its own problems and issues. Often it is not possible to model the phenomena completely; in order to do so would mean a direct replication of it and this is often impractical and defeats the point of modelling. Edmonds also states that *"inappropriate use of any kind of modelling generates more confusion than it sheds light"* and states that this danger is increased for complex phenomena.

In his paper Edmonds identifies three distinct problems regarding simulation modelling of complex phenomena:

- *The presentation of models that are intuitively plausible but with little solid relation to their intended domain. Such models are developed to aid conceptual or formal exploration but then convince their authors to such an*

extent that they then project the model upon their domain, drawing unjustified conclusions about it.

- *Different types of models are conflated in terms of use and judgement, so that a model that was developed and validated for one kind of use is then used or interpreted as if it were valid for a different use. For example a model may be developed as if it were a predictive model but then tested according to criteria suitable for an explanatory one.*
- *Since programming is apparently much more accessible than doing mathematics (going by the numbers able to do each) – many more people can build models and discover something. This has both positive and negative aspects, its accessibility has the effect of democratising a field making it less prone to persuasion via mathematical opaqueness but on the other hand the lack of the implicit filter that mathematical competence means that there are more badly constructed or sloppily applied models around to confuse.*

Edmonds also identifies the use of models.

- *Simulation models can be used for many purposes, including:*
- *Entertainment, as in 'SimCity'*
- *Art, to produce pleasing or expressive artefacts*
- *Illustration, to animate a phenomena one wants to communicate*
- *Pseudo-Mathematics, to determine the properties of the simulation in the abstract*
- *Mediation, as a medium with which to communicate or negotiate*
- *Design, as a way of testing an idea for a design before most costly construction occurs*
- *Science, i.e. helping to understand observed phenomena*

Agent-based modelling or simulation consists of software objects (agents) which interact with each other and a virtual environment. Each agent is independent and can react to and have influence upon the virtual environment within which it resides. This method of modelling or simulation can be used to create models of complex systems.

Examples of models of this nature are computer programs such as "Sim City" (Electric Arts 2008), a computer based game/simulation of a city environment where independent agents interact with each other based on a set of rules defined by the game player. The results of these generated rules (assigning land to residential, commercial or industrial zones, for example) then influence the behaviour of the agents, and in turn this behaviour generates feedback to the game player in the form of economic prosperity, crime rates or approval rates within the virtual city.

Nigel Gilbert of the Centre for Research on Social Simulation (Nigel Gilbert 2004) talks of computer simulations using agent-based modelling approaches, such as the SimCity computer game. He notes that *"the breakthrough came when it was realised that computer programs offer the possibility of creating 'artificial' societies in which individuals and collective actors such as organisations could be directly represented and the effect of their interactions observed."* He also notes that *"another benefit of simulation is that, in some circumstances, it can give insights into the 'emergence' of macro level phenomena from micro level actions."*

A lot of additional simulation tools have been developed within engineering that model and simulate engineering systems. These simulations include packages such as Telelogic's Rhapsody tool which can simulate systems created in SysML. These models can be used to dynamically model systems in software and hardware with software representations. These simulations have proved to be extremely effective in the product development process, and the use of 'synthetic environments' within which real or mock systems can operate is becoming a frequently used tool within industry when developing large systems with many interconnections, or when connecting a series of systems for the first time (networked enabled capability).

2.3 Definitions of Complexity

There are a large number of different definitions of complexity from a variety of different disciplines. In order to enhance our understanding of complexity, an appreciation for the different definitions and their implications is necessary.

Some of the definitions are specifically engineering focused, and perhaps relate to specific engineering domains such as software or hardware. Others are biological or mathematical. However it is necessary to explore as many avenues as possible and

different disciplines will have different focuses for their definitions. Before this can be done, an understanding of what is required by a definition must be formulated.

The International Organisation for Standardisation (ISO) adopts a standard approach when creating definitions (ISO 1999), and this approach is the basis on which the definitions collected here will be compared and contrasted. The standard states that definitions should:

- Be as concise and brief as possible while being complete.
- Describe what the concept is, and not what it is not.
- Avoid circular definitions; a definition of the first concept needs explanation of that concept, and that explanation refers back to the first concept.

These requirements are considered when analysing definitions of complex and complexity in this section.

The obvious place to start when looking for any definition is the dictionary. The American Heritage Dictionary of the English Language (American Heritage Dictionary 2000) describes complexity as *"the quality or condition of being complex"* and *"something complex: a maze of bureaucratic and legalistic complexities"*. The dictionary also describes complexity as "*consisting of interconnected or interwoven parts; composite. Involved or intricate, as in structure; complicated. Having parts so interconnected as to make the whole perplexing."* (American Heritage Dictionary 2000). The Oxford English Dictionary (Oxford University 1992) describes complexity as a *"composite nature or structure"*, an *"involved nature or structure, intricacy"*, and also *"an instance of complexity; a complicated condition; a complication."* The dictionary also describes complex as *"consisting of or comprehending various parts united or connected together; formed by combination of different elements; composite, compound. Said of things, ideas, etc."* and also as *"consisting of parts or elements not simply co-ordinated, but some of them involved in various degrees of subordination; complicated, involved, intricate; not easily analysed or disentangled."*

The dictionary definitions are not directly related to complexity within engineered systems. The first definitions from the American Heritage Dictionary are not particularly useful, as they simply denote complexity as a condition of being complex

without offering any further information – it is, then, a circular definition. The latter definition from the American Heritage Dictionary is more informative, describing complexity in terms of intricacy within systems and the difficulty in understanding such systems, making the whole 'perplexing'.

Oxford's definitions like the American Heritage Dictionary offer two rich descriptions, without much reference back to other concepts, and three somewhat less useful definitions. Descriptions of complexity, such as a *"composite nature or structure"*, a system with *"involved nature or structure, intricacy"*, or *"an instance of complexity; a complicated condition; a complication"* offer little insight into what complexity actually is. As with the American Dictionary, the last description is a circular definition and of little use on its own.

However, in both sets of definitions there are a number of key words that describe complexity well, namely, intricacy, intricate, composite, connected, elements, appear in the definitions, along with concepts such as disentangled and not easily analysed, which bare a good resemblance to large scale systems designed today. The *"elements"* could be interpreted as sub-systems, the *"intricacies"* and *'entanglement'* as the interfaces and the level of coupling between those interfaces. Large scale systems developed by defence contractors do consist of sub-systems that are not simply connected; rather, they are connected with data buses, data lines internal software links, and mechanical links.

A number of definitions of complexity follow the theme of intricacy between components, or elements. Evans (1987) is an example - he describes complexity within the software domain in a more concise fashion as *"the degree of complication of a system or system component, determined by such factors as the number and intricacy of interfaces, the number and intricacy of conditional branches, the degree of nesting, and the types of data structure"*. This definition expands on the intricacy aspect of complexity in systems by providing a source of that intricacy. Although focused on software systems in particular (they are the focus of the complexity sources) Evans identifies a number of factors that influence complexity (or the level of intricacy), introducing the concepts of nesting, conditional branches and data types and structures which are appropriate for the software domain.

There are complexity measures specific to software systems, Cardoso uses this description of process complexity, utilising that taken from (IEEE 1992): process complexity is "*the degree to which a process is difficult to analyze, understand or explain. It may be characterized by the number and intricacy of activity interfaces, transitions, conditional and parallel branches, the existence of loops, roles, activity*". (Cardoso 2005) This is very similar to the definition of Evans (1987), with added specific software terminology (loops, roles, transitions, etc.).

Evans' definition is applicable not only to the software domain, but also to the wider systems engineering domain. Conditional branches (see Cyclomatic Number for measurement in section 2.7 Complexity Measures) and nesting within systems are not only properties of software systems but exist in organisational and technological systems as well; as a result, the definition is appropriate for the systems engineering domain, as well as other domains while meeting the criteria set for the definitions outlined above (concise, not circular, and does not describe complexity using negatives).

Other definitions or descriptions of complexity also take up this point. J. M. Sussman (1999), for example, states that "*a system is **complex** when it is composed of a group of related units (subsystems), for which the degree and nature of the relationships is imperfectly known. Its overall emergent behaviour is difficult to predict, even when subsystem behaviour is readily predictable. The time-scales of various subsystems may be very different (as we can see in transportation -- land-use changes, for example, vs. operating decisions). Behaviour in the long-term and short-term may be markedly different and small changes in inputs or parameters may produce large changes in behaviour.*" Although not specifically created with the engineering domain in mind (it was actually created for transportation systems), there are similarities, or common concepts in this definition, to concepts of complexity found in engineering. The recurring theme of interwoven sub-systems (mentioned directly) and their not fully appreciated or understood intricacy (which is 'imperfectly known', perhaps due to the difficulties in modelling systems in their entirety), along with the additional references to the difficulties in understanding the whole system and predicting emergent properties is quite appropriate to engineering issues. There are other relevant aspects of this description to engineering as well - the concept of different time scales for sub-systems is not mentioned in the previous definitions.

This is also present in engineering design and systems engineering, with many products having sub-systems that are vastly different, some human, some electrical, some mechanical and some software. These different sub-systems run within very different time frames - electrical and software systems are very fast, processing information extremely quickly; human interaction is slower, but despite this they must be integrated to form a complete coherent whole.

The final aspect covered by Sussman's definition, is the sensitivity of complex system behaviour to small changes in input parameters. This sensitivity of developed systems is indeed an important issue for systems engineering, as it implies that small changes in conditions cause large variations in behaviour (see section 2.2.5) making modelling and predictions awkward and error prone. This is different from system instability - complexity occupies the space between stability and instability (although leaning towards instability).

Sussman's definition (or, perhaps more appropriately, his description) meets the criteria for a definition - it is concise and complete, with no circular references or concepts; it does not describe complexity using negatives, and it has highlighted a new concept that should be considered, although the time scale differences in operation is perhaps more a cause of complexity than a definition of it. Finally the addition of system sensitivity is useful, but is perhaps a result of the lack of understanding or inability to model or predict the system correctly.

Modelling and predictability in systems is also a theme found in definitions of complexity. Edmonds suggests that a complex system is "*a system where it is difficult to formulate its overall behaviour in a given language, even when given reasonably complete information about its atomic components and their inter-relations*" (Edmonds 1999), and that it is "*not easy to understand or analyze*" (Edmonds 1999). This indicates that systems complexity is related to the difficulty in understanding the system, or modelling, estimating or predicting the behaviour of systems. The IEEE also describes complexity in a similar manner, as "*the degree to which a system or component has a design or implementation that is difficult to understand and verify*"(IEEE 1990), but a lack of understanding perhaps points to ignorance rather than anything else – although, as Edmonds argues, "*complexity is distinguished from ignorance*" (Edmonds 1999), it cannot be an excuse for it. Sussman's (1999) idea of system sensitivity could be the result of ignorance in the

engineering process. Ignorance or lack of understanding of a system does not necessarily mean that the system is complex; in light of this the IEEE definition is incomplete, whereas Edmonds' definition includes the caveat that complexity is a lack of understanding despite having *"reasonably complete information about its atomic components and their inter-relations"* (Edmonds 1999), which eliminates room for ignorance.

Similar to un-model able, unpredictable, and non-understandable definitions of complexity, Langefors (1967) describes an unperceivable system, which can come *"to mean a system such that the number of its parts and their interrelations is so high that all its structure cannot be safely perceived or observed at one and the same time."* This definition is appropriate for complex systems; indeed an unperceivable system is beyond comprehension, beyond understanding. It is complex, which makes this definition appropriate. Again it is concise, there are no circular concepts; however, it is just an addition to the definitions of complexity that hinge around the unpredictable nature of complex systems. However this concept is perhaps more in tune with the scale issues of complex systems, with the number of parts, and interrelations. Scale is a common problem in definitions of complexity; the two do not necessarily go hand in hand with each other (such as the Boolean NK networks of Kauffman), as a result the definition is perhaps incomplete.

The definitions of complexity outlined thus far centre around complexity properties within systems; the difficulties in modelling or predicting complexity, the levels of intricacy, the coupling within the system, the levels of variety, the size, the loops within the system. However, complexity within engineering systems can be split up into those complexities that are unavoidable or intrinsic to the system and those that are a result of the development process, or induced complexities. Musès (Musès 2002) proposed splitting complexity into three "kinds", and developed definitions or descriptions of those complexity "kinds", the first of which is:

"Complexity I – Inherent in almost all natural phenomenon."

Musès uses *the "wealth of species, the incredibly rich repertoire of weather and sunsets, the turbulent flow of rapids and waterfalls, the endless kinkiness of coastlines and individuality of tides"* (Musès 2002) as examples of this. Musès' Complexity I (although describing natural systems) can be adapted to refer to the intrinsic

complexity of the system. For systems engineering, Musès' Complexity I can be re-phrased as;

Complexity I - The ineluctable complexity which exists within a system as a result of the irreducible functional or capability requirements placed upon it.

Musès continues to describe additional levels of complexity within systems, focusing on over-complication, which he describes as *"simply needless over-complication arising from not seeing things clearly enough – either through lack of experience or ability, or from an excess of obfuscating conditioning beclouding the perceptive mind."* Systems engineering over-complication is induced complexity within the development cycle of products. The difference between Musès' definition and the perception of the systems engineering world is that over-complication is not necessarily 'needless' as a result of being inexperienced of unperceptive, but can actually be both needed and needless.

Experience in some cases would tell us that the over-complication is in fact beneficial as technologies are mature; in other cases over-complication is in fact an accidental and perhaps detrimental result of the development process.

Musès' definition for 'Complexity II' is the following:

*"**Complexity II** - Simple over-complication caused by lack of insight into the problem."*

This needs adaptation before it can be applied to systems engineering problems. From an engineering viewpoint this is clearly the addition of unnecessary complexity into the system that is not beneficial to the system or its development lifecycle.

Complexity II - The avoidable and detrimental unnecessary complexity which exists within a system as a result of the development process.

Finally Musès describes the third kind of complexity as:

*"**Complexity III** – Problems cannot be solved by pseudo-solutions that are not on target, such as whitewashed solutions for large scale problems in elaborate often logorrhoeic terms of complexity of the second kind with large amounts of inappropriately misapplied complexity of type I. Simply diagnosing systems is not enough here."*

Complexity III is quite an appropriate definition when considering the new challenges faced by systems engineering (evolving requirements, capability requirements). It suggests that the process (the solution) may be incorrect and a new solution specific to the procurement strategy may be required (UK Ministry of Defence 1999).

Complexity III focuses on the point that, often, complexity problems cannot be solved using already existing solutions that are not specific to the system in question, but solutions that are created using the long over-complicated Complexity II of miss-applied Complexity I. Complexity III means independent strategies must be realised to find solutions.

In systems engineering, the equivalent is the application of the same engineering processes to all product development programmes, or the same metrics as the method of quantifying various aspects of the development cycle or product. The suggestion by Musès is that often, the process of engineering in some cases may be bespoke to that development programme. This of course goes against the general perception of systems engineering, the lifecycle which is applied to development programmes of all shapes and sizes as a blanket "whitewashed" solution.

Another definition of a complex system is one that exists at the *"edge of chaos"* (Waldrop 1987). This term, although not strictly a definition in its own right as it does not refer to the complexity concept, refers to a system whose behaviour is almost random, at a point where complexity within the system is said to be at its maximum.

Stuart Kauffman developed a game of life model based on Boolean NK networks (Kauffman 1993); these networks were simulations of life based on a natural selection principle. These models demonstrated the transition from chaotic behaviour to order, demonstrating the principle of "self-organisation" in cellular automata.

"Irreducible complexity", a term first coined by Michael Behe (2006), and revisited by William Dembski in Irreducible Complexity Revisited, refers to a system *"composed of several well-matched, interacting parts that contribute to the basic function, wherein the removal of any one of the parts causes the system to effectively cease functioning."* (Dembski 2004). Despite the fact that this complexity definition was initially developed in support of intelligent design, the irreducibility idea could be applied to a system which cannot be reduced any further, indicating that there are no added induced complexities within this system which result from process.

2.3.1 Complexity Characteristics

Charles Perrow describes complex systems as having the following characteristics:

- *Proximity of components that are not in a production sequence.*
- *Many common mode connections between components in a production sequence.*
- *Unfamiliar or unintended feedback loops.*
- *Many control parameters with potential interactions.*
- *Indirect of inferential information sources.*
- *Limited understanding of some processes.*

(Perrow 1999)

Commonality between these characteristics, that define a complex system for Charles Perrow, exist between a number of the definitions explored elsewhere in this thesis (Edmonds, Musès, Sussman, the dictionaries, etc.). The context within which this definition or description of complexity arises is the sphere of process, be it the engineering process, maintenance process or operating process. Nonetheless, the characteristics here are quite pertinent to systems engineering; the more specific reference to "unintended feedback loops" suggests that even closely and carefully developed systems will have unintended behaviours (emergent properties) that could be detrimental to system performance, or as Perrow points out, safety.

Hitchins writes that *"a generic system can be stated as "complex" when it is composed of an open set of complementary, interacting paths with properties, capabilities and behaviours emerging both from the parts and from their interactions"* (Hitchins 1992). Like a number of the other definitions this, too, focuses on the intricacies within the system elements and interfaces, and with emergent properties arising from these interactions. The definition is not so useful in the engineering domain, as it is limited in its description of complexity. There is no mention of testing, the number of interactions, or the lack of understanding or comprehension of the system.

Further definitions of complexity or descriptions of complexity centre on the inability to describe systems, and the inability to model or predict their behaviours.

"Complexity is the property of a real world system that is manifest in the inability of any one formalism being adequate to capture all its properties" (Mikulecky 2007).

This is a definition, again, that only partially tells the story for engineering systems; although relevant, there is no reference to the intricacies that are the root cause of the difficulty in describing complexity. However Mikulecky does not finish here – he goes on to state that complex systems are non-fragmentable:

"It is non-fragmentable. If a complex system were fragmentable it would be a machine. We require the distinction to be dichotomous. Therefore complex systems are not fragmentable. That is not to say that they are incapable of being reduced to parts, but such reduction destroys important system characteristics irreversibly."

(Mikulecky 2007)

In fact Mikulecky continues his description of complexity, and it is highly detailed one; he further explains that a complex system:

"Consists of real components that are distinct from its parts. At least one set of these components is defined by its functions. These functional components are not simply collections of parts. If they were the system would be fragmentable in the above sense. These functional components are therefore defined by the system and have their ontology dependent on the context of the system. Outside the system they have no meaning. Further, if they are "removed" from the system in any way the system loses its original identity as a whole system."

(Mikulecky 2007)

In its entirety, Mikulecky's description is very thorough. Although not referring to intricacies and coupling within systems, it does acknowledge the presence of properties within systems that contribute to their complexity. In addition to this, he advances a concept of complex systems as being unfragmentable or irreducible. A variation, and another description or definition used to describe complex systems often concerns ideas of reducibility. Complex systems are considered to be irreducible; indeed it is often considered that the only way to model the system correctly is in fact to replicate it.

What is needed is an understanding of complexity specific to the engineering domain, one which incorporates elements of definitions or complexity characteristics from

other disciplines that are applicable to engineering. All the definitions so far encountered seem to fit one, or a number of up to four distinct themes:

- Irreducibility or unfragmentable.
- Intricacy and coupling.
- Indescribable or cannot be modelled without complete system replication (intricacies and understanding).
- Level of understanding (ignorance, modelling and predictability)

In engineering, a definition appropriate to the challenges faced by systems engineering is required. Systems engineering, in particular the integration within the systems engineering process, is interested in the intricacy and coupling of sub-systems, the ability to model and predict that sub-system and overall system (perhaps in terms of behaviour, reliability, sustainability, durability and effectiveness). In this context the last two definition types listed above are applicable; however, it is not obvious how irreducibility fits in the systems engineering domain. Obviously the system, if dissected, will not resemble the complete system, but in systems engineering and, in particular, in defence system engineering, reducing the system (perhaps due to failure or error) is not usually a problem due to redundancy, or resilience to failure. In this sense the system still operates despite being reduced; as a result developed systems are in fact reducible.

Finally the level of understanding of the system in question is of vital importance, but again must not be confused with ignorance (Edmonds 1999). The idea is that our understanding of the component parts of systems are, in fact, at a high level, but our understanding of the system as a whole is in fact limited. Our limited understanding of the system as a whole, and its operation is related to predictability, or our ability to model the system. Beyond this there are systems that are simply beyond our understanding, systems in which no patterns of behaviour can be identified.

As result of all of this, the definition for complexity in systems engineering is an amalgamation of different definitions. The appropriate definition of complexity includes the intricacy of the interfaces, an assumed level of understanding of the components and interfaces, and the difficulty in describing the system.

The definition used within this thesis is a combination of Evans and Sussman and describes complexity (in systems engineering) as:

"A system is complex when it is composed of a group of related units (subsystems), for which the degree and nature of the relationships is imperfectly known. Its overall emergent behaviour is difficult to predict, even when subsystem behaviour is readily predictable." (Sussman 1999) *"The degree of complication of a system or system component* [is] *determined by such factors as the number and intricacy of interfaces, the number and intricacy of conditional branches, the degree of nesting, and* [the overall system structure]". (Evans 1987)

2.4 Complex System Examples

In the natural world complexity is in abundance, and ranges from biological to geographical systems (Magee, Weck 2004). Ecosystems and their inherent complexities, and intricacies such as food chains, climate and weather systems are massively complex, although they have been modelled to predict potential effects of global warming (Washington 2005), evolution of species (Adami 2002), landscape transformations and erosion.

There are also a number of systems described as complex in the economic world and the engineering world. World economics and the World Trade Web (WTW) are other complex systems (Arthur 1999, Li, Jin et al. 2003). These two have network properties, with nodes or elements and their intricacies, interfaces, relationships and resulting behaviours (synchronisations) in much the same way that designed engineered systems have; however the scale is somewhat different.

Understanding the effects and causes of climate change is also a highly complex exercise. Climate change and its effects are the result of a vast number of components, variables and intricate interactions existing across a range of disciplines (physics, geography, chemistry, biology, etc.), and these need modelling in a concise and valid manner.

Climate change is a result of interactions between different elements; as with any system, these elements could be thought of as sub-systems, and their interactions as interfaces. An example of the elements within climate change are rainfall, heat retention of surfaces, evaporation rates, particles in the atmosphere (Stanhill 2007), gas content of the atmosphere, oceanic currents, ocean level, ice caps, carbon

emissions, and much more. The effects of climate change are also intricate - the increase in overall temperature means that where food is imported from has to change, as those countries producing some foods will no longer be able, or may have lower yields due to environmental conditions no longer being suitable; conversely, the warming climates of other countries will accommodate the growth of new produce (Parry, Rosenzweig et al. 2004).

Climate change is caused by a vast number of different things: solar radiation, gas composition of the atmosphere, sea temperatures, the size of land masses, sea masses, and more. The effects of climate change are also vast, both directly and indirectly: rainfall changes, temperature changes, food production impact, flooding, desertification, erosion, exports, economies and more. Climate, or the climate system, is a complex system with a large number of interactions, intricacies and different elements, and is very difficult to model (Jim 2007).

Biological complex systems exist in ecosystems or cellular systems like the immune system (Cadenasso, Pickett et al. 2006, Ferdinando, Genuario et al. 2004, Deem 2005, LLC 2008). Ecosystems consist of climatic and biological components: sunlight, temperature, rainfall, herbivores, omnivores, carnivores, plants, immigrant species, and indigenous species. These interact in an intricate way through carbon cycles, oxygen cycles, reproductive cycles, through food chains and much more. Cellular systems, such as the immune system or the human body, also contain different elements or components with highly interactive properties.

As of now we are unable to accurately predict or model all aspects of these systems. The human body has many illnesses that are not understood, Lupus for example (Anderson 2008); weather systems and climate are very complex and our predictions are not always accurate (Jim 2007); ecosystems are fragile and the effect of immigrant species or changes in variables is not always understood or modelable (Bolte, Hulse et al. 2007). These are systems where the knowledge of their component parts is reasonable, and certainly not completely ignorant.

Systems developed within industry do not have the same scale present in biological cellular, ecosystems ecological complexity (Myers 1992), or climate systems: they are not as complex as those that exist in the natural world, they do not contain the level of intricacy or the number of different variables. The internal chemical reactions of

living cells consist of thousands of enzymes, producing thousands of chemical reactions every second. With a collection of millions (perhaps trillions) of cells which may form a single body, the number of reactions taking place at any one moment is enormous. If one imagines a cell with a thousand reactions every second, and a body made up of ten trillion cells, that is ten thousand trillion reactions every second in the cells alone.

Products developed by industry may be fast, and consist of a high number of components which make sub-systems (cells making organs), but the complexity is not on the scale of the natural world as these developed systems have a much lower number and lower level of intricacy of component interactions. The space shuttle consists of approximately 2.5 million parts (Kridler 2002) and the Boeing 747 approximately 6 million parts (Boeing 2007). The difference in the scale is obvious, although larger in size; the number of components in developed systems are not even close to number of "parts" (cells, etc) within a human body numbering in the trillions. The interactions and intricacies within software and electrical systems are relatively trivial in comparison to the weather, climate, biological or cellular systems within the human body, and as a result make modelling and prediction easier.

The systems designed and developed through the systems engineering practice are not random; they have architectures and designed structures. They consist of sub-systems with pre-defined behaviours or properties which in turn consist of known and understood components. The system is then formed from the collection of these sub-systems (which in some cases may become another sub-system higher in the hierarchy). Quite often the interactions of components are limited to the sub-systems, and the sub-systems interact with each other as whole entities, these interactions are then managed in the design process with interface specifications (Haskins 2006).

In the engineering world, systems complexity is often associated with scale; the larger product or system the more complexity is associated with its development. Of course in reality complexity can exist at various scales, from the small cellular systems to large scale climate models, ecosystems or aircraft. From an interaction and intricacy viewpoint, the scale of the complexity in developed products is low, but none the less this complexity is significant as any extra effort required is a cost.

In engineering systems the complexity comes from:

- Amalgamations of companies and their interactions.
- Intricacy and coupling of the system interfaces along with the nature of the architecture.
- The size of the system.
- The number of sub-systems.
- The scope of the product; support and operation as well as the design, development and testing.
- The variation in the skills required developing the product; software, hardware, fluid dynamics, control.
- The budget and resources available to produce the required product.

(see 2.5 The Causes of Complexity)

Examples of complex systems produced by engineering processes and composites of companies include:

- Submarines (Naval-Technology, BAE Systems)
- Aircraft, both military and commercial(Boeing 2007, Eurofighter Jagdflugzeug GmbH, Air Force Technology, Hayles 2005)
- Ships, both military and commercial (Naval Technology)
- Land vehicles, or ground effect (General Dynamics UK, UK Ministry of Defence, Pike 2008, R & F Defence Publications 2007)
- Networked Capability (UK Ministry of Defence 2005).

There are other examples of engineered complex systems; these systems may be organisational systems, communication systems, manufacturing systems, logistic systems or computer network systems. The examples here reflect the typical products of defence industries with some commercial equivalents. However these systems consist of elements and interfaces which can be (with a lot of effort) counted and described from documentation and specifications.

Table 3, taken from (CSCS) shows three examples of complex systems in the natural world. These systems contain a large amount of interaction and also exhibit dynamic and adaptive behaviour.

Field	Population Epidemiology	Immunology	Ecology
Agent	Susceptibles	Cellular material	Individual animals
Heterogeneity	Risk factors	Antigens, antibodies	Eating, nesting, breeding habits
Organisation	Social groups	Cellular organisation	Schools, herds, food chains
Adaptation	Infection avoidance or spread	Immune response	Hunting, mating, security
Feedback	Disease spread	Immune response	Success of failure
Dynamics	Disease spread	Infection spread	Predator-prey interactions, competition
Emergent Behaviour	Epidemics	(Un)healthy	Extinction, niches

Table 3 - Complex system examples.

These natural systems span a different scale (as discussed above) to hardware engineered systems, but the scale and intricacy of an engineering system is significant for engineering as a discipline. Engineered Complex Systems (ECS) are not of the same scale as natural systems, but this is the complexity that requires understanding.

2.5 *The Causes of Complexity*

Causes of complexity, or complexity origins, are partly described within the definitions outlined above. The definitions (see 2.3 Definitions of Complexity) identify four main themes:

- Irreducibility or unfragmentable.
- Intricacy and coupling.
- Indescribable or cannot be modelled without complete system replication.
- Level of understanding.

These definition themes provide a starting point for determining the causes of complexity within systems. The causes of these themes are causes of complexity. However, there are a number of other origins within engineering and these too need to be explored. Taking the themes drawn from the definitions as a guide when exploring the origins of complexity is a logical start.

The intricate nature of systems and the intricacy within systems is a common definition of complexity (see 2.3 Definitions of Complexity). Being intricate or containing intricacy is one origin of complexity within systems; it represents being

"Perplexingly entangled or involved; interwinding in a complicated manner"

"The quality or state of being intricate; complexity; complicated or involved condition."

"An instance of this condition; a complication; an entangled or involved state of affairs; a perplexing difficulty."

(Oxford English Dictionary)

These definitions of intricacy and the intricate suggest that the levels of coupling in a system, that is the number of interfaces, the nature of those interfaces, and the effect they have on the system are a cause of complexity in systems. Coupling or dependency is the degree to which one element within a system relies on the other elements within the system. It is the lack of predictability of the relationships or the cohesion of the system elements that makes the system complex; scale is also a factor.

Along with the definitions of complexity having themes, McDermid (2000) identified a number of themes or key factors;

"Scale – the number of elements in the system;

Diversity – the extent to which systems are made up of different elements;

Connectivity – the inter-relationships between the components. "

He further expands on this by stating that scale is not a problem *"if the system structure is regular, it can be assessed analytically – or if the number of elements is sufficiently large it can be assessed statistically"* (McDermid 2000). McDermid goes on to state that scale can "exacerbate problems with other facets", meaning that the combination of scale and other complexity factors are the result of problems. An example of such ordered but large scale systems are Kauffman's NK Networks (1993). The system here is comprised of cells in a grid that are connected with cells adjacent to them; these cells are attached to a clock, and as this cycles the cell state will change (light or dark) depending on the state of the cells around it. Initially cell states are chosen at random, and as the clock cycles self organisation occurs as patterns are produced. Although Kauffman created these networks to emphasise the idea or concept of self-organisation, they are an example of how scale, or the level of interaction, is not necessarily a factor for complexity on its own. The NK Network is easy to model and predict using computer models, the emergent properties and the self-organisation of the system become evident in these models. This is what

McDermid means by a regular structure (repeated in this case) and easily assessed analytically:

"Diversity increases the number of types of element which have to be analysed. The greater the diversity, the more effort has to be spent in understanding the elements individually – and as a composite."

(McDermid 2000)

Diversity, variability or, perhaps a better description, a lack of commonality in systems is most definitely a key factor in system design. The diversity in the system leads to the need for multi-discipline design optimisation.

"Connectivity again increases the difficulty of assessment – the number of pair-wise interactions increases exponentially with scale, for a (potentially) totally interconnected system."

(McDermid 2000)

McDermid refers to connectivity as a complexity contributor; however this is not always the case, and the Boolean NK Networks (Kauffman 1993) shows this to be incorrect. The connectivity of the system contributing to complexity is a result of the level of coupling in the system. Highly coupled systems, in which elements are highly dependent on other elements or a number of elements, will be more difficult to understand than those with a large number of connections but low coupling. Connectivity itself is not a key factor in complexity in systems - coupling and interdependency is.

Although a fairly concise set of key factors, there are three key factors that are missing here;

- The maturity of the technology within the system.
- The level of commonality within interfaces and components.
- The level of coupling between system components.

It could be argued that commonality is, in fact, diversity, but this is not the same as commonality. Systems with large ranges in diversity may have a large degree of commonality, as the majority of the system elements are in fact the same or similar. Diversity on its own as a concept of complexity in systems does not recognise this.

Biggiero (2001) distinguished between 9 different sources of complexity in human systems;

- Pure Logical Complexity
- Relational Complexity
- Pure Gnosiological Complexity
- Evolutionary Complexity
- Semiotic Complexity
- Semantic Complexity
- *"Pure"* Computational Complexity
- Chaotic Complexity
- Self-Organizational complexity

Biggiero mentions the interactive effect studied in psychotherapy and refers to Watzlawick (1967). The key complexity origins within engineering systems are as follows;

- The number of system elements.
- The number of interfaces.
- The nature of the interfaces and coupling that results from them.
- The scale of the system.
- The level of maturity of the technology within the system (effectively a result of the level of understanding).
- The diversity within the system components, the level of multi-disciplinary skill sets required.
- The level of variation and commonality within the system.

These sources or origins of complexity will affect engineering systems and their complexity and are the key origins that are considered within this project. Understanding and managing these sources, how they inter-relate, and their relationships with other complexity characteristics, concepts and measures is also of importance.

2.6 Complexity Concepts and Classifications

Classifications or concepts of complexity are also of high importance when considering complexity within systems. There are a variety of different complexity concepts that can be drawn from the natural world (irreducible complexity, often used as an anti-evolution argument), the mathematical world, the *"edge of chaos"* (Waldrop 1987) , and the engineering world, in terms of concepts of relational complexity.

Senge (1994) classifies complex systems as exhibiting either *"detail"* or *"dynamic complexity"*. *'Detail Complexity'* comprises of systems with essentially hierarchical relationship structures with no lateral relationship links between the system elements, while *'Dynamic Complexity'* describes a hierarchical structure with lateral relationships.

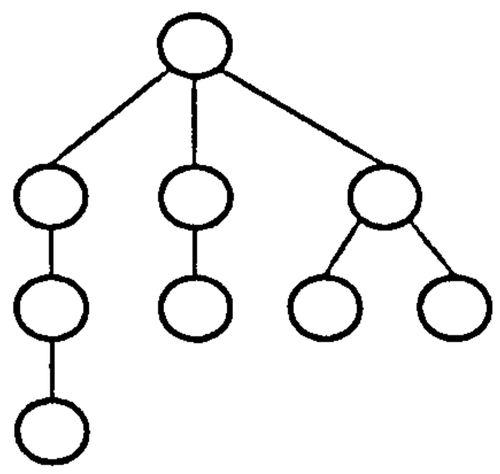


Figure 4 - A system that exhibits detail complexity.

Figure 4 shows a *'Detail Complexity'* system structure. The circles represent the elements within the system (perhaps sub-systems or components), and the links between them represent the interfaces. The hierarchical construction of detail complexity systems means that information flow is restricted. There is no lateral exchange of information within the system apart from at the top level. The number of trees (Irvine 1996) are easily computed, and their associated lengths (depending on the interface nature) are also easily computed.

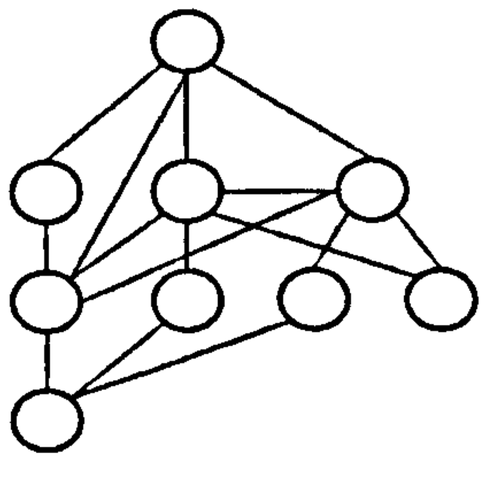


Figure 5 - A system that exhibits dynamic complexity.

Figure 5 shows the *'Dynamic Complexity'* classification of a system, as there both lateral and vertical flows of information across the system interfaces. Subsequently there are a larger number of spanning trees (Irvine 1996) of different lengths that can be calculated for an information transfer from one element to another.

Figure 6 illustrates the differences and potential paths available for systems which exhibit *'Detail Complexity'* and *'Dynamic Complexity'*. The *'Detail Complexity'* system above shows a start and finish point for a slow of information. Assuming the information takes a direct route, there is only one way the information can be transmitted from start to finish, and that is shown as the flow vertically up the hierarchy.

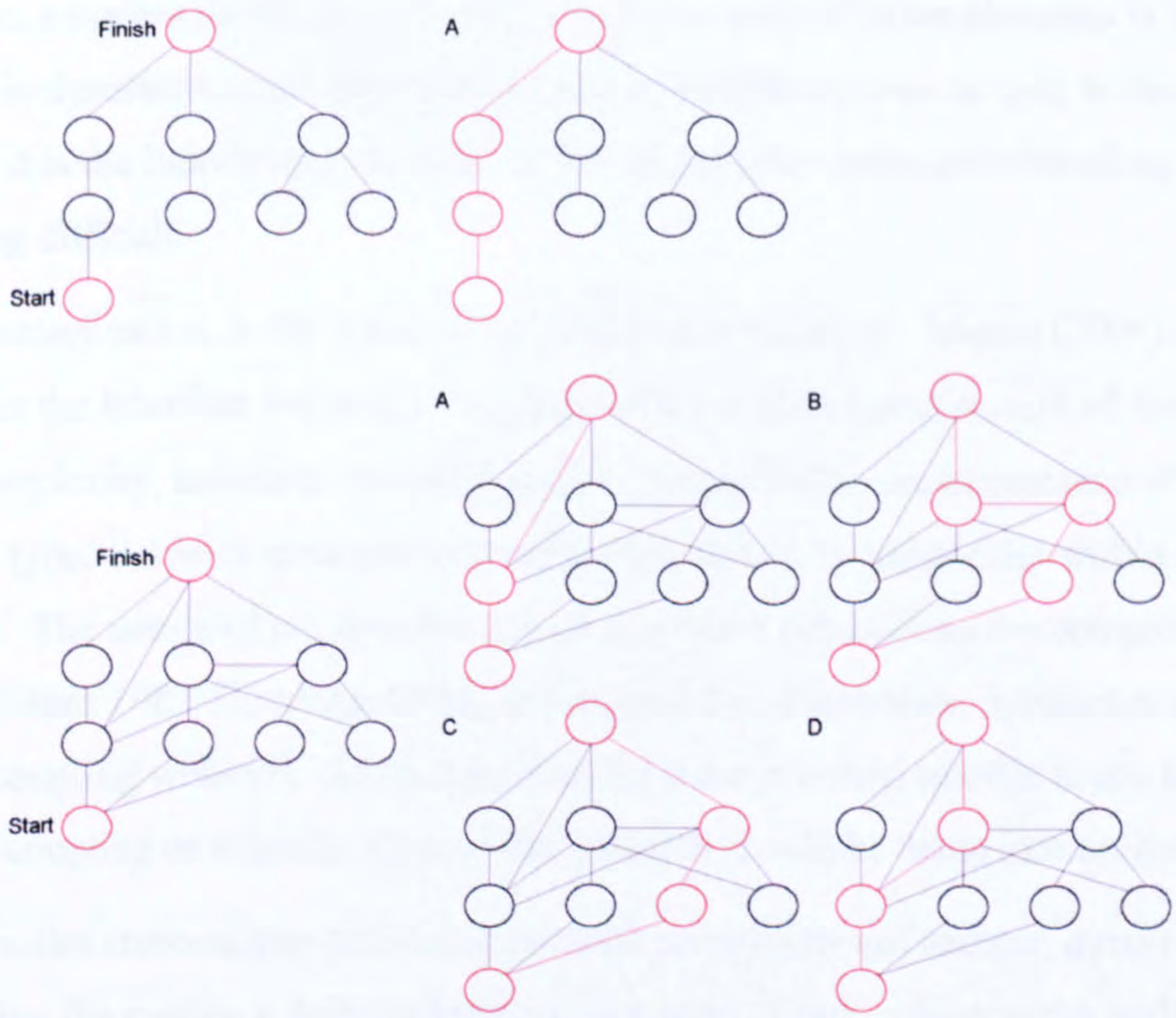


Figure 6 - Path interactions within detail and dynamic complexity.

The lower portion of Figure 6 shows a system that exhibits *'Dynamic Complexity'* (Senge 1994) with links laterally and hierarchically throughout the structure. These links change the nature of the potential information flows. With the same start and finish point for the information flow there are a large number of potential information flows (assuming the flow takes a direct route) that can achieve the transfer, only four examples are shown. In the dynamic complexity system with fixed direction links there is the potential for infinite information flows or loops; within a hierarchical detail dynamic system there is no potential for information loops.

If the links become bi-directional, there are an infinite number of trees that describe the information transfer in both detail and dynamic complexity systems (the information can, in effect, double back on itself an infinite number of times). However the information transfers still follow the rigidity of the hierarchy in the detail complexity system, and contain some order, where as the information transfer in the dynamic complexity system does not have a hierarchical structure, and the transfers as a result are more likely to be erratic.

Dynamic and detail complexity systems link nicely to the definition of complexity, the ability to understand and provide a short description of the system. Hierarchical systems are more easily understood and described than dynamic complexity systems. Of course, a system in which all elements interface with all other elements is the ultimate in dynamic complexity, and in terms of interfaces alone is easy to describe, however it is the functional properties of that system that make understanding and describing difficult.

Further complication, is the effect of the interface vocabulary. Martin (2004) introduces the interface vocabulary as a part of his measurement system of complexity (scale complexity, interface complexity, etc). He highlights the importance of the interface types between elements as contributing factors to complexity within the structure. The nature of the interfaces is critical when considering the complexity of a system (Evans 1987, Edmonds 1999), in terms of the interactions, intricacies and level of coupling within it. Some interfaces have the potential to offer much higher levels of coupling or intricacy than others, and this should be taken into account.

The semantics surrounding detail and dynamic complexity are unclear, dynamic implies that the system is in fact changing, in a state of flux, where as the reality is the system is in fact static, the interfaces are defined, the elements are defined. In both detail and dynamic complexity cases the system is static and subject to no adaptive behaviour - this contradicts the descriptive semantics and can be misleading.

Dynamic as a description goes beyond these concepts of dynamic and detail complexities, truly dynamic systems are capable of adaptive behaviour and self re-organisation (Kauffman 1993), they are able to respond to environmental changes.

The concept of *'Complex Adaptive Systems'* (Heylighen, Dooley, Fryer 1991) or self-organising systems (Kauffman 1993) is a different classification for complexity.

Systems are no longer static in construction as interfaces change in nature or even flip in and out of existence altogether. Organisations of people show the complex adaptive system principle in action, as members are employed into a team, or current members of that organisation move from one team to another (elements being created, or moving), and their interactions with other members of that organisation change, are newly created, or dissolved (interfaces created, destroyed or changing in nature). The overall organisation adapts to the changing environment or re-organises itself. This re-organisation is not necessarily positive, it can in fact be detrimental, as in the case of a system unexpectedly reacting to its environment.

The concept of the *"edge of chaos"* (Waldrop 1987) is a term used to describe complex systems that exist at the boundary of order and complete disorder. It can be thought of as a concept of complexity and not just a description or definition.

2.7 Complexity Measures

A number of numerical and qualitative measures are available that measure complexity within systems, organisations, algorithms and more, some measures even attempt to measure cognitive processes.

Edmonds' thesis (1999) provides a comprehensive study of complexity measures that are available, in particular numerical measures for complexity. There are a vast array of computational and numerical complexity measures that could be applied to systems within this problem domain; it is a case of ascertaining their relevance to the problem domain being explored. The numerical measuring methods explored by Edmonds (1999), Martin and Moody (Martin, Pierre-Alain J. Y. 2004, Moody 1997) who both developed measuring metric and measuring approaches may be useful, but other measuring methods or toolsets of a qualitative nature might also have application within the problem domain.

Some measures have no direct link to systems engineering issues, and as a result, in their current format they are of little direct use. However although the measures themselves are not directly applicable to engineering, the concept of the measures may have an applicability; this was also explored in order to ensure no measures of obvious benefit were ruled out.

2.7.1 Computational Complexities

Computational Complexity (Kimon 2007, Papadimitriou 1994, Du, Ko 2000) is a numerical complexity measure, and measures the required resources to run software or computer system algorithms. The higher the computational complexity of the algorithm, the higher the resource requirement to run that algorithm, attempting to formulate the relationship between algorithm length and the time take for a given system to execute that algorithm.

The application for such a measure within the software field as computer hardware becomes faster and computer resources get cheaper is obvious. Industry can create algorithms within software and then, by using predictions of the technology progression, be able to ascertain whether or not calculation of that algorithm in an allowed time is possible. Processing time becomes particularly important when considering real time systems or safety critical real time systems involving software - here an understanding of computation complexity is of clear benefit.

Algorithmic information complexity (Bennett 1988, Edmonds 1999, Vitanyi, Paul M. B., Ming 2000, Grunwald, Vitanyi 2003) is a measure of the length of the shortest program possible to reproduce a required output string. The more ordered the string, the shorter the program, and hence a reduced algorithmic information complexity, the more random the string, or the closer to the "edge of chaos" (Waldrop 1987), the longer the string must be. Incompressible strings (those outputs with a generation program that is not shorter than output itself, in fact to the point where the generation is a replication of the output string) are indistinguishable from random strings. Closely related to this complexity measurement concept are concepts of arithmetic complexity, which is the minimum number of arithmetic operations to complete a task (very similar to computational complexity, in that the minimum number of operations will govern CPU times, etc.) and also Shannon information theory, using probability to model the output transmitter of a string in order to replicate its output (Shannon 1948).

A measure of the computing resource required to compute a result with respect to the size of the input. As the input increases in size, the computing resource required to compute the result will also increase, this measure, measures that increase. The time

may increase for the computation, or the computing power could increase to compensate for the time increase.

Although the measure is mathematically based, there is an application for this approach within engineering systems. The resource required for the computation could be altered to reflect the resource required to develop a system, and changing that system or adding additional requirements to that system could reflect a change in the level of resource required, in terms of time or other resource (manpower, facilities, etc.). If the resource required increases dramatically with an increase in the number of requirements, the system may be more complex than if the resource were not to increase; either that or the requirement changes were more critical.

Modifying the concept of this measure to assess the level of resource required to develop a system (rather than computing resource to run a programme) is a concept that could be used within engineering.

2.7.2 Information Theory Complexities

Shannon (1948) complexity again focuses on replicating a message string generated from a "transmitter" using a probability based analysis of the message transmitter characteristics (how likely it is to produce an A, or B, how likely it is to produce a C if the previous two letters are A E, etc.) in order to produce an output using a Stochastic Process technique. This is similar to Kolmogorov complexity (Szabo, Li 1997). Peter GrÄunwald and Paul Vitanyi (2003) in their paper Shannon Information and Kolmogorov Complexity distinguish between the two as follows:

"In the Shannon approach we are interested in the minimum expected number of bits to transmit a message from a random source of known characteristics through an error-free channel."

"In Kolmogorov complexity we are interested in the minimum number of bits from which a particular message or file can effectively be reconstructed: the minimum number of bits that suffice to store the file in reproducible format. This is the basic question of the ultimate compression of given individual files."

Peter GrÄunwald and Paul Vitanyi (2003) then further expand on this and explain the difference and its importance.

"A little reflection reveals that this is a great difference [between Shannon Information and Kolmogorov Complexity]*: for every source emitting but two messages the Shannon information (entropy) is at most 1 bit, but we can choose both messages concerned of arbitrarily high Kolmogorov complexity."*

Rissanens Minimum Description Length (MDL) is another information description complexity model (Vitanyi, Paul M. B., Ming 2000, Barron, Rissanen et al. 1998, Grünwald, Myung et al. 2005, Complex Systems Computation Group), similar to those created by Kolmogorov and Shannon. The model describes the MDL of a system, and the length of this description forms the basis for the complexity. The larger the description the more information the system contains and the higher the complexity.

In both Shannon and Kolmogorov (Shannon 1948, Grunwald, Vitanyi 2003) applications for industry are not obvious, but there are potential technical applications in data transmission. However, when trying to use this measure with respect to a development programme the scope is limited, as one would expect since they are not developed for the systems engineering domain. MDL however could be related to the requirements that describe a system, COSYSMO (Valerdi, Boehm et al. 2003) and a number of other basic complexity measures within development programmes already use the number of requirements (the description of the system, the number of requirements being related to the description length) in a system specification to estimate complexity qualitatively. COSYSMO of course expands on this, with difficulty factors for specific requirements, as some requirements may require more effort for compliance than others. This, of course, means that the MDL (if it is assumed the requirement set is compressed to the maximum while maintaining its integrity) although smaller, may in fact be unable to take into account the difficulty in meeting each requirement. The depth of the requirement is in fact the issue - what the requirement actually means in terms of effort and resource, complexity measurement that analyses the true implications of the requirement set along with the compressibility within it would be more appropriate to systems engineering practice.

Bennett's 'logical depth' (Bennett 1988, Edmonds 1999) is defined as the running-time to generate the object (string) in question by a near-incompressible program. The measure gives value to information based on the time taken to calculate or produce the information and its usefulness; the time taken to calculate aircraft take off

data (weights, runway rolls length, reference speeds, etc.) means the data is more valuable than the equations used to calculate them. Edmonds quotes Bennett saying,

"Logically deep objects... contain internal evidence of having been the result of a long computation or slow-to-simulate dynamically process and could not plausibly have originated otherwise." (Bennett 1988)

In terms of output strings from programs, a random string has low logical depth, if it is incompressible and therefore the reproduction requires a simple copy of the string. A very simple string also has low logical depth, as the program required to reproduce it is also simple. There are distinct links between logical depth and the concept of sophistication (Fortnow 2003), which *"is the size of the projectible part of the string's minimal description and formalizes the amount of planning which went into the construction of the string."* Whereas *"Depth,"* defined by Bennett (1988), is the amount of time required for the string to be generated from its minimal description and formalizes its 'evolvedness'.. Both are a variation of the same theme, and like other concepts of information theory (Bennett 1988, Shannon 1948, Barron, Rissanen et al. 1998, Complex Systems Computation Group) the complexity hinges around the length of the shortest programme to model strings, transmitters and receivers in the minimal way.

It would seem that Bennett's logical depth (Bennett 1988, Bennett 1990, Edmonds 1999) contradicts some descriptive complexity measures, in that random strings would in fact have a low logical depth due to the simplicity of the description of that string, but a high complexity in terms of Shannon and Kolmogorov information theory in that the directly replicate the string the entire string must be stored and there is no compressions possible.

2.7.3 Information Flow Complexities

The number of spanning trees relates to the pathways within a system of elements and interfaces. The concept of spanning trees is methodology for understanding system interfacing and the intricacy and coupling of these interfaces. The more "complex" the interface between two system nodes or elements, the longer the "length" of that interface. From these interfaces between nodes, and their associated lengths, paths through the system for information can be measured and compared to gain insight into how the system will operate.

A system with 4 nodes that are all connected, has a total of 16 different pathways for information flow (Irvine 1996). The applicability to developing systems is again related to the interfaces between system components, and the potential flows of information. Generally it is accepted that the more information flow pathways possible, the higher the complexity. However, there are other factors to consider when dealing with interface complexity and this measure alone would not be a sufficient measure.

Spanning trees as a concept relate to the idea of dynamic complexity within systems (Senge 1994), as the number of spanning trees in a system represent the different directions of information flow within a system that are possible, there is potential here to measure (Irvine 1996) complexity in systems with changing structures, or comparing the complexity of the systems in interfacing and information flow terms.

2.7.4 Length of Proof

Proof lengths for mathematical theorem can be used as a measure of complexity. However, mathematical proofs can be short and yet be complex, and long carefully written proofs can be simple depending on how they are constructed (the use of complex axioms and the level of hierarchical information gain (Edmonds 1999), perhaps in an explanatory fashion), possibly even unnecessary lengthen added to the proof to make it easier to follow, giving rise to *"needless complexity"* (Edmonds 1999). If there are two theories or solutions for the same problem, both with equal supporting experimental data, the simpler theory using the Goodman's measure should be used.

Similarly, Goodman (1966) developed a categorisation of extra-logical predicates, based on expressiveness. A general predicate is deemed more complex than a symmetric one, as it includes the latter as an example. Similar to hidden complexity, as a general predicate may contain complex proof, as Edmonds (1999) states, *"the complexity of a complex statement is merely the sum of the complexities of its component predicate, regardless of the structure of the statement."* Kemeny's measure (Kemeny 1953) is similar and also a reformulation of the idea was made by Richmond (1996).

There is a definite applicability for the concepts surrounding Goodman's and Length of Proof as a concept for engineering problems. Along with concepts of

decomposition, reducibility and unnecessary complexity within developing systems they are an analogy for engineering system complexities, in particular within the requirements domain.

2.7.5 Reducibility Complexity Concepts

Holists use the term complexity when referring to systems where reduction of a system is not possible, or at least not possible by current practices. Irreducibility and elements of information theory are linked as information theory attempts to define how reducible information is in terms of "logical depth" (Bennett 1990) or complexity of the algorithmic modelling of transmitters or string outputs (Shannon, Kolmogorov) required to accurately replicate real system transmitter or string output behaviour. Extending the principles of information theory to systems engineering, an irreducible system is like the random sequence example; no model of the system behaviour can be achieved without modelling the system or its output completely (essentially replicating it). The implication of an irreducible system to designed or developed systems is that the system cannot be optimised any further, the complexity that exists is purely intrinsic to the system (the system in its current state cannot be reduced in any manner software, hardware, human interaction, etc.). Reducibility in design implies an over complication in design, complexity induced by the development cycle and not intrinsic to the system. The level of reducibility within a system may be an indication of the induced complexity, some of which may be unnecessary or undesirable complexity.

If in order to measure reducibility or unnecessary complexity, work must be conducted to uncover the reductions that can be achieved, and then reduce the system with these findings if economically viable. Therefore a reducibility measure, measures the level of known unnecessary complexity that exists within a system where optimisation is uneconomical. Reducibility is not just the level of reduction that can be achieved within the system as a whole, but also the possibility of breaking down a system into simpler component parts that overall exhibit the same behaviour, therefore reducing the system to more manageable chunks.

The concept of reducibility can be applied to system requirements. A standard requirements document or specification will contain both intrinsic and induced complexities. Specifying interface types such as Ethernet as interfaces between

components is an increase in complexity. The minimum size of a requirements document (that which Edmonds (1999) describes as a *"perfect compression"*) will include only intrinsic complexity. A cross comparison of this with a requirements document containing induced complexity as well as intrinsic (specifying interface types etc.) will give the level of induced complexity in the system, providing the basis which can be used to reduce unnecessary complexity in design (some induced complexity is necessary to reduce effort levels).

2.7.6 Interpretive Complexity

Löfgren's (1974) description of interpretive complexity of systems offers another method a measurement. Taken from a biological and psychological context, Lofgren's descriptive and interpretive complexity is described by Edmonds (1999) as *"the interpretation process is the translation from the description to the system and the descriptive process is the other way around. For example the "description" could be the genotype and the "system" the phenotype."* From a systems engineering point of view this relationship could be the system itself and the requirements specification. The interpretation is the translation of the requirements base to the system and the description, the translation of the physical or conceptual system into requirements. Two particular situations are applicable here, one the use of legacy equipment in which the descriptive complexity is an issue, interpretive complexity is associated with the generation of new products, perhaps from capability to requirement specification, and then once again from requirement specification to product.

When considering descriptions of complete systems, there may be a number of different solutions or sets of parameters or logical statements that model the behaviour of that system accurately. The number of inequivalent descriptions (Edmonds 1999) or models of a system that can be produced can be extended and used as a basis for a complexity measurement. The system requirements in particular relate nicely to the inequivalent description concept being related to the complexity of that system. This in particular relates to capability acquisition, within which a capability is described using a requirement set, and a system built to the requirement set. The complexity of the system may be related to the number of potential inequivalent requirement descriptions possible for the system, and then additionally the number of inequivalent systems that meet that requirement specification.

The smaller and less coupled and inter-related the requirement set is, the less the likelihood of creating many irreconcilable system designs, the larger the requirement set and the higher level of coupling and inter-relation the higher the likelihood of irreconcilable system designs.

2.7.7 Size and Complexity

Size in engineering systems, in some cases has links to the complexity of that system, and can lead to valuable insights. Generally it is accepted that the larger the product the more difficult the design process will be, but the complexity comes with coupling and intricacy within that product. In some cases the size of the requirements documents, or the technical requirements specifications have been used as measures of complexity (Valerdi, Boehm et al. 2003), these of course are a convenient label, but the application of size in complexity measurement terms must be supported by other measures in order to be meaningful.

Associating the number of parts and interfaces within a system with a measure of systems complexity is a method of assessing size. The relationship is not direct, despite a high number of interfaces, elements or parts within a system the complexity is not always high; products with many parts that are all identical, and do not interact would not be called complex simply due to the size, a panel of 10,000 light bulbs wired in series would be considered large, but not complex.

Edmonds (1999) provides a good example of minimum size, two lists of facts, one list 1001 inter-related facts and the other 1001 unrelated facts. The inter-related facts of the first list would suggest that there is some redundancy due to the relationships between the facts themselves, perhaps there are simplifications of some of those relationships and therefore potential room for compression. This would give this list a lower complexity value, as the information can be compressed in a more succinct language. The unrelated facts however are incompressible, there are no inter-relations and thus more information is held by this list, this would give this list a higher complexity value. This of course heavily relates to the concepts of irreducibility (Dembski 2004), logical depth (Bennett 1988) and aspects of information theory.

A similar complexity measurement metric to that of reducibility and minimum size, Network complexity (Shih, Janet 2007) measures the minimum number of logical gates that are required to implement a logical function. The relationship between the

network complexity measure and system design in the engineering domain is clear for logical network systems, and the concept that can be applied to system design is that similar to reducibility.

2.7.8 The Nature of System Variables and Complexity

Weaver and Shannon classified problems initially as simple and complex (Shannon 1948), now further classification comes with organised and disorganised complexity. This directly relates to the number of variables, a system with a large number of variables is difficult to model due to the level of randomness, such a system is not amenable to statistical analysis and therefore categorised as disorganised complexity. A system with fewer variables and that can have effective statistical analysis conducted upon it is known for exhibiting organised complexity.

The number of variables (Edmonds 1999) within a statement or algorithm is another measure of complexity. The relationship between the number of variables and the system complexity assumes more variables will increase system complexity, or algorithm complexity. This alone is insufficient as a complexity measure - there is not necessarily a correlation between system complexity and the number of variables. The nature of the variables is important, and the interdependencies between those variables also important, a set of independent variables with a simple algorithmic relationship would be less complex than a mix of independent and dependent variables with intricacies within the algorithmic relationship. They affect the number of variables has is dependent on the structure and element interdependencies of the system within which they reside.

Like variables that are required to calculate the output of a formula or algorithm, a number of dimensions are required to describe an object in real or conceptual space. The higher the number of dimensions required to accurately describe an object, the more potential for complexity in that system. Dimensional views within engineering may consist of discipline variation, trade off variables, design variable relationships (such as aerodynamic properties of wings (Kesseler, Vankan 2006) using multidisciplinary design optimisation techniques), the design itself (support, development, design, operation, decommission) and resource allocation (budget, personnel, facilities). Application of complexity measures within these dimensions to

quantify them and a definition of their interactions across a development programme or system gives scope for valuable measurement.

2.7.9 Variety or Lack of Commonality and Complexity

Variety (Edmonds 1999) or commonality within systems has the potential to increase system complexity and to influence the manufacture of products (Kim, Chhajed 2000, Thevenot, Simpson 2007, Alizon, Shooter et al. 2007, Jans, Degraeve et al. 2008). That variety can be the result of different disciplines in design, variety in capability or functional requirements, variety in suppliers and more. Large scale systems with high levels of variety are more likely to exhibit a complexity that makes design and predictability difficulty; thus variety is associated with the complexity of the system. The variety within the behaviour can be measured by the counting of types, spread of numerical values or presence of sudden changes.

Measuring the variety of behaviour within a system, or the diversity of the behaviour can be considered in functionality terms; the variety of system functionality and diversity of operations. Variety within the system environment is also critical when designing systems - a diverse and highly volatile environment that exhibits a large variety in behaviour requires a more robust adaptive system.

2.7.10 System Information Hierarchy or Scale

Changing the hierarchy level or scale at which a system is analysed will change the level of information available in terms of interfaces, components, interdependencies and coupling. The change of depth the system is measured at will change the apparent complexity of the system; higher functional levels will give lower complexities than detailed component levels. Martin (2004) uses a decomposition method to generate complexity measures, but those measures are generated using a selected hierarchy and scale of decomposition. It is possible to further decompose the system Martin (2004) has measured, using the same measurement technique and get a different and perhaps more complex result. This decomposition could include processor architectures, software architectures and even go down to incredible detail modelling the electromagnetic properties of cabling.

Information hierarchy complexity within algorithms, measures the information level change between high level algorithms against the lower level algorithm with detailed

descriptions of the axioms within it. These axioms may be more complex than the overall higher level algorithm, as they are developed from complex proofs, as a result the information gain would be large. This nicely couples with the number of axioms (Edmonds 1999) within an algorithm - the more axioms the more complex the potential information hidden within the hierarchy.

2.7.11 Connectivity, Intricacy and Coupling Complexity Measures

Connectivity and the intricacies of the interactions between components is the focus for a number of complexity measures that directly relate to engineered systems. Connectivity, quite simply, is the measure of the number of interconnections between system elements - the more connections, potentially the more intricate and highly coupled the system is and the more complex. Failure mode analysis is also a factor as failures within systems with higher levels of internal interfaces and intricacies are more difficult to analyse.

The applications for connectivity are obvious to the engineering world, but a direct correlation between system complexity and the number of internal interfaces is not always applicable. Systems may consist of a large number of connections, and these connections may consist of different levels of information transfer or intrinsic complexity; simple Boolean interfaces, or large volume data transfers.

Connectivity can be measured using the number of interfaces, the information that flows through those interfaces, the criticality of that information and vocabulary (Martin, Pierre-Alain J. Y. 2004) of that information, these principles were discovered by Martin (2004) and used to created measures of complexity that were more suitable.

There are other applications for connectivity within other domains (biology, chemistry, ecology), but within the systems engineering domain the key interests are within the software and hardware interfaces that contribute to increased difficulty in development. Martin's (2004) measures of interface complexity (the measure of complexity between the system and its environment) and the link element of internal complexity (the complexity of the internal interfaces) provide a good grounding for the construction of a connectivity measure appropriate for the engineering domain.

The single figure output for interface, external and internal complexity, however, do not provide any insight into the likelihood of emergent properties, the level of

difficulty, feasibility, potential problem areas and the amount of intrinsic or induced complexity within the developed product.

Cyclomatic number, sometimes called program complexity, or McCabe's complexity (McCabe 1976), is the number of independent loops within a graph. The measure essentially shows the complexity of the flow control through some program code, V(g) (cyclomatic number) = Edges – Nodes + Components or V(g) = e – n + 2. In a program with just sequential control V(g) = 1, the greater the number the more execution paths there is. This relates to the Senge (1994) principles of detail and dynamic complexity, in programs consisting of sequential only flow, the system exhibits primitive detail complexity (V(g) = 1), as the program flow complexity increases the system exhibits more detail complexity (essentially a hierarchical system with few other paths) however as the cyclomatic number increases the program flow structure enters the dynamic complexity state.

How can this be applied to developed systems? Obviously there is a direct application of this measure and others (Wilkie, Hylands 1998) similar to it in the software domain, but physical systems can also be modelled in the same way. If the system can be modelled in terms of controls then the cyclomatic number can be replicated for physical hardware as well as software. The measure in the engineering domain is limited due to the shear level of effort required to model large systems with detailed mechanical, electrical and software components, and there is no way non-controlling or indefinable interfaces (such as power, and human interaction) can be modelled and therefore these key elements of systems design would not be considered despite their being key to the overall complexity.

2.7.12 System Complexity Measures

Martin (2004) developed a set of measures for complexity within engineered systems. These complexity measures are called Internal, Interface, External and Socio-Political Complexity.

Internal complexity is a result of what Martin calls Scale and Link complexity, which are functions of the number of elements, the links between, and the level of functionality of the links (referred to as the *'vocabulary'*).

Interface complexity uses the equation described earlier (heading) in which the interface complexity is a function of the information required to describe it along with the probability of meeting the functionality requirements to produce it.

External complexity is a function of system failures, the frequency of these failures and the magnitude or criticality of the worst failure possible within the system. It is defined as *"the ration of the risk associated with all failures of higher than average magnitude over the risk associated with all failures of lower than average magnitude"* (Martin, Pierre-Alain J. Y. 2004).

Socio-Political complexity is linked to risk perception within the market, but it is difficult to place a value on this as complexity measure that can be used in the engineering cycle.

2.7.13 Randomness and Unpredictability and Entropy Complexity Measures

Entropy is the measure of disorder within a system; this could be within information or energy (Baranger 2001, Lemay 1999). In the case of energy, it is energy within a system that cannot be used for work, measuring aspects of gasses such as their state of randomness. In engineering information surrounding the system has a level of order; the diversity and order of the requirement set, skills required and technologies used.

When comparing or combining two systems or two components, entropy (disorder) in each separate system may reduce (tend to order) when integrated with another system. If these two systems are now joined together and a combined entropy measurement is produced, this is the mutual information. This is the extent of the randomness when considered together rather than as a separate entity.

$$\textit{Mutual Information of A, B} = \textit{Entropy A} + \textit{Entropy B} - \textit{Combined Entropy A, B}$$

Mutual information (Edmonds 1999, Moon) measures the relationship between the individual component unpredictability and the combination of the two. In systems this principle is very prudent, it may be that two elements of a designed system when operating or being developed separately require large information investments due to

their high level of randomness, but when considered together rather than apart the information level is less than that required to do both systems separately.

In systems engineering terms, this principle will affect the detailing of sub-system boundaries, if these boundaries are chosen poorly, the resulting resource required to develop the two sub-systems may be higher than that required to develop them as a single element.

Perhaps related to mutual information, understanding the boundaries between system components to reduce the entropy, the difficulty associated with a system being deconstructed into its component parts, or sub-systems can also be considered as a complexity measure, the ease of decomposition. In the case of algorithms and mathematical proofs, Edmonds (1999) suggests that although the proof may be a simple axiom, the meaning of that proof could be complex. An example is the meaning of words within the English language; words may have definitions that go beyond the syntax of the phrase within which they reside, making the phrase more complex and open to interpretation. This is similar to that of hierarchical information scaling and interpretive complexity (Löfgren 1974) , possessing characteristics of both measurement concepts.

2.7.14 The Edge of Chaos and Attractors

A mathematical measure relating to chaotic processes, small changes in state at an early stage cause large changes in state at a later stage making such processes impossible to predict beyond a certain time frame. Despite this chaotic appearance patterns do emerge out of the chaos where systems tend towards particular states as a result of the initial conditions (Waldrop 1987, Lancaster 2007), these are known as 'attractors' (see section 2.2.5), attractors can differ in complexity, and a measure of the dimensions of the attractor for the system is a measure of the complexity of that system.

There is no obvious mapping of this complexity measure to the engineering domain. Even the methodology is unsuitable for deriving measures that can be of use.

The likelihood of complex systems emerging at random is low, and often complexity is associated with low probability. On the other hand, if complexity is fused with entropy or other informal measures then the probability of a complex system emerging is high. This has led to many developing measures that fit complexity

between order and disorder. On the contrary, the philosophy of *"simplicity"* has led to identifying higher priori probabilities supporting the truth of a theory which has a low complexity (Edmonds 1999).

The basic principle surrounding the idea that complexity arises out of low probabilities may have an abstract significance for product systems, but as for measuring complexity using this concept the application is limited and provides little benefit.

2.7.15 Psychological Complexity

Cognitive complexity (Bieri 1955, Andrews, Halford 2002, Green 2004) is a much discussed topic within the psychological world. An example of an element of cognitive complexity is an individual's ability to recognise positive and negative traits without being limited by their own bias. The dimension of this mental model the individual creates is then referred to as their cognitive complexity. By assigning their friends positive attributes and their enemies negative attributes the individual is considered to be cognitively simple, by assigning both adversaries and friends positive and negative attributes an individual is considered to be cognitively complex, possessing a two dimensional model of others.

The qualitative measure is very time consuming to calculate, however it provides little a limited benefit for industry in terms of developing systems. Possibilities for incorporation into an understanding of system complexity may arise from understanding behaviour within human organisations that form part of the developed system, or potentially an understanding of the organisation developing such a system. Mindset has proven to be a contributing factor within system complexity, it is important to consider in developing systems and the case studies will illustrate how mindset can change the outcome of programmes.

2.7.16 System Simplicity

Simplicity (Edmonds 1999, Richmond 1996) is a principle that is used when faced with two theories that are equally supported by evidence; in such a case it is natural to use the simpler theory or look for a simpler theory, and this has been termed "simplicity". All purely logical theories are equally and ultimately certain and hence "simple" - simplicity does not help us distinguish between them, and it was not meant

to. Connected theories are Goodman's and Remeny's (Goodman 1966, Richmond 1996).

How can simplicity be tailored for the engineering domain? I am not sure the concept has a place within engineering design. In industry the simplest solution may not be the most cost effective design, and so the search for simplicity may be uneconomical.

2.8 Complexity Coping Methods and Approaches

A number of different approaches, management techniques and coping methods have been applied to engineering, product development and other areas in an attempt to improve processes (Haskins 2006, Heylighen 1991, Gregory 2000, Barclay, Dann 2000, Marshall 1997, Lees, Branki et al. 1995, Perona, Miragliotta 2004, Shaw, Taxén 2003, Vakil, Hansman R.J. 2002, Stevens 1998, Dunlop, Evans et al. 1997, Gottinger 1983) . These processes, frameworks, approaches or coping mechanisms aim to improve the ability of industry to manage, control, develop, improve, understand, or produce complex systems with reduced risk.

The following sections detail coping methods, or approaches that have been undertaken or potentially could be undertaken to deal or cope with complexity issues within the changing engineering environment.

Coping or approach methods to complexity overall attempt to:

1. Reduce the complexity of the organisation that develops, operates or supports the systems that are developed, with improved modularity in design and better sub-system decompositions.
2. Reduce the intrinsic complexity of a system, by reducing functionality, interfaces and their intricacies and elements.
3. Reduce the induced complexity of the system brought about by engineering approaches and system decompositions.
4. Better handle or process the intrinsic complexity of the system.
5. Produce improved optimised designs with reduced unnecessary complexity.

Coping methods for complexity essentially attempt to reduce induced complexity, or manage intrinsic complexity. The management methods may focus on personnel, resources, time scales, budget, and intricacies of interfaces or system decomposition.

If induced complexity is too high and affects the cost or timescales of that programme, coping methods are employed to reduce that effect.

In some cases it may be better not to employ a solution or coping mechanism to a complexity issue, as it conflicts with other crucial programme constraints. This has been apparent in some case studies (see chapter 4) explored within this research, where problem issue solutions were in fact not to implement a solution at all. None the less it is important to explore the coping methods or approaches that can be used or implemented when systems engineering large integrated designs.

2.8.1 System Scope

Reducing the system scope and size may help to reduce the induced complexity of the system, and also reduce intrinsic complexity. Keeping interfaces between sub-systems simple and well defined, and limiting the overall number of states the system can be in at any instance, not the sub-systems. A very simple but effective coping mechanism, reducing intrinsic complexity by reducing the system scope, however trading off intrinsic complexity against desired functionality is not always an option and this limits the viability of such an approach.

What ideally is required is a coping mechanism, or approach that reduces the induced complexity by design, and does not affect the intrinsic complexity of the system and its functionality, which is key to product success and acceptance.

2.8.2 Managing Modularity and N2

Matching modularity in the design with supply chains and organisational structures is an approach to reducing induced complexity of the system. Sub-systems can be more easily understood if that sub-system functionality is self contained, and not subject to other system areas. In networked systems this is difficult, but producing a modularity based design process may encourage it. Marshall of Loughborough University (Marshall 1997) has produced a workbook detailing the modularity design methodology in engineering, which shows how to approach design in this fashion.

Some approaches match design modularity with supply chains to sub-system decomposition, along with development teams with sub-system decomposition. Essentially, this involves reducing the complexity of the organisation that develops, supports and operates the system by reducing unnecessary interfaces within the

development and supply chains. Modularity uses the mutual information (see section 2.7.13) notion, that changing the decomposition of systems can reduce complexity or create order. Sub-systems with cross functionality split over sub-system boundaries are difficult to design and develop concurrently, whilst also making the system more difficult to integrate with the additional coupling of cross sub-system functionality.

Suppliers might be mapped to functionalities rather than specific sub-systems, in this case the supplier decomposition is inconsistent with that of the design teams and these inconsistencies have the potential to cause problems in the development phases.

There are tools that allow intelligent decomposition or structuring of system design projects using matrices such as N2 (Eppinger, Whitney et al. 1994), or various graphical approaches to functional modelling.

N2 is tool that enables the relationships between requirements, functions and architectures to be explored and understood in the systems context. Figure 7 shows the inter-relationships between requirements, and the relationships between requirements and allocated functions.

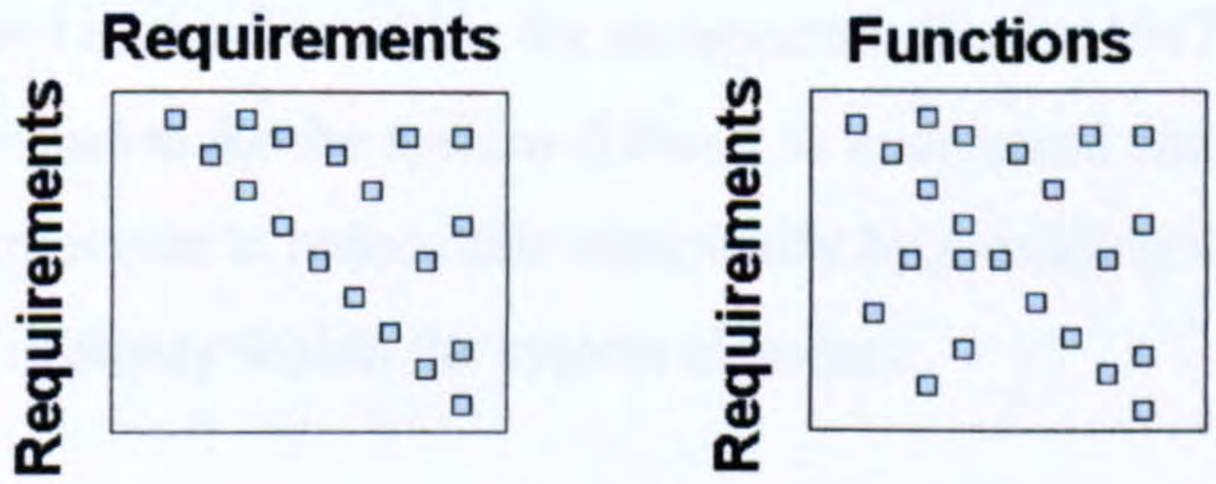


Figure 7 - DSM requirements and function relationship diagrams.

N2 provides an appreciation of the interdependency between requirements and functionality, a process that provides DSM models (Eppinger, Whitney et al. 1994) with data to provide detailed interdependency diagrams which can later be optimised.

The level of requirement interdependency can potentially be an indicator of the complexity in the specification of a system, the more interdependency, coupling and internal loops, the higher the complexity in the specification. The number of functions allocated to a requirement set may be an indication of the complexity of those requirements, the more complex the requirement, the more functionality allocated to it.

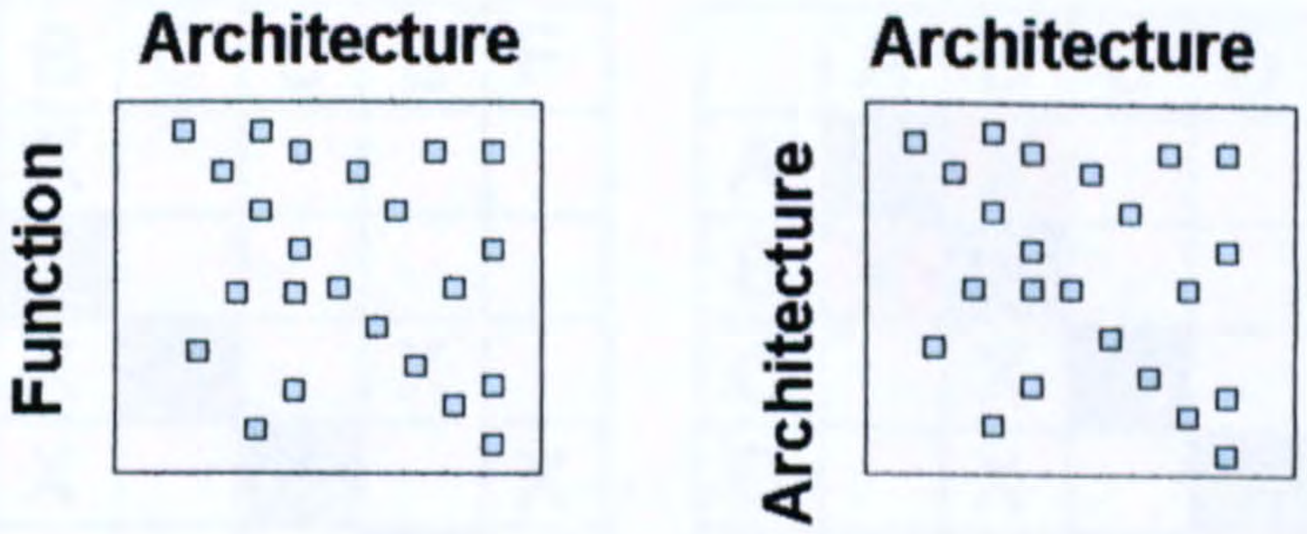


Figure 8 - N2 functionality and architecture relationship diagrams.

Figure 8 shows the N2 allocation of function to architecture elements and the interactions between architecture elements. Induced complexity can be the result of the architecture development process creating unnecessary interdependencies between elements. The use of Commercial Off The Shelf (COTS) or Military Off The Shelf (MOTS) products that have allocated function inbuilt, restricting the allocation of function within the system architecture elements and therefore influencing the complexity of the system.

Interactions and interfaces between architecture elements is partly intrinsic and partly a result of design. The architecture implementation will affect the system complexity, large interactions and intricacies within the architecture (Evans 1987) may result in emergent properties and make the system difficult to understand and service. Architecture principles aim to reduce this complexity by providing a framework to reduce the level of intricacy within the system elements.

2.8.3 Design Structure Matrices

DSM techniques (Eppinger, Whitney et al. 1994, Carrascosa, Eppinger et al. 1998, Eppinger 1991) can be used to dissect problems that occur within programmes using a matrix of programme elements and identifying the interdependencies and influences between them. Figure 9 shows the construction of a DSM and then the optimised example showing the dependencies or influences are within a closed loop or circuit.

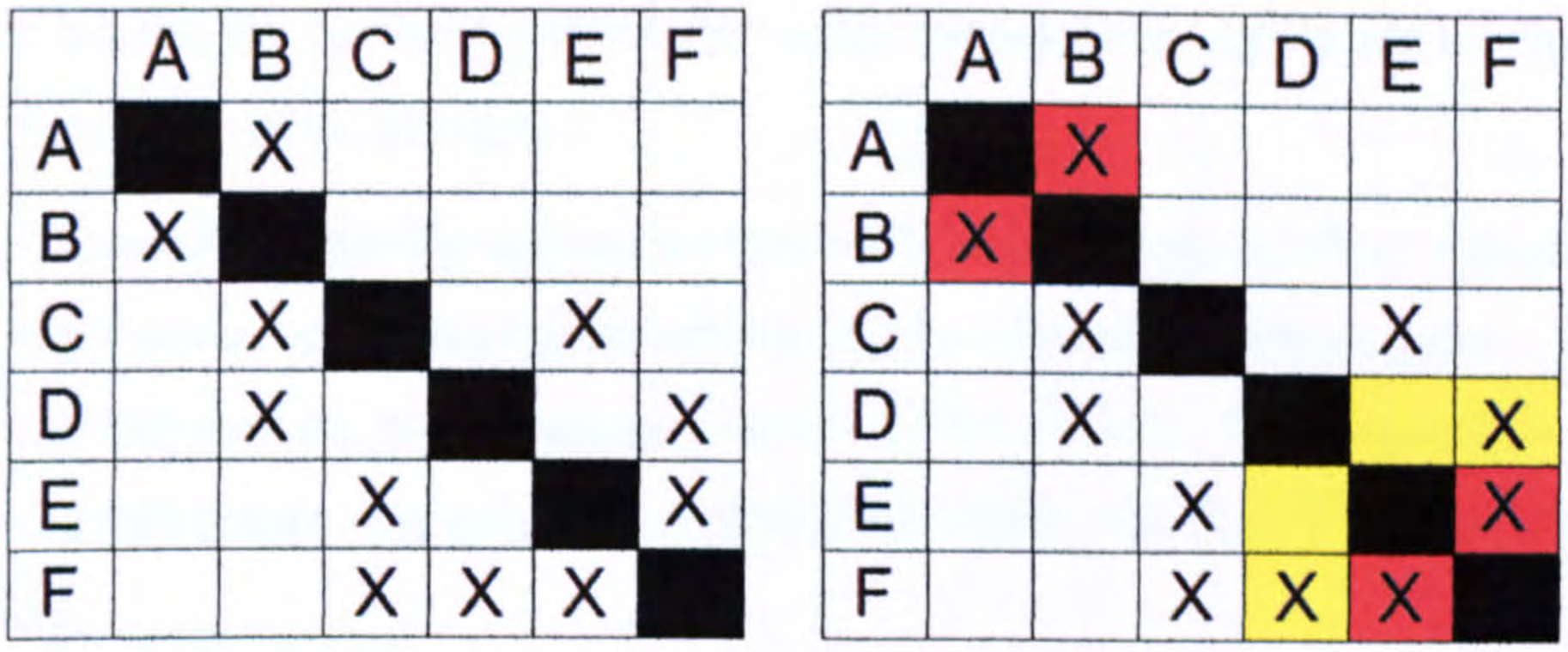


Figure 9 - DSM, developed and optimised examples.

Closed circuits or loops can be identified within the interdependency and influence matrices after re-organisation of these matrices. In order to provide a more efficient development process, or remove the problem issues by developing a solution, these loops or closed circuits must be addressed or removed. The solution may be a change of requirement or re-allocation of function within the architecture. Once this solution is developed the implementation of this solution can be tested within the DSM model, by modifying the matrices to reflect the change. This process will enable a preview of the second order problems that will occur as a result of the change to the first dependency matrix.

DSM can be applied to complexity problem issues by breaking these issues into their component parts. These component parts can then be dissected in the same manner explained above, a solution chosen as a result of the optimisation, and the second order effects of such a solution understood before it is implemented. The process can then be repeated for second order problems.

2.8.4 Difference Reduction Design Techniques

Difference reduction (Heylighen 1991), rather than focusing on the entire system, focuses on differences between the current state and the required state, it is a process of optimisation in design. If there are a potential 100 different simple actions available, which are to be combined into a sequence of 10, this gives a possible 100^10 (10000000000) different potential overall actions that can influence the situation. Incorporating a focus on the differences between the current situation A and the desired situation B, reduces this number of possibilities. There may be 10 objects that are different from the initial situation to the ideal situation, and there will be an action that can be undertaken to induce the desired change, this gives us 100

potential actions for 10 objects, 100 x 10 = 1000, a much lower figure and a reduction in the complexity of the problem.

Further knowledge about the actions, and knowing how the actions affect objects which will further reduce this figure; actions will be selected for reducing the differences between the two situations A and B for the objects. The actions that reduce the differences may only be 2 or 3 for each object, thus 3 x 10 = 30, a further reduction.

In operating in this way, the objects chosen must be as independent as possible so as to not influence any other differences between situation A and B, other than those desired (Heylighen 1991).

2.8.5 Iteration Reduction

Iteration reduction, like difference reduction, is another approach to reducing the workload of a development programme in an attempt to avoid over complication and difficult traceability (Smith, Eppinger 1995). More design iterations, or slower design iterations, slow system development, and it is recommended that a development capability should speed up the iterations within the design space or reduce the number.

Faster iterations can be achieved in a number of ways. These could involve the introduction of process improvements like the following:

- *Computer-aided design systems which accelerate some of the individual design tasks*
- *Engineering analysis tools such as simulation techniques which reduce the need for time-consuming prototype/test cycles.*
- *Information systems involving database management and networking software which facilitate rapid exchange of technical information among individuals on the design team.*
- *Removing extraneous activities from the iterative process.*

Fewer iterations could be achieved by, for example:

- *Improved coordination of individuals whose work depends on one another*

- *Co-location of team members responsible for tightly coupled activities, allowing faster and more frequent information exchanges and faster resolution of conflicting issues*
- *Minimization of team size, which allows a core set of individuals to work more efficiently.*
- *Proper specification of interfaces, allowing for reduced need for interactions between individuals and teams within the development process.*
- *Use of engineering models capable of predicting performance along multiple dimensions, eliminating the need for separate analyses.*

(Smith, Eppinger 1995)

2.8.6 The Systems Engineering Lifecycle Management (LCM)

The Systems Engineering lifecycle (Haskins 2006) is one of the fundamental tools in modern systems engineering design - it forms the basis of all development programmes. The systems engineering lifecycle is a management guide that manages the development process from beginning to end and attempts to control the process in a manner that reduces effort levels by organising more efficiently. In doing so other management practices and tools can be employed; concurrent design management (Eppinger 1991) is an approach that can be implemented within programmes. Better management of concurrent engineering activities is paramount when large complex systems must be designed and built in short time frames, MoDAF and DoDAF are architectural approaches to systems engineering (UK Ministry of Defence 2005a, DoD Architecture Framework Working Group 2003). They are guidelines to build, evaluate and integrate system architectures and form part of a toolset for system development for the UK and US defence industries.

UML is an area currently under development for systems engineering (Weilkiens 2008, Bock 2005). SysML, a UML based language specifically developed for systems engineering is a new approach to product development, and architecture design that can be implemented within programmes. Combining these UML based languages with modelling tools and the MoDAF, DoDAF frameworks can prove to be very effective when dealing with product development and the complexities within it.

Some adaptation is still required however, as UML is primarily a software development tool.

2.9 Conclusion

A lot of the complexity literature that exists is not specifically geared towards the systems engineering domain. A large volume of literature associated with complexity, complexity origins, measures, definitions, concepts and so forth, stem from mathematical research into chaos theory, information science and algorithmic or computational complexities. The systems engineering concept seems to be missing from the majority of the literature within the complexity domain, but duely considered within the engineering domain. With those references that are engineering focused, the predominance seems to be towards the software domain, in which many of the mathematical principles and information theory concepts can be applied more readily.

There appears to be a great lack in complexity literature specifically for systems engineering, Bar-Yam Yaneer (2000) and Martin (2004) appear to be the only two specifically targeting engineered systems rather than natural or mathematical systems. This is perhaps because in reality engineering approaches do not create what would commonly be called or referred to as *"complex systems"*, or *"complex adaptive systems"*. Indeed the systems created are less complex when compared to the scale of natural systems, but none the less the *"complexity"* faced in design is an ever increasing problem that must be addressed, and a common language and understanding of that engineering complexity must be found.

The products themselves, that is the hardware and software within the system without operators, in fact mostly exhibit Senges (1994) dynamic complexity. However, the hardware aspects of the systems - that is, the interfaces or elements within the system - are always the same; any adaptation exhibited by the hardware or software system is usually a deterministic process. The human element within operational systems provides the system with the ability to truly adapt.

Complexity in systems that are engineered can be thought of in this way:

System	System Boundary Includes Hardware	System Boundary Includes Software	System Boundary Includes Human Users	System Boundary Includes System Operations	System Can Exhibit Hierarchical Complexity	System Can Exhibit Non-Hierarchical Complexity	System Can Exhibit Adaptive Complexity
1	X				X	X	
2	X	X			X	X	
3	X	X	X		X	X	X
4	X	X	X	X	X	X	X

Table 4 - System complexity properties as a result of the system boundaries.

As a result, during development of hardware, the complexity characteristics that are in fact of concern are those that related to Hierarchical Complexity, or Non-Hierarchical Complexity. When developing systems including the human element Adaptive Complexity must also be considered.

The various definitions within the literature focus on very similar points and appear to be variations on very similar themes. The clear identifiable characteristics of complexity in engineered systems are:

- Difficulty in modelling
- Interface coupling, connectivity and intricacy
- System understanding (ignorance, maturity)

Those items that are also potential characteristics in complex systems but not necessarily in every case:

- Maturity
- Commonality and variety
- Irreducibility
- Scale
- System sensitivity

A definition of complexity for engineered systems is required for this thesis, and this definition must encompass the first set elements above if it is to be complete. No single definition uncovered throughout the research spans the breadth outlined above. Evans' (1987) definition goes part of the way to defining complexity, Sussman (1999) then completes the definition introducing the concept of knowledge, emergent

properties, and the ability to predict system behaviour. A complete definition of complexity lies as a combination of the two, and this combination is the definition that is used within this thesis.

System complexity is the result of (despite a good understanding of the elements and interfaces within the system) the intricacy and coupling between elements within the interfaces. The system behaviour is difficult to model and emergent properties that are unpredictable are observed.

Measurement of complexity is also limited within the systems engineering domain. There are a large number of complexity measures that are applicable to other domains but not systems engineering or engineering specifically. These measures however have concepts that have relevance; however it is not always clear how such measures can be implemented.

What is clear however is that the measurements here, specifically target certain complexity types, and no one measure covers the full spectrum of complexity; modelling difficulty, interface connectivity / coupling / intricacy, understanding, commonality / variety, irreducibility, scale and system sensitivity.

Martin (2004) attempts to model the complexity of a system simplistically, with careful detail paid towards the interfaces and elements. The problem with Martin's approach is that the system complexities are almost unrecognisable from the output, and it overlooks some quite key elements of complexity in engineering development (maturity, level of understanding, system sensitivity). Martin's metrics attempt to quantify a complexity value for the system based on purely the functional or physical characteristics of that system only. From a development point of view this is only part of the story; system complexity in this regard may be high but in reality the complexity or difficulty associated with creating this product may be low as the majority of the technology may already exist, the product may be an upgrade in which case most of the complexity of the system is already dealt with. Calculating the complexity value for a system is not enough for development programmes, converting a "complexity value" for a system into an estimation of difficulty or effort required is difficult without considering other factors such as maturity. The methodology for calculating some complexity values for systems are useful, in particular when

considering "vocabulary" of links and element functions. These aspects of the complexity metrics have been adopted for this thesis.

The concepts or classifications of complexity are very relevant to systems engineering problem issues, and highlight the key differences between simplistic and complicated engineering problems. Hierarchical systems are more easily understood and modelled as information paths are restricted, whereas those systems with no linear hierarchy require a great deal more effort and understanding.

2.10 Summary

There is a large amount of literature covering complexity in various different subject areas (mathematics, computer science, etc.). The literature review has been structured with categories of complexity attributes; definitions, origins, classifications and so on. These gathered characteristics from the literature form the Complexity Component and Characteristic Store (CCCS). The CCCS also stores information regarding problem issues within engineering systems. These were not covered within the literature review as this information was collected later from case study analysis, and complemented the literature review. Figure 10 shows the CCCS along with the attribute categories that are contained within it.

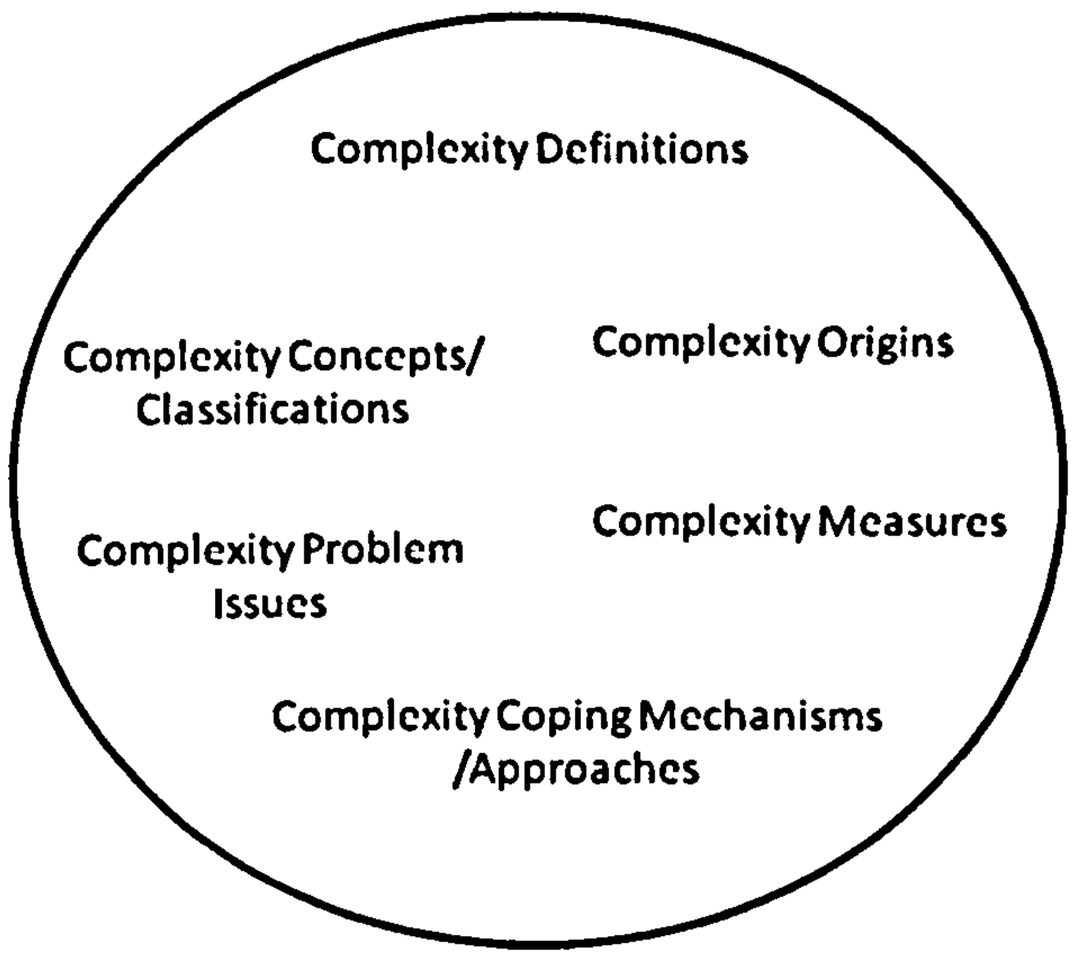


Figure 10 – The CCCS and the information it contains.

The CCCS contents are described as follows:

Complexity Problem Issues - Problem issues and case studies that are gathered from industry are stored within the complexity component model. These complexity issues

are driven from those problem issues that have been collected and analysed from the case study activity.

Complexity Definitions - A set of complexity definitions, not just linked with the systems engineering domain are stored within the complexity component model. These definitions range from natural world definitions relating to evolutionary principles or weather systems, to engineering specific definitions. There is a possibility that definitions from the natural world may apply to the engineering domain.

Complexity Origins - Where does complexity come from, how does complexity manifest itself? Again the component model is not limited to the engineering domain, but is an attempt at an all round view of the origins of complexity within complex systems. The following are potential origin areas for complexity within systems and development programmes.

Internal Factors – Complexity found within the system itself and not outside of it. These are the intrinsic origins that induce complexity during the development process, that are not as a result of human induced design complexity within that process.

External Factors - Complexity factors that are exhibited outside the system, within the system environment that are not under system control. These are the intrinsic system factors that induce complexity during the development process, that are not as a result of human induced design complexity within that process.

Complexity Inducers - Organisational changes, re-structures, personnel changes, operational support organisations etc. may have effects on the system in early concept, development, service, support or disposal phases. These will be organisationally induced factors which influence the level of complexity within the programme domain.

Complexity Concepts / Classifications - Complexity can be classified for a system as in terms of detail, dynamic, complex interface adaptive, or in terms of complex interface and element adaptive. These classifications are explored in detail above (heading 2.6) and are useful when determining the level of complexity that is likely to be exhibited by the system under analysis.

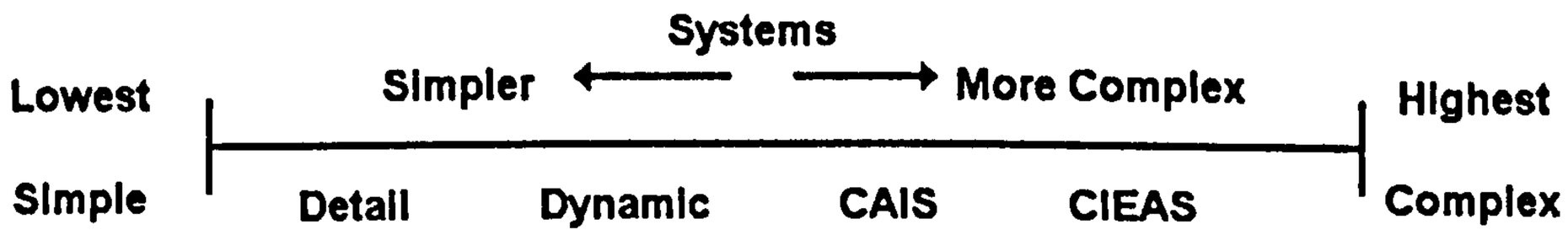


Figure 11 - Complexity scale of complexity classifications.

In general, systems become more complex as they approach complex interface and element adaptive. Figure 11 shows the scale for the complexity classifications that are contained within the complexity component model.

Complexity Measures - A number of complexity measures are contained within the complexity component model. These measures range from quantative techniques to qualitative approaches, and are stored as potential measures against the problem issues encountered within the case studies.

Complexity Coping / Approach Mechanisms - Potential coping mechanisms, used or theoretical are stored so they may be applied theoretically, practically, or their impact understood from previous use within business. These mechanisms will be linked to problem issues within the complexity domain in systems engineering.

The CCCS is essentially the product of the literature review and provides the attributes of complexity used in later stages of the research.

The roadmap within Figure 12 shows the links between the literature review findings and the development of the CCCS within chapter 5.

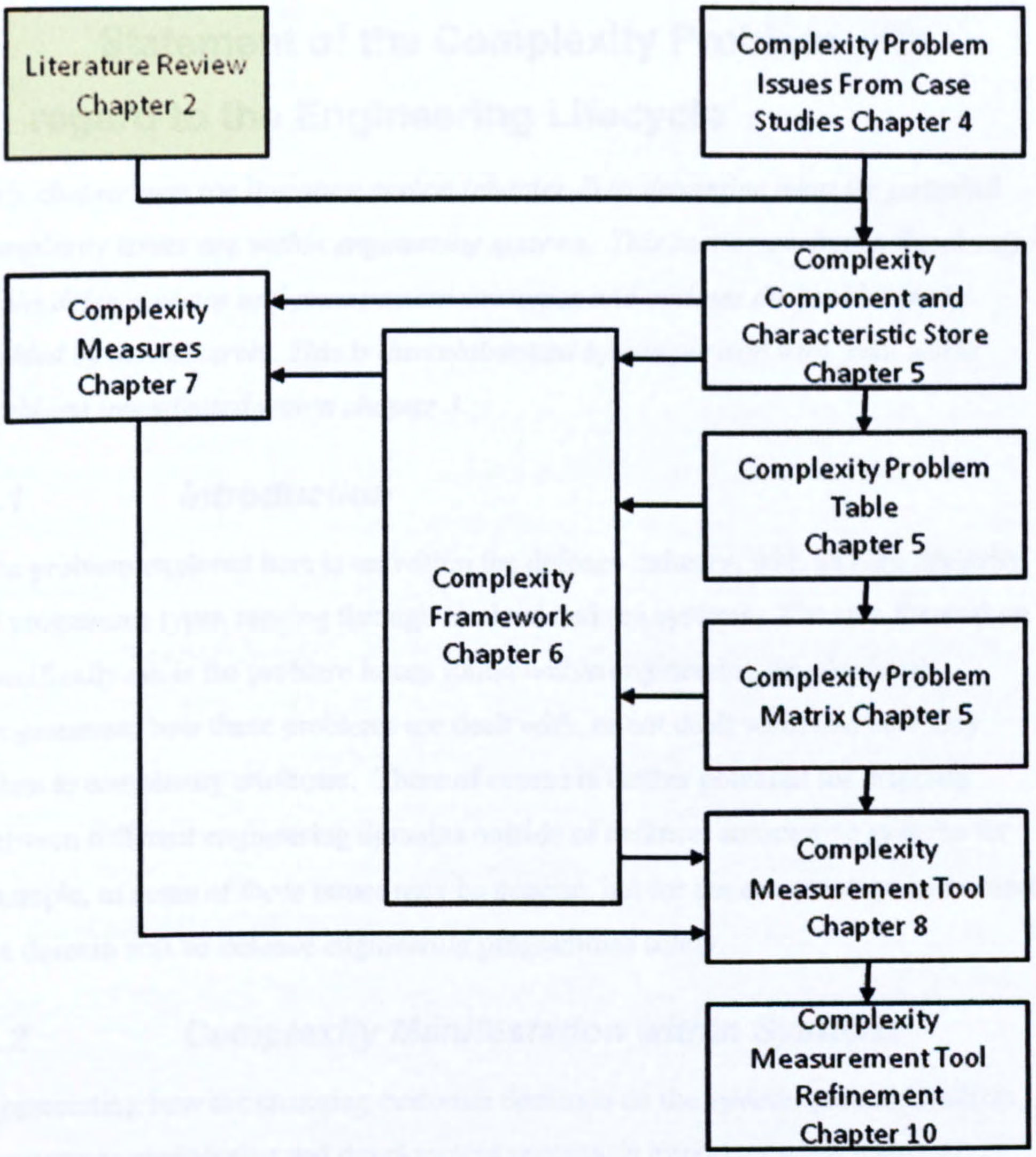


Figure 12 - The layout of the work and the thesis outputs roadmap.

3 Statement of the Complexity Problem with regard to the Engineering Lifecycle

This chapter uses the literature review (chapter 2) to determine what the potential complexity issues are within engineering systems. This section evaluates the changes in the defence sector and procurement strategies and outlines the problem to be tackled by the research. This is then elaborated by comparison with 'real world' problems investigated within chapter 3.

3.1 Introduction

The problem explored here is set within the defence industry, with its high diversity of programme types ranging through air, land and sea systems. The area focused on specifically are is the problem issues found within engineering development programmes; how these problems are dealt with, or not dealt with, and how they relate to complexity attributes. There of course is further potential for mapping between different engineering domains outside of defence, automotive systems for example, as some of these issues may be generic, but for the current stage in research the domain will be defence engineering programmes solely.

3.2 Complexity Manifestation within Systems

Appreciating how the changing customer demands on the systems procured affects the systems engineering and development process, is important when trying to understand how complexity might manifest itself.

The systems engineering approach (Haskins 2006, Bock 2005) has been fashioned in response to the change in system properties - size, functionality, interoperability, functionality and technology. As a discipline systems engineering was created to deal with the issues of large scale interoperability, functionality and technology integration, while still remaining competitive within a product market.

Technologies evolve, so that they support ever more integration and interoperability, and thus generate ever more complex systems (Modis 2002). Coupled with the scale of some programmes and the programme life (sometimes in excess of 40 years), customers demand more functionality (in new scenarios not necessarily apparent at

the start of the programme), improved reliability and sustainability over the life of the product.

The traditional engineering approach used for product development started from a stagnant user requirement definition based on defined scenarios for the product drawn from requirements established by the customer, which were not subject to any evolution over the product lifecycle. Once requirements and scenarios were so defined this was an almost linear process. The product was defined, requirements created, test plans for those requirements and acceptance criteria created, initial concept designs fashioned, detailed designs and interface specifications created for sub-systems, and these were then integrated and tested against the test plan created in the early stages of development.

This fixed requirement approach enabled a number of engineering practices which can accelerate the development process with little risk. Concurrent engineering can be easily managed with a clear understanding of each sub-system and the interfaces between them; as a result it is suitable to the fixed requirement based development programme. The lack of the need for a lengthy iterative development process reduces the timescales for development.

Figure 13 shows the development process in the context of the relationship between the *'system operational capability'* (i.e. the system under development by the contractor and operated by the customer) and the *'development capability'* (the contractor or organisational structure that facilitates the conceptualisation, design, integration and testing of the system), along with the influential factors affecting both.

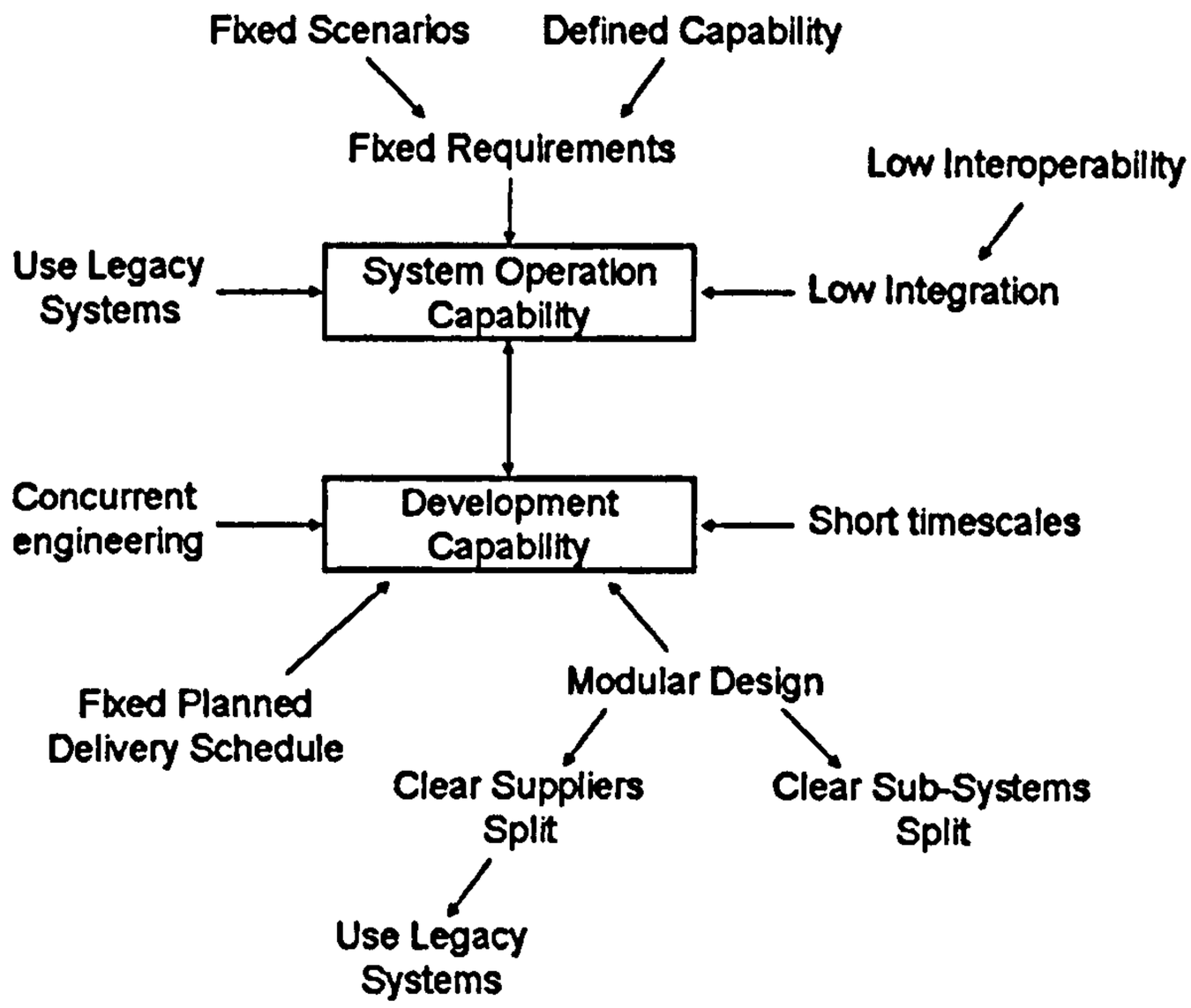


Figure 13 - Traditional systems engineering problem.

The system operation capability as discussed is shown to have a fixed requirement base, with low integration and interoperability due to the lack of supporting technology with the additional reuse of legacy systems. A great number of development programmes do not start from a blank sheet of paper, but are often upgrades or enhancements of existing products.

Modern development programmes do not often have the rigidity in their requirement and capability definition that traditional programmes used to, and are subject to evolution over the development cycle. The MoD UK has introduced capability based procurement strategies or Smart Procurement (UK Ministry of Defence 2001, James 2004) which is geared towards this evolutionary requirement concept. The traditional engineering approach would not easily cope with requirement evolution - it could be argued that our current processes are not much better, either. The change in procurement strategy means that the demands placed on systems engineering have changed, for both how the process of systems engineering is conducted, and to what engineering practices can be adopted to accelerate the development. Modern systems require more careful management and control for this very reason. The dynamic nature of the requirements definition constitutes a much greater risk in the development of modern systems than for traditional systems; thus, understanding complexity would help in quantifying and mitigating this risk. Systems are expected to have a longer life, probably due to high costs of development, and the products

themselves, in some cases, are developed as *'immortal systems'*. These immortal systems have a huge life, and do not necessarily remain the same product from start to finish; rather, they are subject to upgrades and additions throughout their operational life. This means requirements fashioned initially in a programme may not relate to the real requirement at the end of a programme, so there is no longer a fixed target for engineering to aim at - customers are more likely to want to make changes to their requirements as the programme progresses to meet their current demands.

Figure 14 shows the modified relationships within the development process between the *'system operational capability'* and the *'development capability'* for the modern systems engineering activity.

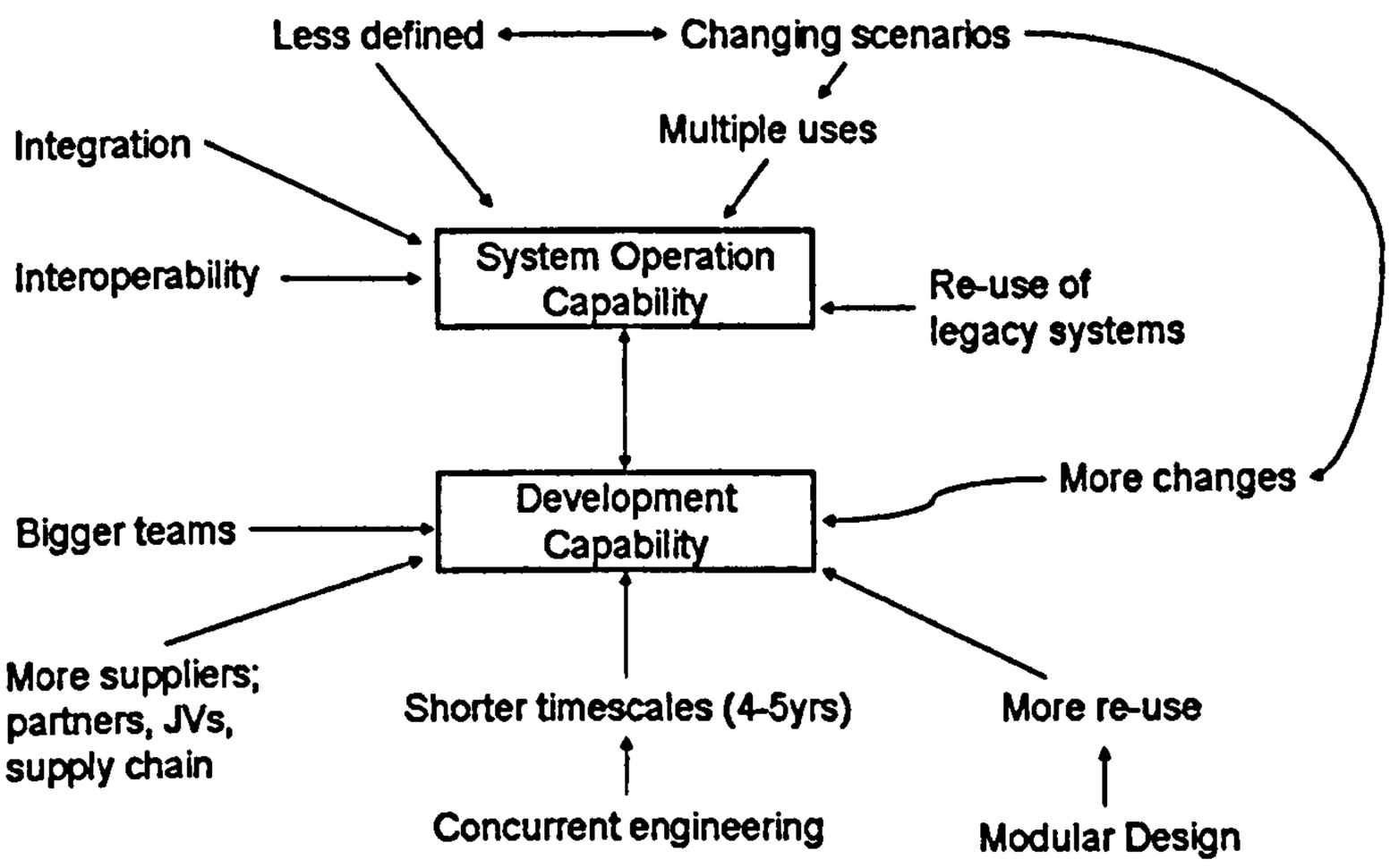


Figure 14 - Modern systems engineering problem incorporating system complexity.

The changing demands have made the overall process much more difficult. Systems are larger, subject to evolution; the development capability (the organisation including suppliers and business units) that creates them are also larger, needing more skills and people to develop the more interoperable and integrated systems with improved functionality. In addition, there has been a significant change in the scope of the engineering activity including more emphasis on support, safety, reliability and the concept of interoperability which have become requirements for new systems (Link 16, tactical data links). The development capability develops operational support mechanisms, detailed safety and reliability assessments and more vigorous testing programmes to test the increased functionality, all of which requires more people, more suppliers and a larger overall organisation.

In addition the move towards smart procurement (UK Ministry of Defence 2001, James 2004) means requirements are becoming more capability based, in other words taking the format *'we wish to have an ability to achieve set out goals'*, rather than, *'the system shall do x'*. This approach requires the contractor to translate these capability needs into an agreed system requirement specification from which a product can be designed. This means more work, and requires contactor interpretation of the capability need, but essentially it is the application of the same engineering strategy for traditional systems, but with a more iterative layer built into requirement derivation; a capability request is fashioned, the engineering approach uses this capability request to fashion a set of rigid requirements (often with no room for evolution), and the system is produced from those requirements.

Industry is struggling with development of systems in this manner; programmes are late, over budget, do not satisfy customer requirements, the requirements evolve and due to a lack of flexibility within the product or process industry cannot cope; examples include defence programmes and commercial airliner programmes (BBC News 2004, Airbus 2006). In addition, the customer demands and expectations continue to increase, in particular within defence and civil aerospace. Engineering needs to be flexible and agile, and adopt a strategy or process that supports evolutionary, rather than 'design to requirement' product development.

Are some of the problems within programmes a result of complexity within the product, the development process or the development capability organisational structure? It may be a mixture of these factors. Products on the scale of modern systems have never been conceived before. Older systems perceived at the time as complex are, by modern standards, easier to design and implement. The technology was older, of limited functionality, and did not require or facilitate the interoperability of systems, as is the case today. An example is computer microprocessors, a technology that has advanced very rapidly. This continuing advancement in microprocessors, (Mollick 2006, Hamilton 1999, IEEE 2006) and computing power have enabled the higher levels of interoperability and functionalities which are demanded of modern systems.

This additional complexity (intricacy, size, information flows, interoperability, functionality, etc.) in modern systems is a major issue for developers; system reliability, emergent system behaviours or properties (be they desirable or

undesirable), become more and more difficult to control and also more difficult to analyse, predict and model. This complexity within systems needs to be understood so that appropriate steps can be taken to reduce any detrimental effects within development programmes. In order to do this industry must be able to understand complexity (origins, effects, classifications, potential problems, concepts, etc.), measure it, and intelligently decide upon action to reduce, eliminate, or cope with it within the engineering processes.

The kind of complexity that is of special concern is not the intrinsic complexity (which is there irrespective of process and a part of the requirement) but the unnecessary complexity induced by engineering practices. A clear understanding of the difference between intrinsic and induced complexity is vital, and the ability to identify, and perhaps measure induced complexity within systems of great benefit during development.

When considering complexity outside the engineering domain (i.e. mathematical, natural systems, etc.), it is apparent that engineered systems are some of the more 'simpler' systems. Complexity on a totally different scale exists within nature - evolution, weather systems, fluid flow dynamics, atomic physics, quantum theory, particle flows, chemical reactions in cells, and so on. We have been modelling and predicting weather system behaviour for some time, but despite this our models are not always accurate; our models, perhaps, are limited by computer processing power, or the scope of the systems is so large it is impractical to model them accurately. In contrast our understanding of designed electronic, mechanical and software components (their functionality, operations, tolerances, and so on) is far more accurate and subsequently their behaviour within a system far easier to model and predict. With our ability to better model and understand the systems created by development and engineering processes, it would seem that it is within our grasp to better understand complexity in our systems, and improve our ability and efficiency in developing them.

Industry already has some coping mechanisms that attempt to deal with the complexities within product design (see section 2.8 above), but there is still room for improvement. Industry needs to continually innovate and create new products to remain competitive and effective, this means even more complicated and intricate systems will be required, while at the same time delivering these systems with high

reliability, reduced cost and increase capability effectiveness. An increased understanding of complexity is a beginning at dealing with this situation.

3.3 What is needed

Within industry there is no common complexity understanding or language when describing complexity in systems that are either operational or in development. This understanding or complexity in system frameworks will form the basis from which industry assesses, understands, measures, and deals with complexity in system development, operation and disposal. The elements of the understanding required are as follows:

- A common understanding of how complexity is defined.
- A common understanding of how complexity manifests itself within systems, the origins of complexity.
- A common understanding of the classifications or concepts of complexity in systems and their implications.
- A common understanding of sensible measurement approaches that measure complexity in a manner that helps the design and development process.
- An understanding of complexity problems, their causes and how they can be avoided.
- A common understanding of intrinsic and induced complexity within systems and the implications of both on development programmes.

The framework itself needs development, and then within each element of the framework work conducted to enhance the understanding within that area.

Within this thesis the complete framework is outlined and created and each element defined ready for further exploration. To explore every element within this framework in the detail required is beyond the scope of this work and quite possibly spans several different research strands into a number of different techniques and approaches; nevertheless, this thesis establishes a basis for all of the elements of the framework and builds upon one element in a little more detail.

The measurement of complexity in systems for system development programmes is tackled in more detail, as no complexity measurement strategies exist that measure

complexity in a manner useful to development programmes (that is providing a complexity figure from a measurement technique without considering the context of the programme doesn't help improve the development process).

3.4 Conclusion

Industry already has the systems engineering lifecycle models which come in different forms; the company within the case studies uses its own Lifecycle Management (LCM) guide set. Industry in general uses the systems engineering development profiles, or what is commonly referred to as the *'V-diagram'* (Haskins 2006), and supports this with various other processes, in some cases house processes, such as change management, strategic business strategies, bidding processes and phase reviews. These support the development activity in particular, and were developed to tackle the complexity within engineering systems. However, the demands on systems engineering have changed and will continue to do so, and the systems engineering model must cope with these changes, by improving or developing additional tools to improve industries ability to develop more complex systems.

Many customers have their own processes when procuring their systems, in particular the UK MoD which uses Smart Procurement (UK Ministry of Defence 1999, UK MoD 2001, James 2004), and pays particular attention to the Lines of Development (of which there are now 8 lines) within this procurement framework.

Combined, customers and industry have developed processes and tools that help them manage the product lifecycle (requirements management tools (Telelogic Doors), system development and modelling tools (Telelogic Rhapsody), modelling languages (Weilkiens 2008, SysML Partners). It is not certain whether these tools or processes give industry and the customer a view or understanding of complexity in their products, nor how it might affect them in the future. Further development of these tools is required to deal with the changing nature of systems engineering.

4 Initial Complexity Case Studies

The following section analyses a set of case studies which identified typical industrial problems and also how they may be complexity issues, or related to complexity issue. It expands on the theoretical from chapter 2 and the problems identified within chapter 3.

4.1 Introduction

A set of case studies were used in order to gather information surrounding problems within industrial programmes. The case studies covered the problem issues in detail and any action that was taken to mitigate or deal with those problems. Each case study section details the problems taken from each case study and an evaluation of that study with the relevance to complexity characteristics (from the literature review) explored.

These case studies were not specifically selected because they contained complexity issues as it was considered too difficult at this point to differentiate between a problem resulting from complexity from one that did not. However, a number of the problems identified within the case studies had characteristics that implied complexity may be a cause.

Understanding the links between problem issues within development programmes and complexity characteristics within a system is vital if a technique is to be developed to quantify complexity in a way that can improve the development process within the product lifecycle. With clear links between issues in development programmes and complexity characteristics (if of course these characteristics are quantifiable) the extent and origin of the complexity within the system can be established.

4.2 Problem Case Studies

Data was collected from a series of interviews with a defence contractor spanning a number of different development projects. The individual interviews focused on a single project and attempted to gather an appreciation of the types of issues that occurred within that programme which could later be analysed for links to complexity origins and also to determine any common issues between the different programmes (case studies). The programme case studies are outlined briefly below:

Case Study 1 – An upgrade of a legacy missile system.

Case Study 2 – A technology demonstrator for a ground surveillance system.

Case Study 3 – A technology demonstrator for a large ground anti-aircraft system.

Case Study 4 – A new attack submarine programme.

Case Study 5 – A new technology demonstrator for an air stealth system.

Case Study 6 – A integration programme for integrated capability in the battle space.

The data collection exercise, yielded a set of twenty seven issues ranging from simple problems with low complexity caused by hardware failures, to problems with high complexity caused by large optimisation issues in design or requirements evolution during development.

4.3 Case Study 1 – An Upgrade Missile System

The case study data comes from a development programme for an upgrade to a previous missile system used by the navy. The upgrade consisted of the development and integration of some new functionality while preserving all the previous functionality of the legacy system it would replace. Due to the lack of data surrounding the old system, a reverse engineering process was required.

4.3.1 Problem 1 - Company Organisation

There were two specific business units working on the project; these business units were part of the same company. Initially, a single business unit was the prime contractor for the programme, but portions of the programme were put to tender. These programme elements were bid for by both external and internal businesses. The internal business unit secured one of these contracts, and a contractual arrangement was drawn up between the two internal company business units. A second contract was secured by an external business which also had a contractual relationship drawn up.

As the sub-contract is between two internal business units, and the work was initially subject to external competition, a formal prime to sub-contractor relationship and contract had to be created. For the two internal business units of the same company

the formal contractual relationship (necessary for fairness to be maintained as a result of the initial competition for work) created unnecessary boundaries. These boundaries included communication with other the external contractor, which was all done via the prime contractor internal business unit. This was particularly unfortunate as the internal sub-contractor and external sub-contractor businesses would need to exchange a lot of information during the development of this product.

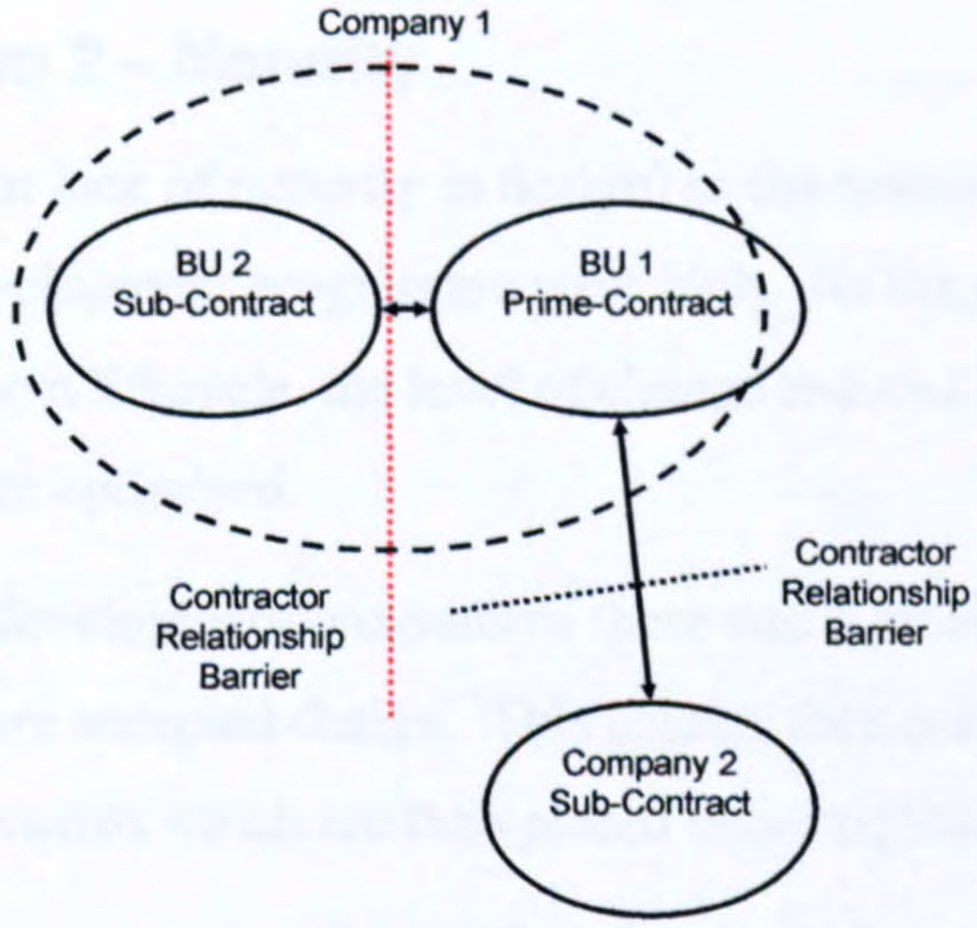


Figure 15 - Company contractual organisation.

Figure 15 shows the company organisational structure, with the contractual relationships and boundaries. Eventually the organisational structure was re-organised to remove the contractual boundary between the internal business units (see Figure 16).

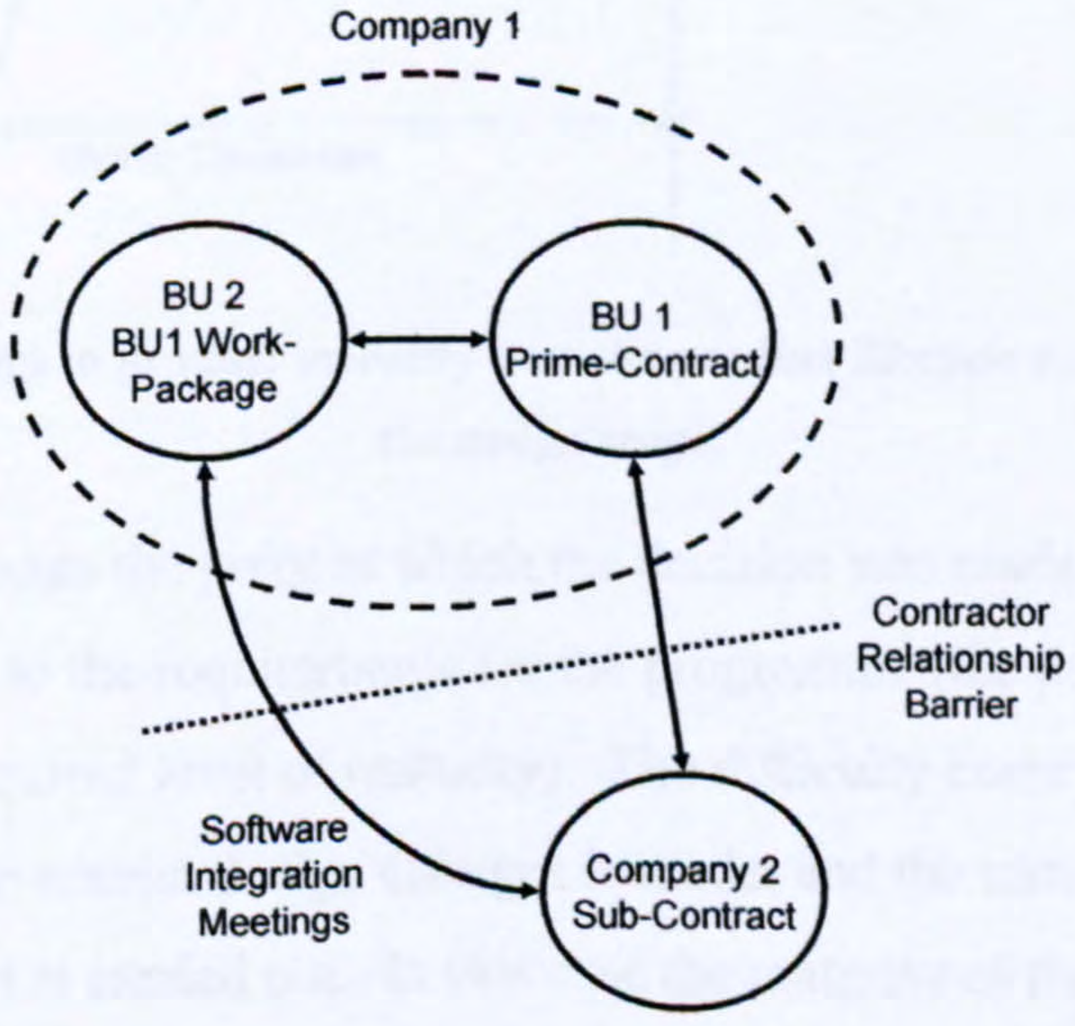


Figure 16 - Company contractual organisation, after the change.

The contractual relationship between prime and the other internal business was removed and replaced with a direct feed. This turned the sub-contract for the internal business unit into a work package for the prime contractor; however the prime contractor still carried overall responsibility for the programme. The contractual communication channels could now be relaxed allowing easier communication with the external sub-contractor, helping enormously with the development of the software.

4.3.2 Problem 2 – Maturity

The level of change (or lack of maturity in design) in the system design within the early stages of the development programme were high. As the programme progressed through the development lifecycle, the level of change reduced and the system became more and more optimised.

At a point within the development programme there was a move from the conceptual design to a more mature accepted design. This change then restricted any further changes to the requirements which are then placed under tighter change management.

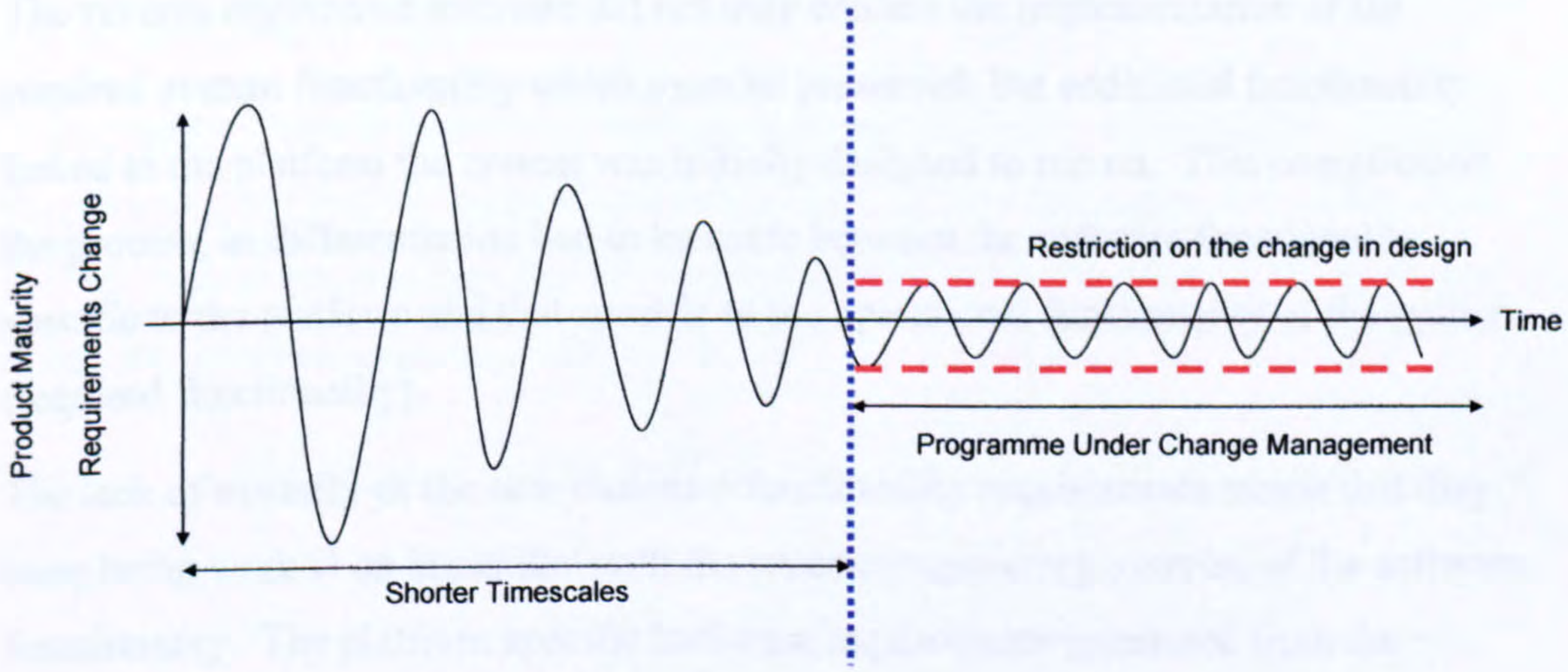


Figure 17 – The change in product maturity over the product lifecycle and the effect of limiting the design scope.

The blue line represents the point at which the decision was made to reduce the level of change available to the requirements for the programme (the point at which the design reached a required level of maturity). The difficulty came when deciding when the decision to restrict design changes is made, and the transition into strict change management is carried out. In this case the maturity of the design (or the reverse engineering of the legacy design) was reached very slowly and didn't fully

settle, as a result deciding when to make the switch to the more tightly controlled set of requirements became much more difficult.

4.3.3 Problem 3 – Re-Use Legacy

The project was essentially an upgrade, with the preservation of legacy system functionality with added functionality along with improved performance and reliability included as part of the upgrade. Due to a lack of a complete specification of the legacy system, understanding of its functionality and operation was not complete. Modifications to the legacy system during the product life may also have contributed to this lack of understanding as to how the product worked. In order to enable the customer requirements to be met, the functionality of the legacy system and all its subsequent upgrading must be maintained; consequently the system had to be reverse engineered into a specification of some kind which can then be followed. This reverse engineering exercise was primarily focused on the software aspect of the product.

The reverse engineered software did not only contain the implementation of the required system functionality which must be preserved, but additional functionality linked to the platform the system was initially designed to run on. This complicated the process, as differentiation had to be made between the software functionality specific to the platform and that specific to the operational functionality of the system (required functionality).

The lack of maturity of the new customer functionality requirements meant that they were being worked on in parallel with the reverse engineering exercise of the software functionality. The platform specific hardware requirements generated from the reverse engineering and the new requirements generated from the customer gave requirements incompatibility which needed addressing.

The initial programme to develop the first product shows a typical systems engineering "V" style approach to the design process. Once in service the system is modified and altered to improve aspects of the design (or perhaps improve compatibility with other potential platforms) that have been found to be inadequate, or that might want subtle change to improve effectiveness. As these activities continue the knowledge of the system is reduced, so once the upgrade programme has begun the understanding of the system is low and there needs to be a re-engineering exercise

to collect information about the product in service so that it can be upgraded. However, the nature of upgrade or modification programmes means that knowledge must be increased in the current product domain before progress can be made on any modification or upgrade to that product, meaning any modification will trigger a need for a familiarisation process. Obviously in this case the modifications carried out did not require familiarisation with the whole system, or significant parts of it, as a result knowledge was low when this programme started.

There is also a distinction between human knowledge and stored information: human knowledge or tacit knowledge will reduce over time (natural wastage, etc. (Gordon, Smith 1998)) and become forgotten due to the long time scales since it was last considered important to any programmes currently being worked on. The knowledge contained within stored information is kept within documents, databases, drawings, requirements specifications or test results, and will be retained until it is deemed no longer useful to any programme or future programmes, or the technology that supports it is obsolete. Often of course this information is stored in a legacy and not always user friendly fashion.

In the event of human knowledge being reduced dramatically, which in this case it is, we are reliant on the non-human knowledge stored within the organisation to increase the tacit knowledge enabling the work to be carried out; in this case unfortunately the stored information was not available either.

4.3.4 Problem 4 - Lifecycle Mismatch

The development programme for the software of the product was done using an incremental process (software produced in builds, each build containing more of the required functionality, see Figure 18), despite the overall approach being a systems engineering approach (Haskins 2006).

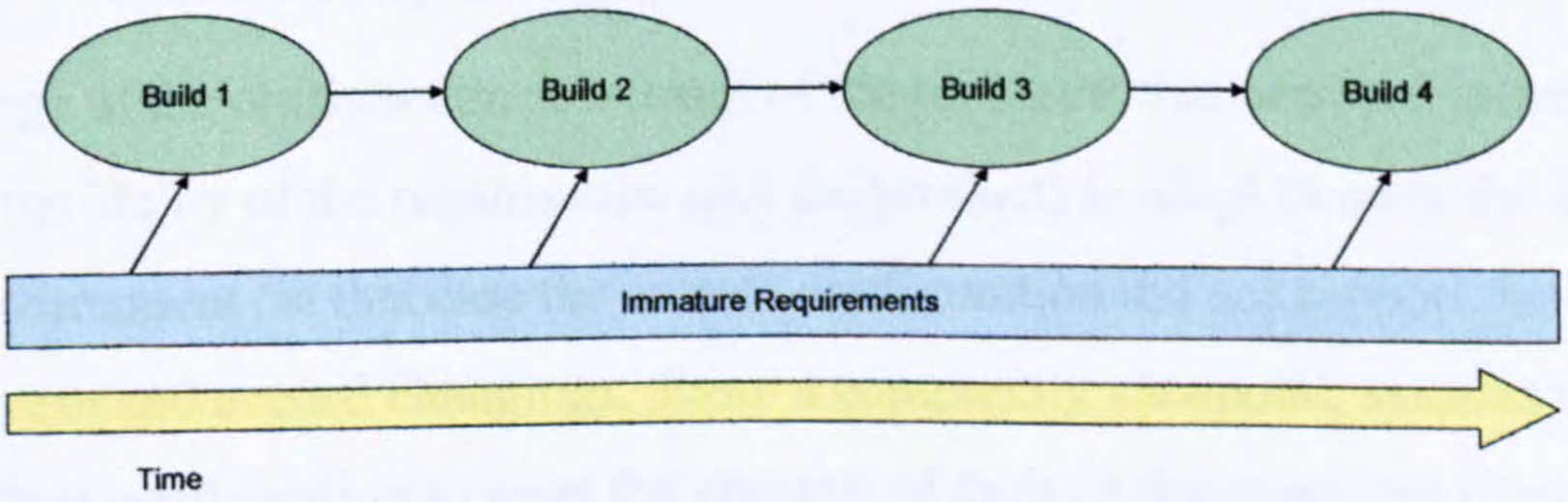


Figure 18 - Software builds and maturity.

The overall approach did not work - the requirements used to develop the first software builds were not mature enough in the early stages and the hardware the software was to run on was not available. Frequently software was implemented on the hardware that was available at the time, rather than the intended hardware.

The maturity level of the requirements used to develop the software was an issue. The low maturity of the overall requirement set meant that software developed for the first incremental build may need complete alteration for second build. As the requirements for the software matured between those incremental stages and became available for the next software builds, in some cases they were incompatible with previous builds, making each software build an entirely custom piece of work rather than an incremental improvement with additional functionality. This large level of re-work meant increased costs, time and an increase in the amount of work that needs to be scrapped.

4.3.5 Problem 5 – Mindset

The mindset of the organisation has an influence on the overall programme. In this particular case some staff considered this project a support programme, meaning it was an upgrade and not a redesign, and subsequently it was believed the engineering process could be accelerated. However, in reality due to the nature of the requirements, the information available and the systems themselves, this project was essentially the creation a new product, and required a development process to suit.

The implication of the upgrade style approach to the project reduced the agility of the organisation to respond to the volatility of the programme, upgrade projects are often simpler and easier to understand, but when they encompass the level of work required here, which was unexpected the organisation is slow to respond.

4.3.6 Case Study Evaluation

The change of the organisation as a result of the problems that occurred initially, indicate the ability of the organisation (not the product) to adapt to meet the demands of the environment (in this case the current configuration did not support the development and needed changing). From a complexity viewpoint, systems that can modify their configuration to meet the changes of their environment are Complex

Adaptive Systems. Although the product itself is not affected, it is clear that the development organisation has complex system characteristics.

The problem is not a complexity problem but a restriction of information flow within the organisation itself. One of the characteristics of complexity is the lack of understanding despite a good understanding of elements and interfaces within the system. In this case there is no lack of understanding at all, the problem is clear and is dealt with accordingly.

The problems associated with the software builds are requirement maturity and development process related. There are distinct incompatibilities between the two processes, and it is this incompatibility that causes the problem. The complexity characteristics one can associate with this problem issue are the effect the lack of maturity of the elements in use (the software and hardware immaturity) has on the complexity of developing systems, which in this case is extremely detrimental.

When the knowledge of the legacy system that is to be upgraded is low, a reverse engineering exercise must be undertaken to develop the functional requirements for the system. This is not a complexity issue in its own right, it is a process issue and a result of the IT technology and data store technology available at the time. In the modern world requirements would have been kept within a database such as DOORS (Telelogic) and modified for each change. Documentation would have been available in number formats and electronically, applications would have provided searching mechanisms to enable the required information to be extracted quickly as assimilated into the new upgrade programme.

The lack of knowledge is not a complexity issue, as it shows a lack of a full understanding of the interfaces and elements within the system (referred to as ignorance). The complexity resides within the engineering process, the ambiguity regarding the system itself is a result of poor information flow and recording.

In the initial stages of development programmes requirements are immature, no programme begins with a completely known and understood requirements definition. The difficulty is increased here with the compressed timescales and the need to produce something quickly. The problem is not the immaturity of the requirements but the reduced time in which to conduct the engineering exercise.

This is not a complexity issue, this is a problem of time scales commitment to deliver, setting a difficult task initially and having problems due to requirement maturity being low while actual development is going ahead concurrently. These factors will make the development process more difficult but it is not a complexity issue.

This programme suffered due to unrealistic demands on the time scales for delivery of an initial operating capability. A distinct quote on the visit, when a member of the team mentioned that when they joined the team, after it had been running only about 6 months a status was asked, and it was said they were already 2 years late. This is not due to contractor fault, but that the time scales required were unachievable. This is a circumstance issue, and cannot be attributed to complexity as a cause. The mindset issue too was a cultural problem within the organisation, and is really independent of complexity.

4.4 Case Study 2

The programme was initially set up as a joint collaboration between two organisations (BAe and GEC) to build two technology demonstrator platforms. The intent was that the system demonstrators would compete be fed into two larger parent programmes (FRES (General Dynamics UK, UK Ministry of Defence) and FCS(Pike 2008)), forming part of the their solution. The two companies producing these platform demonstrators had their own sub-contracts, however, later these two companies merged. The projects were still kept separate to maintain the competition between the platforms for a share of the overall solution to the larger programme.

4.4.1 Problem 1 – Organisational

After the merger between BAe and GEC (which formed BAE Systems), the portion of land and naval systems sub-contracted by the Lancer project to develop the control systems for the Lancer vehicle became C4ISR. Royal Ordinance (RO) Defence also had a portion of the contract allocated to them, and since they were on the same site a lot of the team members were already known to each other. Essentially these were two different business units, but the close proximity of the two and the familiarity between them meant a relaxation in the contractual agreement between the two (Figure 19 shows the organisational relationships between the business units).

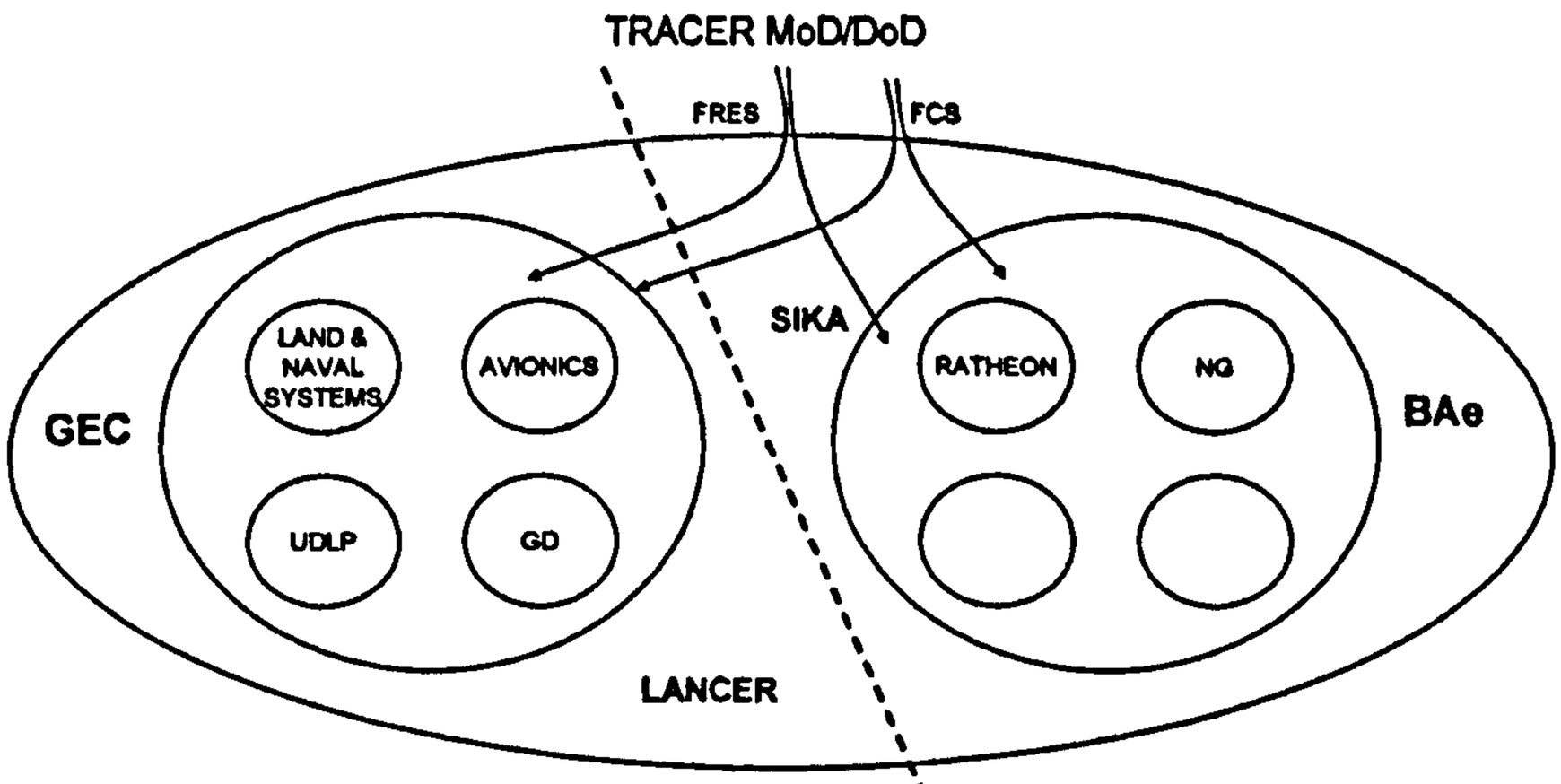


Figure 19 - Tracer organisational Structure.

Problems here included the lack of similar assumptions, lack of recording information accurately. Reduced need for vigorous requirements management and documentation, a lot of the work was conducted by word of mouth and personal agreement to save time and money. Managing the relationship between the business units as a contact was considered a large overhead and this was done partly to save money and time.

4.4.2 Problem 2 – Engine Controller

The Lancer drive system consisted of a diesel engine that provided electrical power by means of generators to motors which drive the wheels. The power to the motors was controlled using a solid state controller, which provides power using a square wave with changing properties. When the wave is at state 1, power is provided, when it is at state 0, power is not provided. The power to the motors is varied by changing the amount of time the controller allows the wave to remain in state 1, for low power, the periodic square wave remains at 0 throughout most it's cycle, and for high power delivery the opposite. The nature of the drive system means that to make the vehicle move faster one simply provides more power through the controller.

The Lancer vehicle has a handbrake system, this is applied whenever the system is stationary for obvious reasons, however due to the nature of the brake and the controller (along with the fact that there is very little in the way of warnings for the brake being applied) it is possible to move the vehicle with the brake still applied by simply providing more power through the controller. What actually happened on the trials was just that: the users moved the vehicle using the controller unaware the

handbrake was still applied. As a result a high amount of power was pushed through the controller for an abnormally high amount of time. When the controller finally failed, the users exited the vehicle and it was collected by engineers, and they found it with the handbrake applied. It was assumed that the users had applied the handbrake to the stationary vehicle, not that it had always been applied. This problem proceeded to confuse the engineers throughout the trials programme.

4.4.3 Problem 3 – Map

Lancer used a mapping system consisting of a liquid crystal display (LCD) that showed the location of the vehicle using a global positioning system (GPS) mapped onto a conventional map display. The mapping system however was found to be quite inaccurate in defining the location of the Lancer vehicle. This mapping error was eventually linked to the curvature of the earth distorting the conventional maps that were flattened and used in the system, which do not take into account earth curvature and therefore produced an inaccurate position. Eventually this was recognised and a standard agreed, the WGS84 standard was employed across the board and as a result the mapping problem was fixed.

4.4.4 Problem 4 - Coiling Cable

The sensor pack on the back of the Lancer vehicle is connected to the internal systems via a large cable. The pack is raised and lowered on a periscope style fixing; as it is raised and lowered the cable must be coiled within the vehicle and protected. During the design stages there was difficulty in getting the cable to coil and lock away, however a design solution was implemented that allowed the cable to be stored and coiled correctly as the sensor pack was retracted.

4.4.5 Problem 5 – Design Process

The Lancer design process employed the use of a simulation setup known as ESIL (electronic systems integration laboratory), this coupled with a CSIL (computer systems integration laboratory) and the real Lancer vehicle provided the basis for the integration exercise and testing exercise for the software running on the system hardware.

ESIL – The ESIL used PCs to simulate different system components, each PC mimicked the behaviour of system hardware within the Lancer vehicle. This was then used to test various software components and interfaces.

CSIL – This is the next stage of the integration process, real hardware is integrated into the simulation ESIL, to for a CSIL integration lab. This allows real hardware to be tested within the simulation environment while connected to PCs that simulate the rest of the hardware.

LANCER – The CSIL can be connected to the Lancer vehicle, which is the last stage of the integration process, essentially if hardware will work within the CSIL simulation it should operate without problem within the vehicle environment, if there is a problem the problem will most likely be something to do with the vehicle environment and not the software running on the components.

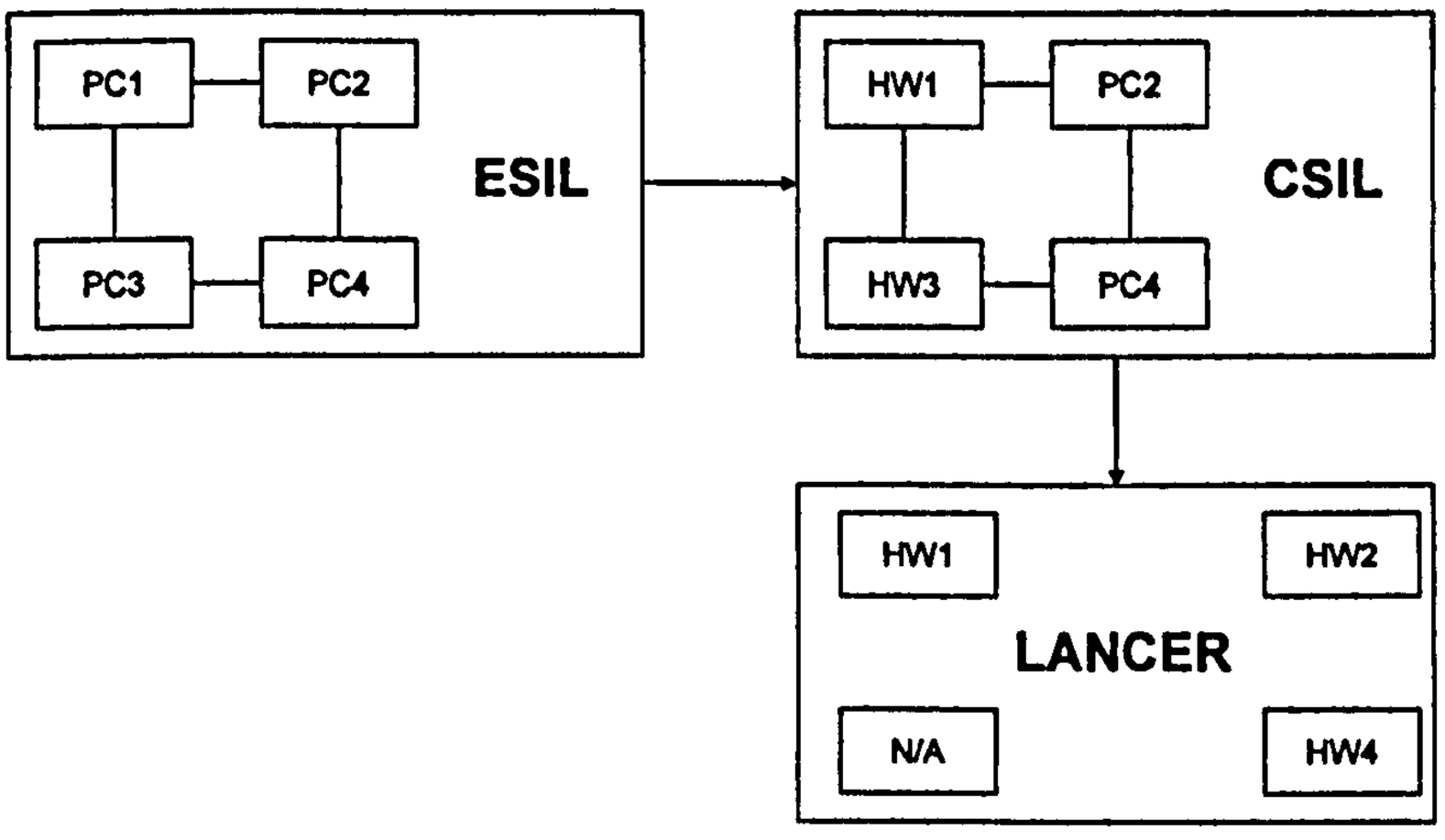


Figure 20 - Lancer development process model.

This integration strategy was never employed on the actual Lancer programme, the ESIL was essentially bypassed and all the testing was conducted within the CSIL environment before it was integrated onto the final vehicle platform. The integration process may have been too long for the whole integration exercise to be employed; as a result it may have been cut short in an effort to complete the prototype on time.

4.4.6 Case Study 2 Evaluation

The problems within this case study are for the most part independent of system or product complexity. The only exceptions are the engine controller and the mapping, however even these could be simple failures in functional design or operation.

The engine controller is a classic case of understanding the functions of the system in terms of elements and their interfaces, but despite this understanding and knowledge this emergent property is realised. The mapping issues once again are similar, the knowledge was present but the emergent property was realised due to a slight flaw in the design. This is perhaps has tenuous link to complexity within the product system, one is a fault with the operational guidelines for the vehicle (engine controller) and the other is a fault with the initial designs as something was overlooked.

The coiling cable problem issues when retracting and extending the sensor pack, are more a result of engineering design than complexity in the design, it was a challenge that had to be overcome not a result of intricacy, scale, coupling or lack of understanding or modelability.

The design process and organisational change were simply implemented to reduce timescales and budget requirements; as a result once again this is independent of complexity within the system.

4.5 Case Study 3

The project started in 1981, and was for an anti-air tank operating at a short range as a private venture, which involved the UK government (which provided £10m funding incentive for the project). The Marksman system developed by Royal Ordinance and BAE Systems is an anti-aircraft turret, with 2 35mm barrel machine guns capable of 70 round s per minute. The system was developed for the Finnish Defence Force, and was built and is in service with their army.

The system consisted of a new radar concept, a set of Oerlickon guns (Swiss) and a new predictor system. The prototype was designed and developed very rapidly with the first Marksman prototype being designed, and built in about one year.

When the company attempted to sell the product to the UK military, the MoD had not identified a capability gap that Marksman could fill, subsequently the MoD didn't purchase any of the systems, and consequently there was a great deal of scepticism from other customers. Eventually a sale of 6 systems was confirmed with the Fin government.

4.5.1 Problem 1 – Drawing Packs

The technical drawings that were used to develop the system were not kept up to date with the design changes as they were developed. The changes to the designs were actually recorded within change packs which were separate to the main technical drawings and the two never reconciled due to cost issues before the vehicle went to manufacture.

The number of items within the change packs were substantial, and in some cases there would be several changes to one item within the change pack. As a result reconciling these drawings to a consistent set for manufacture was an extremely difficult task.

4.5.2 Problem 2 – Manufacture

Once the system was built it was tested, and completed a set of trials with very good results. The system however was still unreliable at the time, but as this was only a prototype the reliability issues were considered acceptable. It is important here to note that the Marksman product only required the development of the turret and not the base platform, this was simply purchased as a military off the shelf (MOTS) product.

The low order rate meant very little was spent on the pre-production phases of the system, these stages finalise the manufacturing techniques tools and processes required for the building of the system. The lack of a pre-production phase meant that each system was essentially custom built, and as a result were completely different. The lack of commonality and interchangablility of the components of the system meant that maintainability of these systems was at a massive cost which made them impractical economically.

4.5.3 Case Study Set 3 Evaluation

The issues within the Marksman programme, much like those in Lancer, are for the most part not complexity issues. The programme suffered from lack of budget due to reduced interest by the UK MoD which led to difficulties in reconciling the design technical drawings and also a manufacture process that meant each system was in effect a bespoke custom system, making the product support costs much higher.

There were some complexity issues that can be drawn from this case study, and they hinge predominantly around the commonality and variation issues of complexity in systems. The lack of pre-production work in the Marksman programme meant the systems were all independent builds and lacked commonality, this made the systems massively impractical and expensive from a customer perspective. There was a cost associated with the lack of commonality and variation within the system due to the difficulty it would give customers when it came to maintaining such systems as the build complexity of the systems was a factor.

This is different from the lack of a concise drawing pack (main drawing pack and large number of change packs added). The decision to not reconcile the drawings was taken to cut costs, and contributed to a lack of understanding of the design later, not the complexity of the product. The complexity for Marksman, was within the low commonality and independent building of systems.

4.6 Case Study 4

Initially the Astute programme was contracted to GEC, after the merger between GEC and BAe. The contract was for a complete platform, the first contract that actually procured the whole system from a single company. It was initially set out as a procurement exercise and was handled by the Procurement Contacts Office (PCO). The programme estimated cost was in the order of £3bn and has been running for a total of 9 years to date (Naval-Technology).

The submarine is essentially a replacement for the Swiftshore class submarine, and is intended to be a stealthy smaller boat. The original requirement was set against the cold war threat, which has obviously is no longer valid as a requirement at this point in time.

4.6.1 Problem 1 – Requirements Capture

There were 10,000 requirements generated for the Astute system, these requirements were either not complete, inconsistent, ambiguous or design constraining in nature. There were requirements generated for design of the submarine, but key elements of that design did not become requirements. For example, performance requirements were very specific in some areas of the design, but totally omitted other areas which

are also key to the design rendering them incomplete. Capability based requirements were more appropriate than specific performance requirements in most cases.

There were system requirements that were actually contradictory to each other, and subsequently distorted the view of what the system should be able to do.

There were a number of very ambiguous requirements within the requirement set for the Astute programme, and these requirements can be found in many different projects, examples of ambiguous requirements are *"Must be fit for purpose", "Must perform better than the current system".*

Although requirements like these needed to be addressed and proven, they were not adequate in their current form and were highly ambiguous which would result in difficulties getting product acceptance later.

The requirements were analysed, in an attempt to deal with ambiguity, incompleteness and inconsistency within the set and 500 requirements (5% of the total) were labelled as erroneous, or invalid. It is important to consider that at this time, the original 10,000 requirement set containing all the issues still had to be managed and designed against until the analysis was completed which commanded a massive overhead on the programme.

3 years into the programme, a Concept of Operation (CONOPS) was produced along with a 100 page document containing 1,000 requirements that provided a good grounding for capability procurement. The requirements were not design constraining and could be traced to the original 10,000 requirements for compliance purposes but omitted some of the erroneous specifics. This meant that the programme had a set of requirements that were manageable and sensible to work from.

The CONOPS gave the *"fit for purpose"* requirement the grounding that was necessary to achieve acceptance by the customer. Without this agreement regarding the CONOPS there could be no way of satisfying this requirement. Figure 21 shows how the CONOPS helped to identify key scenarios from which the system capability could be tested to prove it was *"fit for purpose"*.

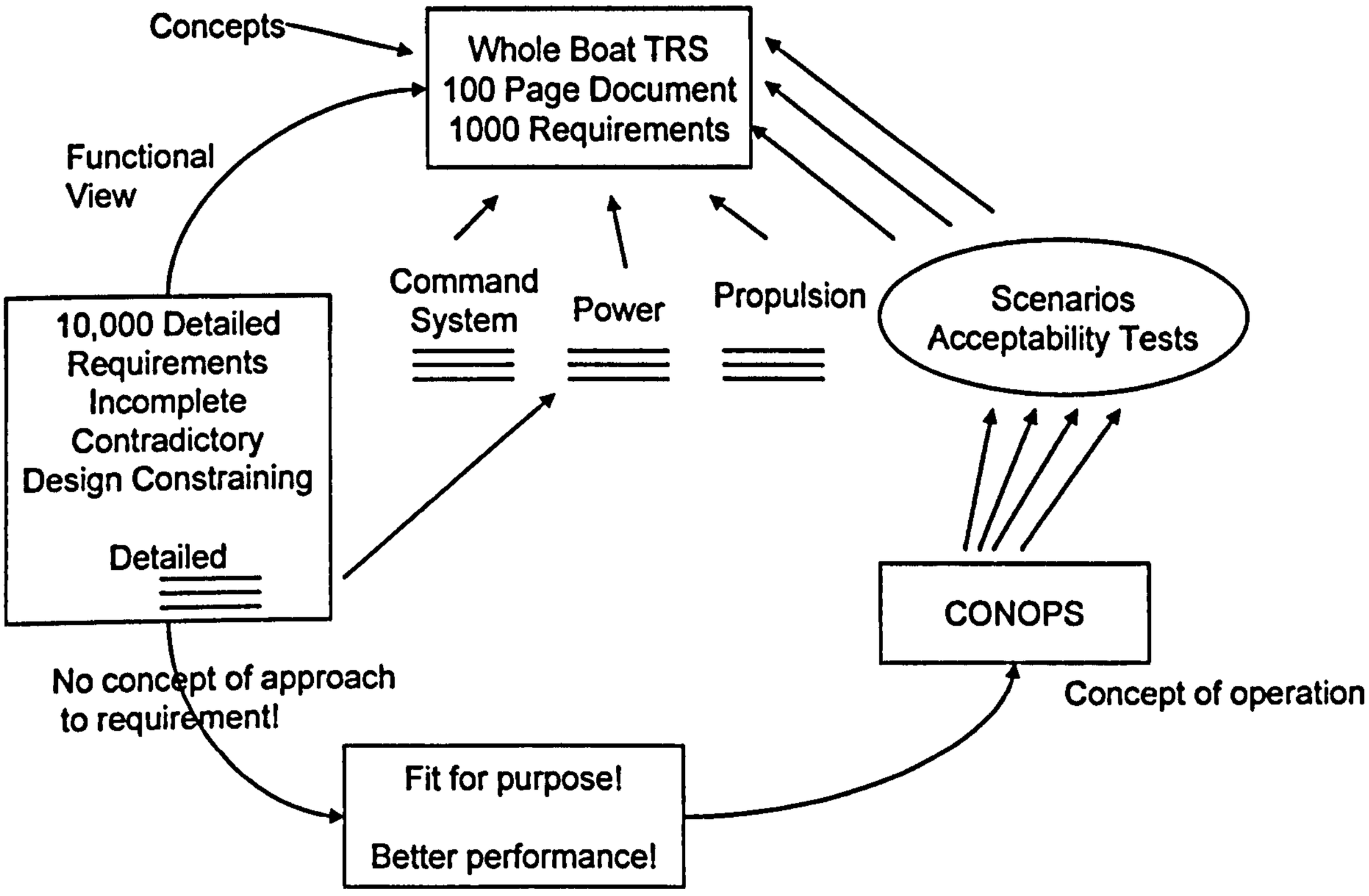


Figure 21 – The internal relationships for the requirements concepts of Astute.

"Better than existing platforms" is also open to interpretation, to be better than the existing platforms, all that may be required technically is an improvement in just one area, perhaps a speed increase of 1 knot for example. Subsequently the CONOPS encompassed all areas where the platform must improve on previous platform performance within its concept. This formed the basis for acceptance against the *"better than existing platforms"* requirement.

4.6.2 Problem 2 – Organisation

There were organisational issues within the Astute programme as well. Below is a typical diagram of the layout of the organisational structure (Figure 22) that sits behind the Astute programme. The project was treated as a procurement exercise and therefore the PCO has control over the programme utilising a set of 200 high level systems engineers.

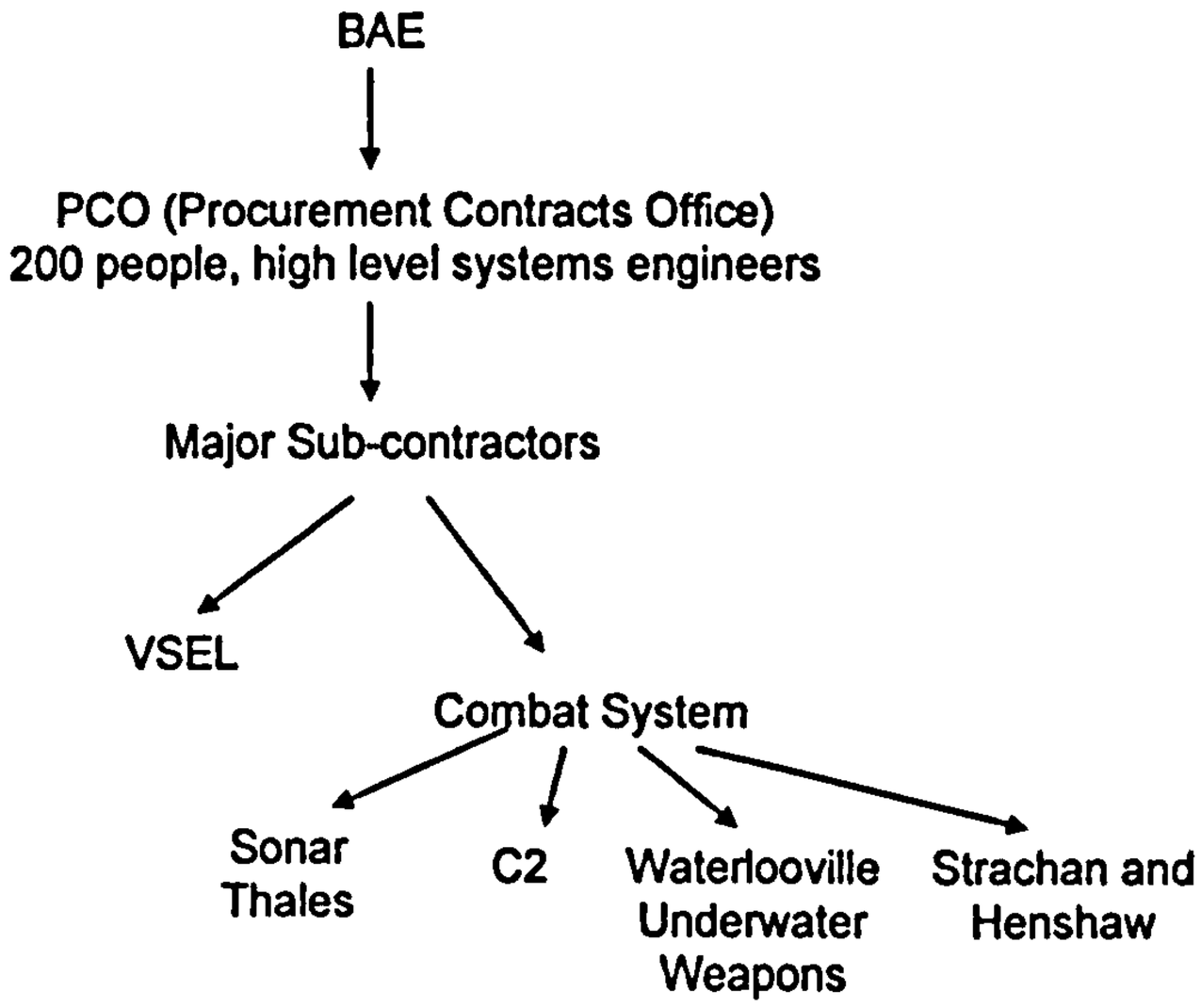


Figure 22 - Organisational relationships in the programme.

These were given the task to assign sub-contracts to procure equipment for the Astute submarine. The amount procured eventually turned out to be measure of the progress of the programme; the more procured the further along the programme would appear to be. This would make perfect sense if the programme was simply a procurement exercise managed by the PCO, which in this case and at this time it was perceived to be, but in reality this was the design and development of a completely new platform, and not a simple case of procurement of equipment, meaning the programme was not controlled correctly.

VSEL (another bidder for the Astute programme) were later purchased by the prime contractor so they could manufacture the hull for the Astute submarine. There were conflicts between the prime contractor and VSEL as they tried to take a more control over the programme, and as manufacture began, more and more of the work has moved to the VSEL sites, and they have fallen into working how they used to work for other programmes which is not appropriate for the Asute project.

4.6.3 Problem 3 – Service Routing

There are a large number of services required on a submarine in order for it to function correctly, and these services all must be routed throughout the boat to the sub-systems that require these services. Routing of services in submarines is more complex than in domestic circles, the following are examples of the considerations that must be made when routing services (electrical, water, data, gas, fuel, waste, power):

- Standards – Standards for cable sizes, loads, separation requirements, protective coating requirements, proximity requirements to other services or heat sources.
- Space – The space on a submarine is very compact and routing of services must be as efficient as possible.
- Fire – The routing of services could provide easy pathways for fire or fumes through the hull, this must be considered when routing is being determined.

Optimising all the variables here is a problem that is ongoing throughout the development of the platform, and there is always the possibility that new services will require further routings.

4.6.4 Case Study 4 Evaluation

The organisational issues are contractual obligations, and configurations are not a result of complexity within the product or the organisation. The organisation simply exists in this way and problems associated with it are process orientated.

The requirement issues are a result of poor definition and a lack of maturity. However in a system this large (10,000 requirements) there are obvious complexities of scale, and interactions will be high within that requirement set. Deciphering that requirement set into a meaningful output and then creating system specifications from that set opens the possibilities for concepts of complexity such as logical depth, descriptive or interpretive complexities or the number of in equivalent descriptions, this is most certainly a complexity task.

The routing of the services throughout the platform is also most definitely a complex problem. The routing can be thought of as an optimisation activity, in which there are a number of variables, constraints and requirements that must be realised in the most efficient manner possible. This is similar to the routing of tracks on a two dimensional printed circuit board, but with a three dimensional system with further additional constraints besides magnetic field effects. This problem is truly a problem attributable to complexity.

4.7 Case Study 5

This was a technical demonstrator programme for the MoD, and was a 50% split between industry and the Defence Procurement Agency (DPA). The system to be created was a radar system for a future aircraft and a conglomerate of companies worked on the project. The contract took 1 year to get agreed, and a further 5 to complete making the project running time 6 years in total.

4.7.1 Problem 1 – Organisational

When the project started the company was about to undergo large organisational changes due to a merger between GEC and BAe. Despite this there was little effect to the team of individuals working on the programme and it maintained the same management and the same key technical staff, benefiting in particular from a single lead from beginning to end.

Maintaining the key staff meant that the tacit knowledge contained within the working team could also be maintained and decisions made early in the programme could be understood later without the need to revisit the problems again.

Other aspects of organisational change which are outside the company, but are key issues that needed to be taken onboard, was the structure of the DPA. The head of the DPA can only stay in post for no more than 2 years before they are replaced. This leaves a 6 year programme the potential of 4 different DPA heads during its lifecycle. A changing customer means a changing customer mindset and understanding of the product which could be disastrous to the programme success. Within this programme however each new head was invited to come to the site and be educated about the project, what it was, how it was being run, and its current stage of development. This education kept the DPA informed and gave them the understanding that was required to provide the right mindset and relationship between the contractor and customer that was beneficial and not destructive to the process.

4.7.2 Problem 2 – Technology

The product being produced was a hardware demonstrator. The sub-systems involved were a radar, radome, radar control system and mounting. There are of course constraints to the design parameters; power, bandwidth of the radars, signatures

produced by the radars to reduce detection probabilities and level of functionality required.

Radomes are materials that are transparent to radars but only within a specific frequency range, Frequency Selective Surfaces (FSS) are used to reflect radar waves or indeed any other wave unless the wave fits within the bandwidth the FSS is designed to let pass. This is achieved using tiny metallic cells within the material, which are sized appropriate to the frequency they must be transparent to, the manufacture of such a radome is difficult, due to the level of the geometric constraints, especially if that radome is to be curved, the spacing and geometrics of the cells must remain exact for the radome to function correctly.

This radar system was to be placed on an airborne vehicle, and as a result must be resilient to the same environment as the vehicle; resilient to lightening, bird strikes, temperature changes, etc, whilst also reducing drag to enable the aircraft to function efficiently. These constraints must all be optimised for the system so that the best design solution can be achieved and built.

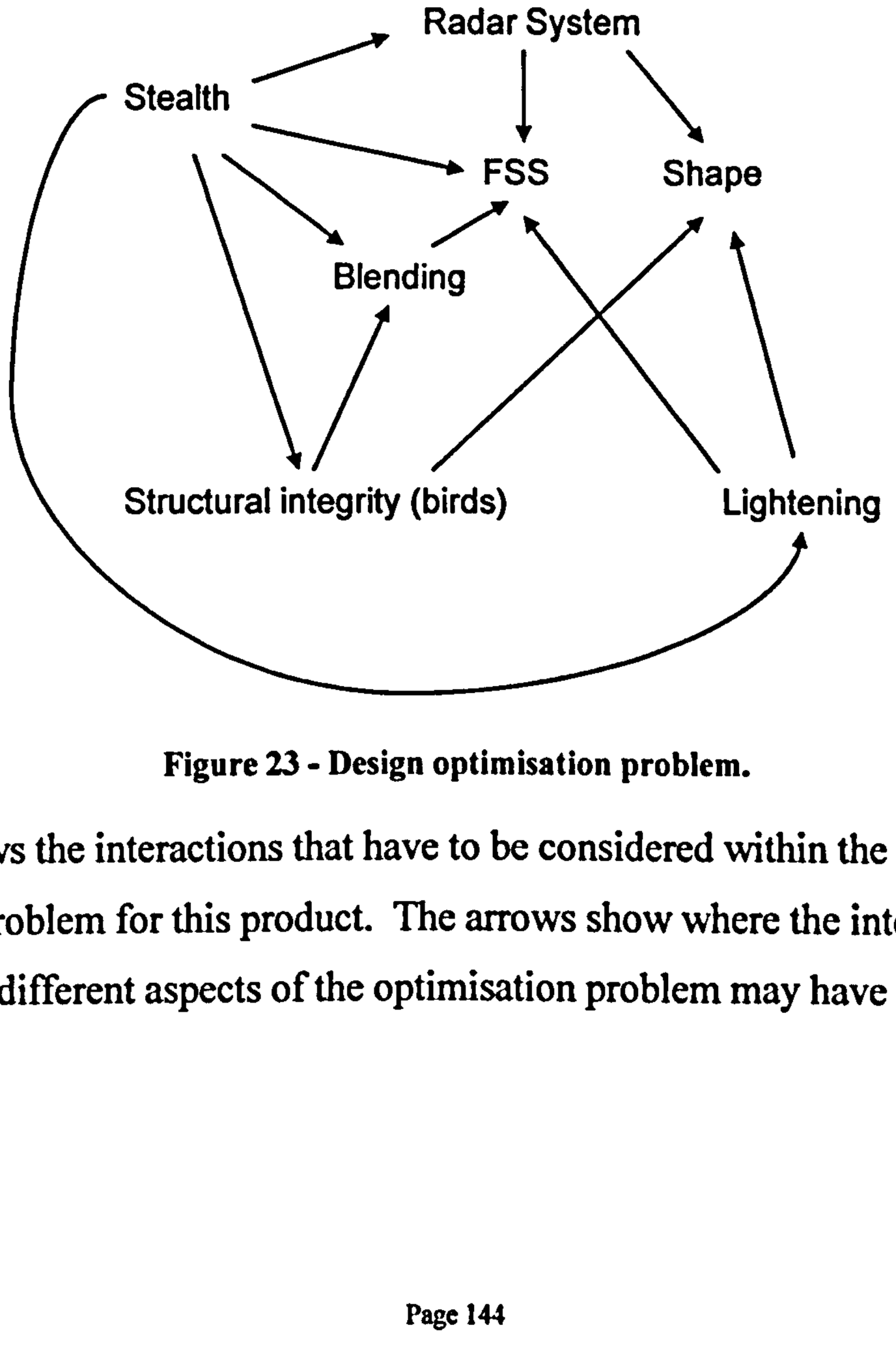


Figure 23 - Design optimisation problem.

Figure 23 shows the interactions that have to be considered within the design optimisation problem for this product. The arrows show where the interactions between these different aspects of the optimisation problem may have dependencies.

4.7.3 Case Study 5 Evaluation

The changing organisation was an issue, but not a complexity issue; rather it was something that needed to be managed and controlled. Along with this was a need for the management of knowledge through loss of personnel which could reduce the understanding of the system.

The majority of the complexity of the system lies within the multi-discipline design optimisation problem. There were a number of variables, some of which are no easily quantifiable, or comparable with each other. This of course over complicates the design optimisation, with inconsistent variables from a number of different disciplines.

4.8 Case Study 6

Integrated Capability Programme (ICP) is the integration of all the BAE Systems component business products, it is essentially the development of an integrated capability that the customer has not yet realised. Network Enabled Capability is the realisation of information transfer between different elements of the battle space; unmanned air vehicles, aircraft, soldiers, ground vehicles, ships, submarines. This information exchange within the battle space is what ICP is set up to deliver to provide additional benefit to the customer and further influence procurement decisions to BAE Systems products.

4.8.1 Problem 1 – Organisation

ICP does not really have a direct organisational structure, and does not really develop a product as such, but by distributing resources from one business unit to another aims to integrate the whole business product line with current and future platforms that may be procured by the MoD. ICP distributes budgets within the business units based on their product lines and invests in the development of technologies that are applicable and to creating capability for information exchange within the battle space that the customer is interested in, or may be interested in, in the future.

4.8.2 Problem 2 – Technology Development

The perception of BAE staff at the time was that ICP produces added value on other programmes within BAE by incorporating NEC into products. ICP was intended to provide the incorporation of NEC into products and relies heavily on new technology

that supports the development of the desired communications to support that network enabled capability. These concepts for NEC capability would be created and then turned into designs, many of which would require technologies still within their infancy. These technologies needed investment to ensure they reach the appropriate technology readiness levels (TRLs) to ensure adequate maturity when the concept moves into the design and production phases, and the capabilities demonstrated for the customers.

4.8.3 Problem 3 – Technology Investment

Figure 24 shows how the technologies (in this case T1, T2, T3) link to the development programmes. There are four levels to this diagram, initially the customer intention is outlined as the programmes the customer will commission in the future. These programmes, and their anticipated start dates will allow the ICP to determine the customer requirements for NEC and align the business investments and aspirations so that they match this development profile. ICP manages the technologies and their development so that the investment strategy ensures they mature at the exact right time. The diagram indicates the point at which they must reach the required level. Investment, buy outs etc. will ensure that these technologies are available and support the ICP concepts.

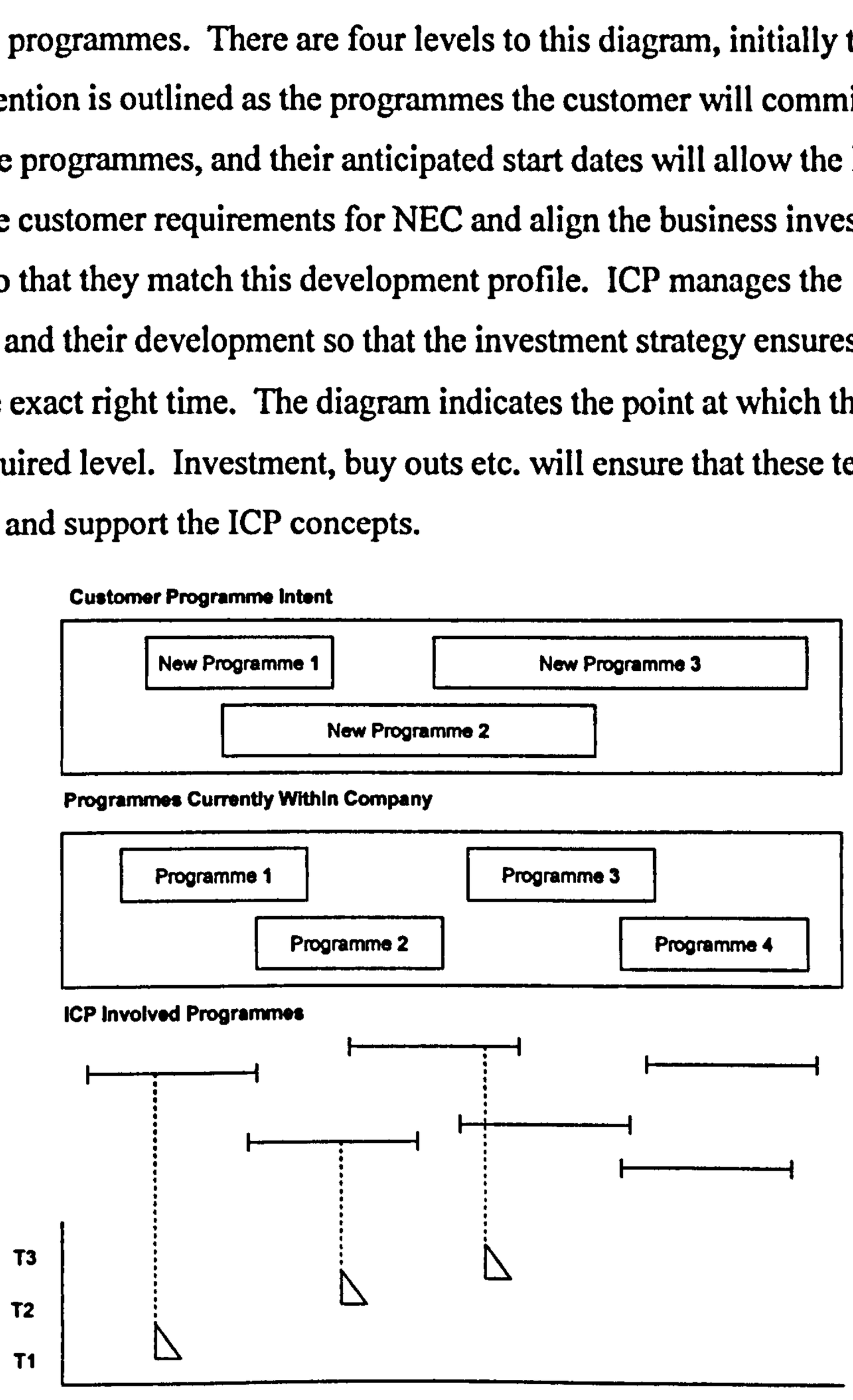


Figure 24 - Technology flow and investment diagram.

The programmes that are currently working within the company also contribute to the more immediate customer and business aspirations in terms of NEC. The ICP will have an involvement in some of these programmes, contributing to the NEC development within those particular product lines.

4.8.4 Problem 4 – Battle space changes

Ironically the point of ICP is to reduce complexity within the battle space by providing cross communication and a common use of technologies. But changing the operation of the battle space with the introduction of extra communication means a huge change in how operations are carried out (CONOPS), and how information is distributed. Effectively ICP has the potential to change the battle space architecture from a conventional hierarchical system, to a non-hierarchical system in terms of information flow and availability. As discussed before, non-hierarchical systems are inherently more difficult to build, predict and model, this may be the biggest complexity challenge ICP will face, modelling and controlling the changes in behaviour this new architecture will have on military operations.

4.8.5 Case Study 6 Evaluation

This programme suffers from the same difficulties as the other case studies we have mentioned here, but there are clear complexity issues within ICP.

Organisational change and fluctuation is a problem for ICP as it spans all the different businesses in an attempt to integrate them. This is a complex issue as the system (as an overall concept of all business) is constantly changing and adapting to customer demands. We have an adaptive organisation, which adapt their business models and practices to meet the demands of the ever changing market, ICP must deliver into this fluctuating market despite its instability.

Technology investment and technology development are a part of the problem of integrating between businesses and products and also steering technological demands for future customer requirements.

The changes within the battle space are definitely complexity issues that need addressing. The changes in communication methods throughout the battle space mean a totally new system is produced that is no longer hierarchical in nature and as a

result new approaches must be undertaken in order to understand and develop this system.

4.9 Overall Case Study Summary

The problem issues were collected from the initial case studies (see chapter 4). The complexity problem issues identified within the initial case studies can be split into different categories, depending on where they occur in the product lifecycle; development stage, manufacture stage, operational support, overall programme.

4.9.1 Development Stage

Poorly defined requirements or capability at the early stages is a clear problem issue in engineering. Additionally poorly defined scenarios for testing and unrefined context of use and operation due to an evolving "threat" (which generates the requirement) during the life of the programme (which could be 20 or more years) means systems need to be constantly updated during their development life. Products on the outset are often unrecognisable at completion of the programme, the changes in terms of the design requirements and operational requirements are enormous. This uncertainly makes design and the employment of the systems engineering development lifecycle even more difficult.

Testing and configuration control is a difficult task for large scale systems consisting of large numbers of interfaces, sub-systems, variety, low commonality and multiple variants. The configuration processes must manage all this information, and tests be carried out on all variants, while also testing appropriately for their differences.

After testing, acceptance is also a problem due to the lack of predictability within the system operation (complex systems cannot be easily predicted or modelled, see the various definitions within 2.3 Definitions of Complexity), emergent properties could cause systems to have a high risk of being unacceptable for the customer.

4.9.2 Manufacture Stage

Supply chains are harder to manage for large scale systems, with large numbers of suppliers and components. The steady supply of components from suppliers must match the need for those components during the manufacture of the products or effect the lead time on the product(s) and the delivery schedule.

Multi-variant systems require different processes for developing the different variants, also a different tool set, skill set and resource load, there will also be differences in the times required to build different variants of the same system.

Finally, changing the manufacture processes to incorporate changes from the evolving requirement set for upgrades or just as a result of the time taken to develop the product is very difficult. In some cases changing the manufacture process incurs a massive cost to the programme, and changes that seem small from a design point of view have a massive effect on the method and order in which parts are assembled or created.

4.9.3 Operational Support

Increasingly procurement strategies are moving towards supported systems for customers. The need for customer support is an additional workload for the development team, and in the more engineering complex systems the more difficult cause and effect is to ascertain in small time frames. As a result systems require more expensive diagnostic equipment and built in test capabilities which need development, and of course increase overall system complexity. As systems become increasingly complex, it is often easier to replace rather than repair, as repairing increases cost of the system maintenance substantially more than simple replacement of units.

4.9.4 Overall Programme

There are the obvious problems within programmes, such as late delivery, over spending and management problems with large scale organisations. But the main problem is the changing nature of engineering in systems. Systems are becoming larger, and the scope of those systems is increasing to cover built in testing, operational support and other attributes. This combined with the manufacturing process and the issues associated with it, makes the product lifecycle process itself a complex problem.

4.10 Conclusion

The case studies identified a series of different problem areas within industry. A set of key problems were carried forward for further detailed analysis. A large majority of the problem issues covered within the case studies were centred on design optimisation issues, immature requirements, organisational change, pre-production work and organisational culture.

Table 5 shows the list of case studies and their corresponding sections and numerations used within the Complexity Problem Table (CPT) which is discussed later with section 5.2 and found within Appendix B.

Case Study Number	Reference
UP01	4.3.1 Problem 1 - Company Organisation
UP02	4.3.2 Problem 2 – Maturity
UP03	4.3.3 Problem 3 – Re-Use Legacy
UP04	4.3.4 Problem 4 - Lifecycle Mismatch
UP05	4.3.5 Problem 5 – Mindset
LA01, LA02	4.4.1 Problem 1 – Organisational
LA03	4.4.2 Problem 2 – Engine Controller
LA04	4.4.3 Problem 3 – Map
LA05	4.4.4 Problem 4 - Coiling Cable
LA06	4.4.5 Problem 5 – Design Process
MK01	4.5.1 Problem 1 – Drawing Packs
MK02	4.5.2 Problem 2 – Manufacture
AT01, AT02, AT03	4.6.1 Problem 1 – Requirements Capture
AT04, AT05	4.6.2 Problem 2 – Organisation
AT06	4.6.3 Problem 3 – Service Routing
IP01	4.8.1 Problem 1 – Organisation
IP02	4.8.2 Problem 2 – Technology Development
IP03	4.8.3 Problem 3 – Technology Investment
IP04	4.8.4 Problem 4 – Battle space changes
TB01, TB02	4.7.1 Problem 1 – Organisational
TB03, TB04	4.7.2 Problem 2 – Technology

Table 5 - Case studies and the references within the problem table.

Each case study is numbered for reference later (LA – Lancer, MK – Marksman, UP – Upgrade Programme, AT – Astute, IP – ICP, TB – Test Bed), usually comprising of letters from the case study name.

Not all problem issues identified within the case studies are a direct result of complexity. Those issues that were determined to be complexity related are discussed within this section and are as follows:

- Multi-discipline Design Optimisation (MDO(Koch, Simpson et al. 1997))
- Requirement Maturity
- Organisational Change
- Pre-production Induced
- Mindset (Organisational Culture)

Design optimisation issues occurred within system design that incorporated the use of a large variety of variables and influential factors that needed a trade off in order to produce the best solution to suit the requirement. The majority of design optimisation issues within the case studies were cross disciplinary, and the variables were in some cases not easily measured numerically for comparison; for example the shape of a surface was an influential variable in one particular design problem, but how is this measured? And how is this measured so it can be compared within the trade off exercise against other variables?

Design optimisation was not directly linked to the programme scale (in terms of budget, personnel working on it, size of product, requirement database size).

There are different design optimisation issues that exist ranging from complex mathematical optimisations to engineering decision based optimisations (Koch, Simpson et al. 1997, Bartholomew 1999, Bennett, Fenyes et al.), the issues within this case study are the latter.

Requirement maturity is a problem that has occurred in more than one of the case studies. It was caused in some cases by the lack of understanding and knowledge in relation to a legacy system which needed to be preserved, and in other cases it was a lack of maturity resulting from incomplete or ill defined requirements in the initial requirements capture process. The problem issues following immature requirements have quite catastrophic effects on the development process, within one case study requirements had to be re-defined in order to complete the set. In another case requirements had to be reverse engineered from legacy equipment, but the reverse

engineering included aspects of the legacy design that were not required and in fact detrimental to the system being developed.

Organisational change or organisational landscape change is a problem that appears to be common among all programmes. In particular those programmes with long lifecycles seem to be heavily effected by the changes in the company organisational structure as development continues.

Pre-production work is important when developing systems where the intention is mass production, as it irons out the manufacture process and ensures it is generic for every system, preserving commonality. This phase would amalgamate design drawing packs, and set out clear manufacture processes to build the system. Without a pre-production phase, the manufacture process is ad hoc and it is likely each system will be different, bringing the importance of system commonality into perspective. Systems with low commonality are more difficult to maintain than systems with high commonality, parts are interchangeable and maintenance procedures remain the same within systems with high commonality. If each system is more or less a custom manufacture, it is not possible to support the systems with an overall service plan or common parts, increasing the complexity of the support task. More elements, more information mean more complexity, and in one programme this actually occurred, although the problem cause was deliberate, the system number to be manufactured was small and a trade off of pre-production work cost and the cost of supporting the small number of manufactured systems with low commonality meant that the pre-production phase could not be warranted. In an attempt to mitigate against this cost, the service period for these systems was reduced.

The mindset. or organisational culture plays a key role within the development process. There have been two cases analysed where the nature of the programme has been miss understood or miss interpreted by the organisation developing it. One case focused on the programme as a simple upgrade programme, and the other as a procurement programme, neither interpretation was correct in both cases, the actual task was a full blown product development process. The upgrade programme was such a significant upgrade that it really warranted the attention of a full product development activity, and the procurement programme was really the development of a completely new platform.

4.10.1 Complexity Problem Types from Case Studies

The case study findings have been summarised in the two tables below into problem types that occur more than once in the initial set of case studies. Table 6 shows the nature of the complexity within the problem issue, Table 7 shows the coping methods or solution options that were used to deal with the complexity issues defined within Table 6.

Table 6 is divided into complexity related issues (horizontal rows) and problem complexity characteristics (vertical columns) which are discussed below:

Human / Technology – Distinguishes between complexity or problem issues caused by the technology being implemented or the people implementing or developing that technology.

Induced / Original / Intrinsic – Characterises the origin of the complexity or problem issues as either intrinsic to the system and therefore cannot be avoided, original to the system and therefore pre-existing within the development programme, or induced by the developing organisation.

Environment / Support / Product, within the Development / Operation of the system – Associates the area within which the problem issues sits in terms of the either an operational or developing system. Further decomposition of the problems is carried out are they arising as a result of the support service, the system environment, or the product itself?

Information Fixed / Moving – Classifies the state of the information within the problem domain, is the information associated with it evolving (moving) or static (fixed). Generally programmes with large amounts of information evolution during the development process are more difficult to manage than those with a fixed information set.

Knowledge Distribution / Configuration – Characterises the knowledge distribution within the problem domain as either good or bad. Problems with poor knowledge configuration or distribution mean poor information flows within the organisation which makes the development process more difficult.

	Design Optimisation	Immature Requirements	Organisational Change	Pre-Production Induced	Organisational Culture
Human / Technology	Tech	Human / Tech	Human	Tech	Human
Induced / Original / Intrinsic	Intrinsic	Induced	Induced	Induced	Original / Induced
Development / Operation **Environment / Support / Product**	Development Product	Development Product	Development Support / Environment / Product	Development Support / Environment	Development Environment
Information Fixed / Moving	Fixed	Moving	Moving	Moving	Fixed
Knowledge Distribution / Configuration	Good	Bad	Bad	Bad	Bad

Table 6 - Characterisation of case study problems.

Table 6 clearly demonstrates that from the sample taken the majority of the problems issues (be they complexity issues or not) are induced within the development processes and are not intrinsic or original to the systems. The only example with clearly intrinsic complexity is that of design optimisation.

The development process in the cases explored here is always the area within the product lifecycle where the complexity or problem issues originate from. In these cases here there have been several problem issues that have arisen within the operational phases of system development, but have always been caused by developmental failures. In every case here, the development activity is deemed the root cause of the complexity issues. It is important however to appreciate that this is not e definitive set of problems, and this problem category set will expand as the research continues.

The information flow, for the most part is continuously moving, and when coupled with a knowledge distribution or configuration that is 'bad', the overall understanding of the system is low. Low understanding of the component parts of the system is termed ignorance, and should not be confused with problem exhibiting high complexity.

Does this mean that the knowledge distribution or configuration, coupled with the moving information is the direct cause of the complexity within these problems? Is this the inducing factor within the development process (all of these problems are as a result of a development lifecycle failure) that leads to the problem issues occurring as described?

Table 7 shows the coping methods or solution options available for the different case study problem issues decomposed within Table 6. Like Table 6 the solutions for each problem issue are decomposed within a set of criteria, and these criteria are outlined below:

Evaluation / Visualisation – Classifies the problem solution as sitting within either detailed problem evaluation or visualisation processes or tools.

Numerical Analysis – Ascertains if numerical analysis or numerical approaches are likely to provide a useful insight to the problem issue described.

Process Efficiency – Determines if an improvement in development process efficiency will provide a solution to the problem issue.

Improved Architectures – Ascertains if changing the structure of the product, organisation or product support mechanisms will in fact reduce the problem effect.

Focus On Complex Area – Establishes if an increased focus on the area of complexity will enable a solution to be developed.

Preview Problems – Determines if the preview of the problem before it has occurred using tools such as synthetic environments likely to provide insight to the problem before it occurs.

	Design Optimisation	Immature Requirements	Organisational Change	Pre-Production Induced	Organisational Culture
Evaluation / Visualisation	X	X		X	
Numerical Analysis	X				
Process Efficiency		X	X	X	X
Improved Architectures	X		X		X
Focus On Complex Area		X		X	
Preview Problems	X			X	

Table 7 - Coping methods from case studies.

Table 7 shows a representation of the solutions that were implemented, or in some cases could have been implemented to reduce the complexity of the problem issues. Each problem has been allocated a potential solution, or solutions within the categories described above.

Organisational change and culture both seem to employ similar coping mechanisms, there may be a link here, but it is difficult to understand the nature of that link using just these tables as the detail is not here. The lack of numerical analysis as a potential solution to most problems (apart from that of design optimisation, and even then it might not always be appropriate) could suggest that most problems are non-numeric in nature. Whatever the problem appears to be, in most cases improving the architecture, or process efficiency is the mitigating factor which should reduce the problem issues within the development processes.

These tables, although a useful overview of problem issues within the case studies, do not provide a detailed enough basis to understand the exact nature of all problems and the effect of the proposed or implemented solutions. Further analysis is required giving a detailed look at classifying these problems and solutions more accurately, while continuing to collect different case study problems and solutions. Links need to be established between problems and their prospective solutions, and in some cases links between comparable solutions for problems, for example the link between organisational change and culture, as solutions both seem to tackle process efficiency and architecture changes.

4.10.2 Links between the CCCS and the Case Studies

Figure 25 shows a CCCS (section 2.10) and case study representation, and the links between the two based on the information contained within the case studies in terms of the problem causes, nature of the organisation and the coping mechanisms.

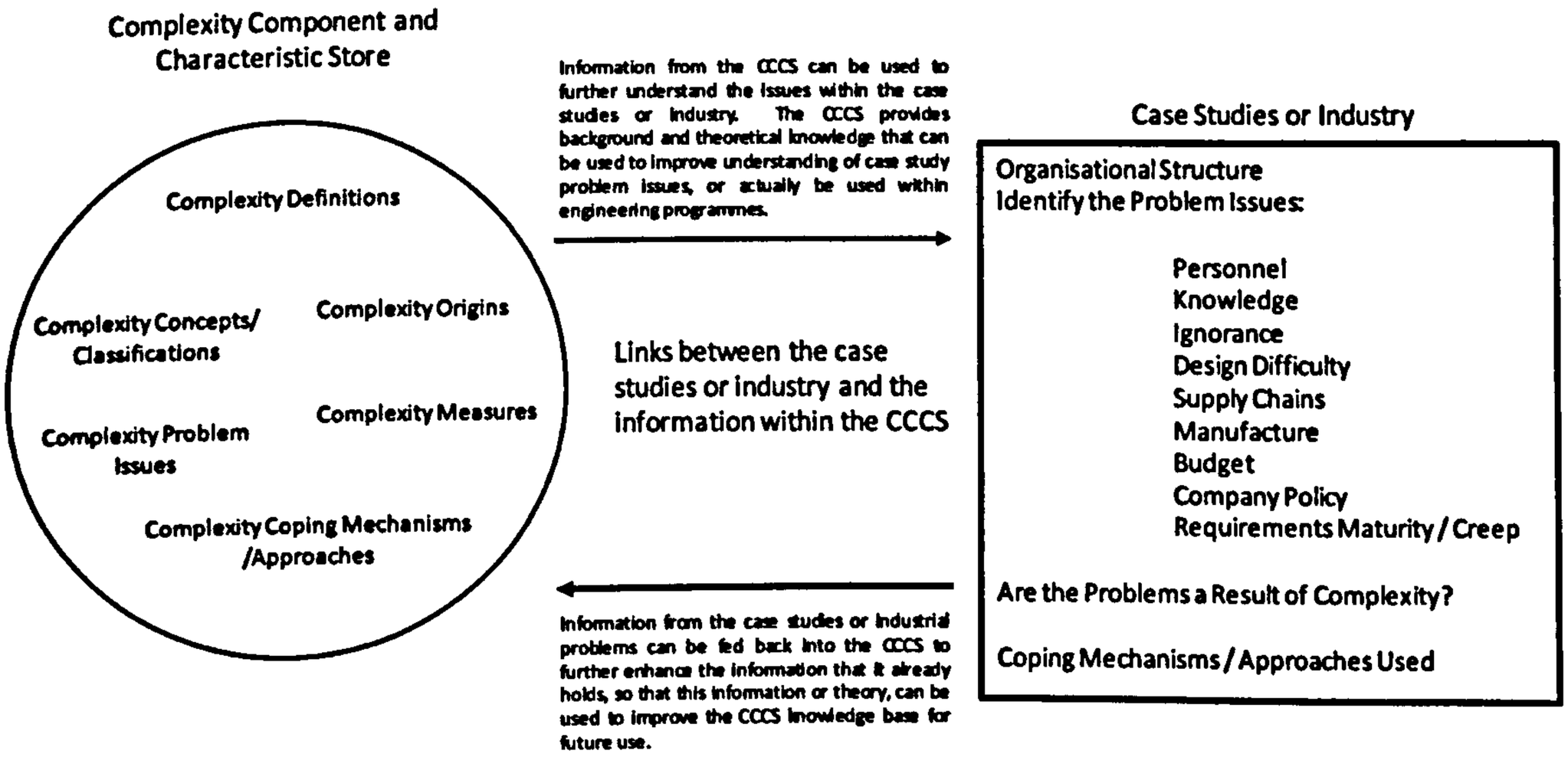


Figure 25 - Component model and case study links.

Table 8 shows in more detail how each of the case studies can be related to components within the CCCS.

Complexity Characteristics	Design optimisation	Immature Requirements	Organisational Change	Pre-Production Induced
Is the complexity intrinsic of induced?	Intrinsic to the system; in most cases as variables are run through a series of optimisations to achieve the most favourable system performance.	Induced by the development process; the requirements are never fully matured, but engineering work must begin early in order to meet timescale and budget targets.	Induced by the changes within the organisation; organisational change makes knowledge, and skill management difficult to track and internal policies limit communication.	Induced by the development process; due to reduced funds and effort in the manufacture planning stages, resulting in custom built systems.
What complexity definitions are appropriate?	Irreducible, intricacy and coupling; the system is highly intricate, and coupled in terms of the relationships of the variables.	Level of understanding; the requirement understanding is low, and the development process is hindered as a result.	Lack of understanding; the change in the organisational configuration and communication limitations reduce the overall understanding of the system.	Indescribable; the system is not easily described (plans, drawings) and manufacture is hindered.
Which classifications or concepts of complexity are appropriate?	Non-Hierarchical; the system variable interactions are often very intricate with high coupling, but static.	N/A – no classification can be directly mapped to the requirement maturity issue.	Complex Adaptive; the development capability is constantly changing, and it is difficult to operate robust processes within the organisational flux.	N/A - no classification can be directly mapped to the pre-production induced problem issues.
What complexity measures are useful?	System specific measures; the number of elements and interfaces, and the intricacy of the links.	System specific measures; measuring the level of maturity, and stability within the requirements (requirement change metrics).	System specific measures; measuring the deployment of organisational skill, and resource allocation, along with comprehensive measures of communication.	System specific, and randomness measures; measuring commonality within the system as a result of the custom manufactures, assessing the level of customisation.
What are the complexity origins?	Intrinsic origins; the level of cross fertilization of information, the number of interfaces, and components, and the intricacy of those interactions.	Induced origins; the internal polices, and external polices of partners or customers forcing design due to pressing timescales despite requirement immaturity, and evolutionary requirement tendencies due to the customer wanting additional functionalities.	Induced origins; the organisational structure in flux causing resource and skill allocation difficulties, and information sharing barriers while system development is taking place.	Induced origins; the mindset of those working on the project not appreciating the manufacturing of the product early within its development.
What are the effects of the complexity?	Engineering processes; the difficulty of the optimisation slows the engineering process.	Engineering processes and the organisation that creates the system; the need for quick development early means immature requirements are used within the development processes and results in; the need for re-work, possible non-compliance, and prolonged development.	Engineering process; the constant flux in the organisational configuration has the potential to effect knowledge and the speed of the development along with timescales, and budgets.	The product; if the manufacture stages were not matured the subsequent products are all essentially custom builds creating; added maintenance costs, poor reliability, systems that are very difficult to support,
Complexity coping mechanisms that could be or were used?	N2, DSM; to understand the inter-relationships.	Limiting the project scope removing difficult requirements to mature.	None.	More detailed pre-manufacture plans.

Table 8 – Example mappings of complexity characteristics to the typical problems within industry.

Essentially these links provide the basis from which to determine if a problem is a complexity problem or not. However for the purpose of the research, all problem

issues are tackled and there is no distinction between complexity and non-complexity problem issues.

4.11 Summary

The case studies identified some key and common problem issues when developing large systems. These common problems fitted into five common categories: Pre-Production Induced Issues, Organisational Culture Problems, Organisational Change Issues, Design Optimisation Difficulties and Immature Requirements. The majority of these problems were induced by the development processes within the organisation, and not intrinsic to the systems.

Like the problems, the approaches for dealing with those problems also fitted into categories; Evaluation / Visualisation, Numerical, Process Efficiency, Improved Architectures, Focus On Complex Area and Preview Problems. These were applied within some of the case studies analysed.

Understanding the common problems, their causes and characteristics, along with prospective solutions, links can be made with the information collected and held with the CCCS (a store of complexity information from the literature review and case study findings). These links to the CCCS, and perhaps between items of the CCCS improve the understanding of complexity within systems and can further enhance the ability industry has when dealing with it.

Figure 26 shows the flow of information found within the case studies within this chapter enables the creation of the CCCS within chapter 5.

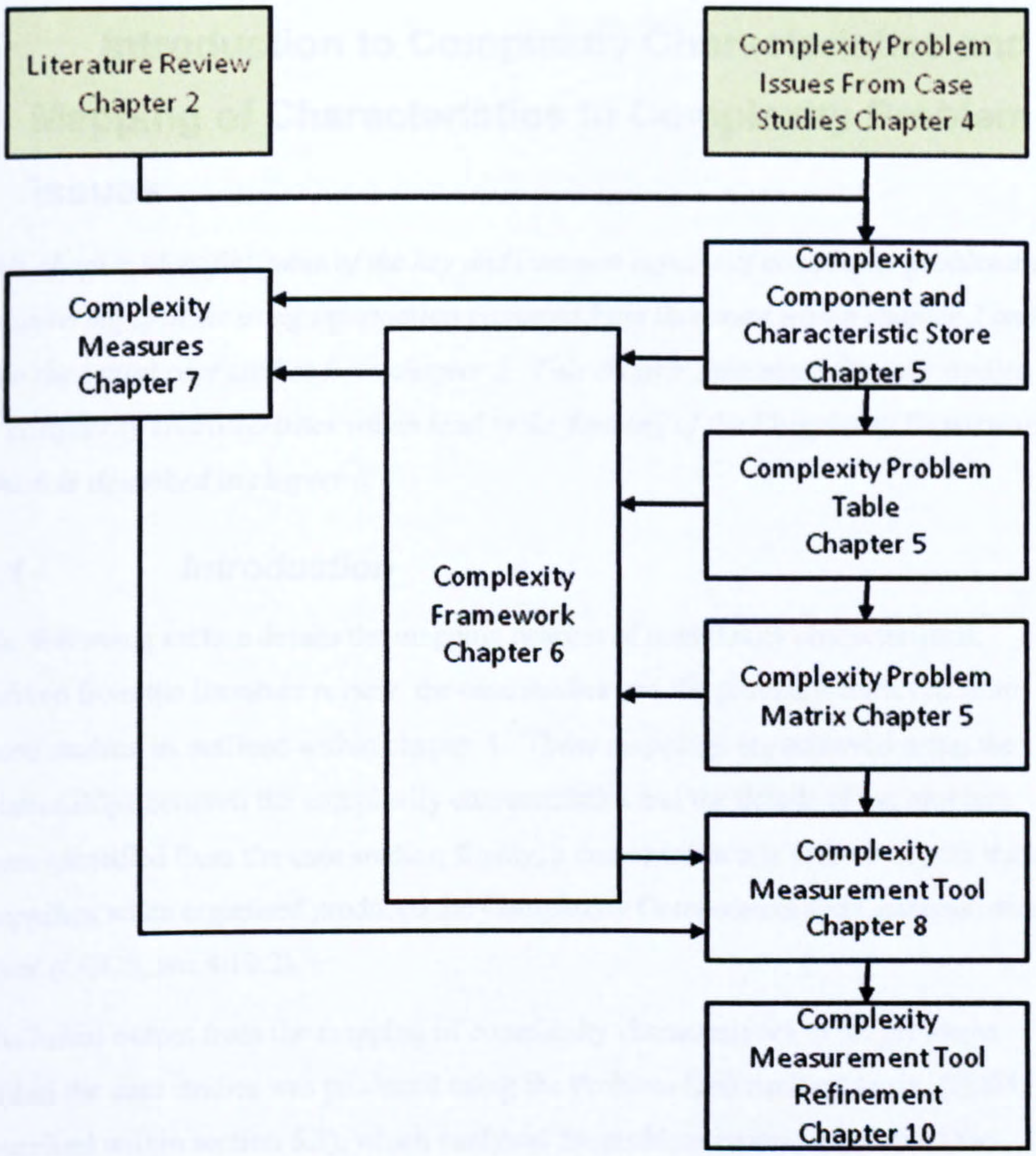


Figure 26 - The layout of the work and the thesis outputs roadmap.

5 Introduction to Complexity Characteristics and Mapping of Characteristics to Complexity Problem Issues

This chapter identifies some of the key and common aspects of complexity problems in engineering systems using information gathered from literature within chapter 2 and also the initial case studies from chapter 3. This chapter then maps the case studies to complexity characteristics which lead to the forming of the Complexity Framework which is described in chapter 6.

5.1 Introduction

The following section details the mapping process of complexity characteristics, derived from the literature review, the case studies and the problems retrieved from those studies, as outlined within chapter 4. These mappings are achieved using the relationships between the complexity characteristics and the details of the problem issue identified from the case studies; finally, a demonstration is offered of how these mappings when organised produced the Complexity Components and Characteristics Store (CCCS, see 4.10.2).

The initial output from the mapping of complexity characteristics to the problems within the case studies was produced using the Problem Description Matrix, (PDM, described within section 5.2), which analysed the problem issues and enabled the Complexity Matrix (CM, see section 5.2) to be produced. This matrix provides the detail of the relationships of various complexity characteristics to engineering problems. The complete Complexity Matrix output can be seen within 14 Appendix A – First Case Study Set Details.

5.2 Introduction to the Complexity Matrix and the Problem Description Table

The complexity matrix (see section 5.2.1) is a view of problem issues in relation to the Complexity Components and Characteristics Store, allowing the problems to be deconstructed in relation to the complexity elements found within the Store. The matrix uses three quantitative measures based on team size, interactions and skill sets contained within the situation when the problem has occurred, and then the result after

a solution has been implemented. The problems are then examined further individually, using a matrix of development stages or activities within which the problems reside, which enables a further detailed description in relation to key engineering processes or factors, thus furthering our understanding of each issue.

The Problem Description Matrix (see section 5.2.3) is a further attempt to develop the understanding of the industrial problems presented by the case studies from a systems engineering lifecycle perspective. The matrices identify links between problems within a case study, and different elements within the engineering lifecycle. The matrices broaden the understanding of the problem issues, in order to make it easier for those problem issues to be related to complexity characteristics or components.

5.2.1 Complexity Matrix and Analysis

The complexity matrix is a Microsoft Excel based spreadsheet. The columns for the Complexity Matrix are taken from the CCCS (see section 4.10.2), in order to map effectively problem issues to complexity components or characteristics.

The first pages include the details surrounding the case studies and an interpretation of the problem issues within each case study. Figure 27 shows a portion of the descriptive section to the first table. Within it, there is an extract from a case study detailing the different problems identified, (numbered LA01 to LA06) the problem description, nature of the complexity within it, the driver that caused it, and where it sits in the product lifecycle. The same is repeated for every case study example analysed.

Problem Decription And Comments						
Programme Case Study	Brief	Problem Number	Problem	Nature of Complexity	Driver For Problem	Lifecycle
Lancer	Competition between Bae and GEC before the merger, when the two companies were forced to maintain the boundary split and isolation in order to maintain the competition environment. The systems involved were Lancer (this case) and Sika. The programmes produced two prototype land recce vehicles. The two systems would compete for the FRES and FCS programme options for the Recce vehicle.	LA01	Overfamiliarity between the members of RO defence and C4ISR due to the close proximity meant that the processes were often relaxed and documentation and interface specifications were not always kept up to date.	No, this is just a result of putting two sets of people that already have a level of familiarity in an evnironment where they were allowed to mix and control was not taken over documentation. Incidentally this was not a major problem on the programme. But it did cause a lack of common assumptions	Speed agility and the familiarity that already exists between the two business units. Also the proximity of the business units helped as it allowed easy communication, as a result some development process was relaxed.	Problem Occurred In Stage(s Concept phases and design The Problem Affects Stage(then on.
		LA02	Lack of common objectives of business units (BUs) within BAE, means that BUs will always seek to retain work rather than feed it out to other areas of the business that are better equipt.	There are not a large number of interactions it's just a state of ultimate competition internally. A modification of BAE SYSTEMS values may change this approach. Important to realise though that there are advantages to this approach, without it those business	Each business unit has it's own objectives and must achieve those targets.	Problem Occurred In Stage(s Problem Affects Stage(s): A
		LA03	Engine controller failure, the engine controller kept overloading and failing during the trails for the Lancer vehicle. The cause was unknown, everytime engineers approached the vehicle the handbrake was applied and the vehicle swtiched off, there was no obvious cause of the failure of the solid state	The cause of the fault was not difficult to find, and this in itself is evidence to show the problem is not complex.	Lack of understanding of the vehicle controls? Or perhaps a poor man machine interface.	Problem Occurred In Stage(s Development phases when d the concept of use and man interfaces The Problem Affe Stage(s): Testing and design
		LA04	The coiling of the cable for the sensor pack was having trouble retreacting into the hull of the vehicle, due to it's size and weight.	A engineering problem within the design that was rectified.	Need to raise and lower the sensor suite.	Problem Occurred In Stage(s Development and testing ph Problem Affects Stage(s): T Acceptance
		LA05	The map display for the vehicle was out by quite a large margin. The mapping system is used to display the vehicle location in relation to the terrain around it, a mixing of GPS data and surface map data. The surface map is soewed due to the curvature of the earth, essentailly it goes through a flattening.	Was a design decision and fault finding exercise, the problem was easily identified and corrected.	Lack of understanding of the mapping systems used and the lack of sync between the gps and ordinance maps.	Problem Occurred In Stage(s Development stages (unkno point) The Problem Affects Testing
		LA06	ESIL, CSIL integration process not	This is simply a choice not to follow	Reducing cost and timescales.	Problem Occurred In Stage(s

Figure 27 - Complexity matrix problem description and analysis example.

Links to the CCCS are identified in the second portion of the table. Figure 28 shows the mapping to the CCCS categories (heading 4.10.2) within the table.

Problem Decription And Comments			Component Model Mapping			
Programme Case Study	Brief	Problem Number	Problem Issues	Complexity Definitions	Coping Mechanism	Coping Measures
Lancer	Competition between Bae and GEC before the merger, when the two companies were forced to maintain the boundary split and isolation in order to maintain the competition environment. The systems involved were Lancer (this case) and Sika. The programmes produced two prototype land recce vehicles. The two systems would compete for the FRES and FCS programme options for the Recce vehicle.	LA01	Culture and over familiarty between business units. The lack of a formal contractual link between business units causing specifications to be neglected and increase workloads later on in development.	None that I can think of apply to this problem.	To my knowledege this was never addressed specifically, but the programme did complete and the product was sucessful.	Continue working with the ov familiarty between the two bus units. Cope with the effects a occur.
		LA02	A generic problem across all companies, the problems are lack of common direction, usually a reduction in the overall profitability of work, no optimisation for business units for their line of work.	Perhaps a complexity definition suits this as it is a set of organisational entities that are exhibiting independent behaviour while interacting and the effect on the whole may not be obvious?	There was no coping mechanism introduced to deal with this issue.	I am not sure how this is cope even if the business units sho with it as such? Perhaps this whole nature of the problem, it's nature it is not coped with within the development lifecy
		LA03	Engine failure during testing resulted from missuse.	Fault finding, cause and effect difficult to asscertain perhaps.	Fault finding was the coping mechanism that was employed here, it was not an obvious problem, later it came to light that the handbrake was being left on while the vehicle moved as there were no indications within the vehcile it was applied. Thus the vehicle was moving with the brake applied an	N/A
		LA04	Difficult to ensure safe raising and lowering of the sensor suite.	I can't think of any that apply?	An external contractor was employed to re-visit the problem and a device was developed that ensured the coiling of the cabling was done as the sensor pack was retracted.	Employ external contractor to the problem.
		LA05	The lack of understanding of the GPS system mapping, and it's incompability with conventional mapping producing a vehicle location that is not accurate.	Not sure I can think of an application of a complex definition.	There are standards that are employed to compensate for this effect WGS84 is one of these standards. The problem was solved by applying this standard to all the mapping systems of the vehicle and obtaining 2d surface maps that were also WGS84 compliant. This solved the problem and the	Rework the problem and repa

Figure 28 - Links between the CCCS and the case studies.

For each component model heading, the problem is broken down and the most appropriate information within that category from the component model is populated

for the problem. In some cases a direct map to information within the component model is not possible for the category being analysed, so not all aspects of this table are completed. This also provides a feedback to the component model to identify any obvious gaps within it which need population.

5.2.2 Basic Complexity Problem Metrics

Three basic qualitative measures were developed to assess the case studies outlined within section 4. These measures were an estimation, produced in order to see how measures could be implemented in a meaningful manner, if at all, within the complexity matrix. The measures assess the problem, not the system within the problem; that is to say, the interdependencies, skill variety and number of people involved in the problem, which are not necessarily part of the actual product system.

These measures were based on estimation of a qualitative nature and assessed interdependency, skill sets required and the size of the organisation involved. The measures were created in this manner as the data available from the case studies was not sufficient to make a detailed assessment of the complexities that were involved, and these measures covered aspects of complexity covered within the definition for the thesis, namely intricacy and interdependencies, variety or commonality of skills, and a scale fact based on the organisation size. The qualitative measures are defined as follows:

Interdependencies Between Different Factors (IBDF) - This is a simple qualitative scale between 1 and 10 and is purely estimation, with no specific semantics for measurement definitions. The measure estimates, using knowledge of element numbers, interface numbers, an understanding of the interface intricacies and strengths the potential of the interdependencies within the problem under analysis. The higher the interdependencies within the problem, the higher value for the measure will be.

Number of Skills Involved (NSI) - This is a simple qualitative scale between 1 and 10 and is purely estimation, and not based on any defined semantics. The measure is based on the number of disciplines electrical, mechanical, software, aerodynamic, hydraulic, etc and the assumption would be that the number of skills used (the diversity) would be proportional to the complexity

of a system. The higher the number of skills, the higher value for the measure will be.

Persons Involved in Analysis (PIA) - This is a simple qualitative scale between 1 and 10 and is purely estimation, and not based on defined semantics. The measure assumes that the size of the workforce and the need for interaction within the large workforce will be proportional to the complexity within the problem. The larger the organisation that is involved in the problem the higher the probability for complex issues.

If these measures were to be used in the future to assess problems within the complexity matrix, it will be necessary to set further constraints to the selection of the value for each of these measures for any problem or solution situation. Semantics will need development for this purpose, for the initial stage of the research the estimation approach is sufficient to evaluate the approach. However for the purpose of this analysis, the qualitative estimation approach is sufficient given the information available regarding these problem issues.

		Scoring Problem			Scoring Solution			Coping Solution Complexity Affect		
Problem Number	Applicable Approaches	IBDF	NSI	PIA	IBDF	NSI	PIA	Percentage Change	Solution Affect	Comments on Scoring
LA01		5	6	6	5	6	6	100%	Complexity Remained The Same	There was no clear solution so the co the problem did not change.
LA02		3	3	3	3	3	3	100%	Complexity Remained The Same	There was no clear solution so the co the problem did not change.
LA03		2	2	1	1	1	1	60%	Complexity Decreased	Fault finding reduces the problem to where it is simplistic.
LA04		2	2	1	2	1	1	80%	Complexity Decreased	Introduction of an external contractor number of people inwovled, and redu number of interactions as the work is concentrated.

Figure 29 - Matrix scoring mechanism example.

The scores are then totalled, and the percentage change of the complexity within the problem after the solution has been applied is calculated as a rough guide. If the solution that was imposed has in fact increased the complexity of the problem then the percentage displayed will be over 100%. The solution effect column displays if the complexity has gone up or down and the comments column is a comparison between the complexity values that were produced, and what was expected as validation. If

these values show a small decrease in complexity, and one would expect a large decrease, then there may be a inaccuracy with the metric representation.

It is necessary to correlate the information provided by the metrics, despite only being estimations of a qualitative nature. These correlations will confirm if the measures, despite their definitions being different, are in fact measuring different aspects of the problem. High correlations between two measures would suggest that they are in fact measuring the same effect within the problem or solution, and if this is the case both measures may not be necessary.

Correlation Table Problem

Correlation	Correlation Data
IBDF With NSI	0.478938103
IBDF With PIA	0.373899298
NSI With PIA	0.121271296
IBDF With SUM	0.851511905
NSI With SUM	0.663921673
PIA With SUM	0.702230524

Correlation Table Solution

Correlation	Correlation Data
IBDF With NSI	0.36633016
IBDF With PIA	0.59807754
NSI With PIA	0.271878826
IBDF With SUM	0.838900042
NSI With SUM	0.68505566
PIA With SUM	0.814588704

Figure 30 - Correlation of metric values for the problem and solution situations.

Figure 30 shows the correlations between the set of values for each measure in both the problem and applied solution situations. Correlations between the measures IBDF and NSI for the problem domain seem to be a higher than expected, but this same correlation is not so high for the applied solution domain, which would suggest the measures are in fact independent. The other correlations for the problem domain (ignoring the total correlations) are generally low, suggesting they are in fact highly independent. The high IBDF, PIA correlation within the applied solution domain suggests there might be a dependency here, but this is not the case with the problem domain so both measures are still valid overall.

The size of the data set (below 30), will limit the accuracy of these correlations, however it would be fair to say that for the majority of systems that exhibit high complexity within the case studies, most exhibit high values for 2 of the 3 measures, for example problem TB04 has values of IBDF 10 and NSI 8, with a low value of 1 for PIA, and IP01 has high values of 5 and 6 for IBDF and PIA with a low value of

just 3 for NSI. This will have an effect on how the correlations interpret the results, but for a test purpose these metrics as estimates are suitable, for detailed analysis the metrics will need to be re-developed.

5.2.3 Problem Description Table

An understanding of the problem issues is assisted by a Problem Description Table. The table contains various categories along the top and side which correspond to elements and phases of a product lifecycle using the systems engineering lifecycle guides (Haskins 2006) as a basis and other aspects which may be important in development programmes (as they appeared within the case studies), such as sub-contracting, manufacture, and budgets.

The links are established between the relevant elements for the particular case study within the table and justified within a comment and contain the assignment code shown within Table 5 above, so they can be easily traced back to their originating case study. These links are established for the problem domain (denoted by the yellow area, see Figure 31) and applied solution domain, if a solution or action was implemented (denoted by the green, area see Figure 31). An example of a link is as follows:

LA01: The sub-contracting between the two business units meant that it was not strictly controlled due to familiarity and close proximity. As a result links can be found in the problem domain: Current Organisational Structure and Timescales, Sub-Contracting and Current Organisational Structure, Parallel Development Strategies and Organisational Governance. Since there were no actions taken to attempt to solve this issue, no links appear within the solution area of the table.

(see Table 5, Appendix A – First Case Study Set Details and Appendix B for details)

Problem domain links show aspects of the problem and how the components of the problem link to other aspects of the programme within which the problem is located, or how elements of the problem interact with each other. It is important to realise that the links found within the problem domain will not necessarily be directly mirrored by those within the applied solution domain - they will not be identical. Not every solution applied addresses all the problem domain elements; in addition, some solutions can create additional links that did not exist in the problem domain after they have been implemented.

Once established, these links are then used to further understand and dissect the problems within the case studies, leading to a better appreciation of real and common industrial problems within programmes. This much improved understanding enables complexity characteristics to be more easily associated later on.

5.2.4 Problem Description Table Categories

The follow sections detail the categories used within the Problem Description Table (PDT). They have been grouped into common subject areas consistent with systems engineering lifecycle definitions (Haskins 2006) and concepts of project management (2007, Hastings 1995, Shenhar, Dvir et al. 2001).

5.2.4.1 Programme Characteristics

Programmes have specific characteristics and scopes, the following elements of the PDT attempts to address these as categories to be linked.

Timescales – The timescales with which the programme is running. Problems and solutions may increase the timescales required or reduce them.

Budget – The budget the programme is constrained by, or is allocated. Problems and solutions may increase the funding required to conduct the activities on an overall or isolated scale, or reduce the funding required.

Size and Scope – The scope of the programme overall can be increased or decrease as a result of complexity problems in the development process. Solutions to these increases if they impact cost to highly may reduce size and scope if implemented properly.

5.2.4.2 Organisational

The organisational structures of all supporting elements to the development capability, not just confined to the organisational structure of the prime contracting company, but expanded out to customers, suppliers and the governance that surrounds them.

Current Structure – The current structure of an organisation, or element of the organisation within the development capability or operational product.

Structure Change – A change applied to the current structure of an organisation, or element of the organisation within the development capability or operational product.

Governance – The governance found within an organisation, or element of the organisation within the development capability or operational product.

Sub-Contracts – The sub-contract relationships found within the development capability or operational product.

Customer – The customer organisational structure or an element of it.

5.2.4.3 Human

Some elements of problem issues or solutions involved a change in the human behaviour or perception. These elements are covered within this section of the PDT.

Mindset – The mindset of those within the organisation reflecting the motivation, understanding of the programme, interpretation of success within the programmes and interpretation of the programme goals or objectives.

5.2.4.4 System Requirements

Requirements are a major contributing factor to the problems found within the case studies. The problems stem from the derivation of the requirements, and the level of maturity within the requirements late on in the programmes.

Definition – The level of definition of the requirements within the requirement set applicable to the problem.

Maturity – The level of maturity of the requirement set applicable to the problem.

5.2.4.5 System Design

Elements of system design will obviously influence the problem issues that will occur within a programme. In particular interfaces related to the product, can contribute to the complexity within the programme.

System Internal Interfaces – The interfaces internal to the system (the product under development) boundary.

System External Interfaces – The interfaces external to the system (the product under development) boundary.

Legacy Equipment – Use or understanding of legacy equipment within the system under development.

5.2.4.6 System Testing

System testing is a major aspect of product development within the lifecycle, and will be a serious contributing problem issues.

Planning – Planning for the trails activities that will be used to prove the system is fit for purpose and meets its requirements.

Trials – The trails activity itself.

Acceptance – Having the system accepted as a result of the trails programme completion.

Certification – Having the system certified as a result of the trails programme completion.

5.2.4.7 Development Process

Aspects of the development processes within a programme play key roles within the problem areas found within the case studies. The type of engineering process and the deliverables that must be produced by those processes will be elements of some of the problem issues encountered within programmes.

Concurrent – A concurrent engineering development process implemented within the development programme.

Serial – A serial engineering development process implemented within the programme.

Deliverables – The number of attributes of the deliverables expected from the development process by the customer, can include products, documentation and training.

Rework – The rework required additionally to the standard development process practices.

Parallel – A parallel engineering development process implemented within the development programme.

5.2.4.8 Manufacture Process

Once developed, projects move to the manufacture process aspect of the development lifecycle. This process is critical, developing efficient effective manufacturing

methods, builds towards commonality and thusly better support and reliability for the designed products.

Drawing – The drawing pack updates for the products.

Pre-Production – The pre-production work developing the efficient streamlined approaches to the manufacturing of the systems.

5.2.4.9 Support

The support provided to the customer once the product has completed its development, testing, acceptance and manufacture phases.

Maintainability – The maintaining of the product once the system is in service with the customer.

Reliability – The reliability of the product once the system is in service with the customer.

5.2.5 Problem Description Table Links

Each case study has its own Problem Description Table. Figure 31 shows the PDT for the upgrade programme case study. The yellow area represents the problem links and the green area represents the solution links between the categories (see 5.2.4.1 to 5.2.4.9). Each problem was analysed in terms of the links between different categories and their links identified for both the problem domain and applied solution domain within each portion of the table. Additional comments were added to justify the link for that particular problem or solution for the categories.

		Problem Causes								
		Programme Issues			Organisational					Human
		Timescales	Budget	Size and Scope	Current Structure	Structure Change	Governance	Sub-Contracts	Customer	Mindset
Programme Issues	Timescales				UP01					
	Budget									
	Size and Scope									UP05
Organisational	Current Structure									
	Structure Change	UP01			UP01					
	Governance									
	Sub-Contracts					UP01				
	Customer									
Human	Mindset									
Requirements	Definition									
	Maturity									
System Design	System Internal Interfaces									
	System External Interfaces									
	Legacy Equipment									
Testing	Planning									
	Trials									
	Acceptance				UP04					
	Certification				UP04					
Development Process	Concurrent									
	Serial									
	Deliverables									
	Rework	UP02			UP02					
	Parellel									
Manufacture Process	Drawing									
	Pre-Production									
Support	Maintain									
	Reliabilty									

UP01:
The current setup meant that the internal business unit was treated like an external sub contractor, which constrained progress. Due to the programme being strucutred in this manner, timescales were extending.

Figure 31 – Problem Description Table example showing comment field for solution link placement.

It is important at this stage to remind ourselves again that there is not necessarily a direct link, or mirroring between the problem and solution domain links. In some cases solutions may have only tackled some elements of the problem in order to reduce affecting other aspects such as timescales or budget, in some cases no solution was implemented at all.

Overlapping each PDT for each case study enables commonality between the problem issues to be explored, and those areas which are less critical. This data produced a frequency for the occurrence of the various problem and solution links. A simple frequency was not sufficient, as some links are more significant than others and so a weighting factor was added to each link depending on the perceived level of complexity based on the three basic qualitative measures PIA, IBDF and NSI calculated within the Complexity Matrix.

In order to produce a more realistic result, which shows the actual level of complexity within the links as a sum rather than the occurrence of a link, the complexity metrics must be used.

The sum of the complexity metrics for the problem domain, are spread equally among all the problem domain links for that problem issue (of which there are several per case study). The same is done for the solution domain using the solution metrics for that problem after the solution has been implemented. This gives a spread of the complexity within each problem for each case study, however this is still limited as some links within the problem or solution may have more criticality than others, this is not represented within the link matrix and work should be carried out to include this additional feature to get a more accurate representation of the complexity spread.

For the purpose of evaluating the potential for this as a method of understanding and breaking down problems, all the links and complexity values for all case study problems were placed within two summary link matrices, the first showing the frequency of the links, the second showing the level of complexity within the links.

5.2.6 How the Complexity Matrix can be used

The Complexity Matrix contains the problem issues from the initial case studies and the relevant complexity characteristics for those problems. The matrix can identify common complexity characteristics for problem types, such as common relevant measures for certain problem types. These common links can then be used to identify problem types (or potential problems) from complexity characteristics.

Mapping problems to complexity origins, complexity definitions and complexity concepts improves the understanding of why, where and how complexity problem issues manifest themselves. Understanding the problem and its components enables better solutions to the problem, complexity approaches and potential complexity measurements to be selected.

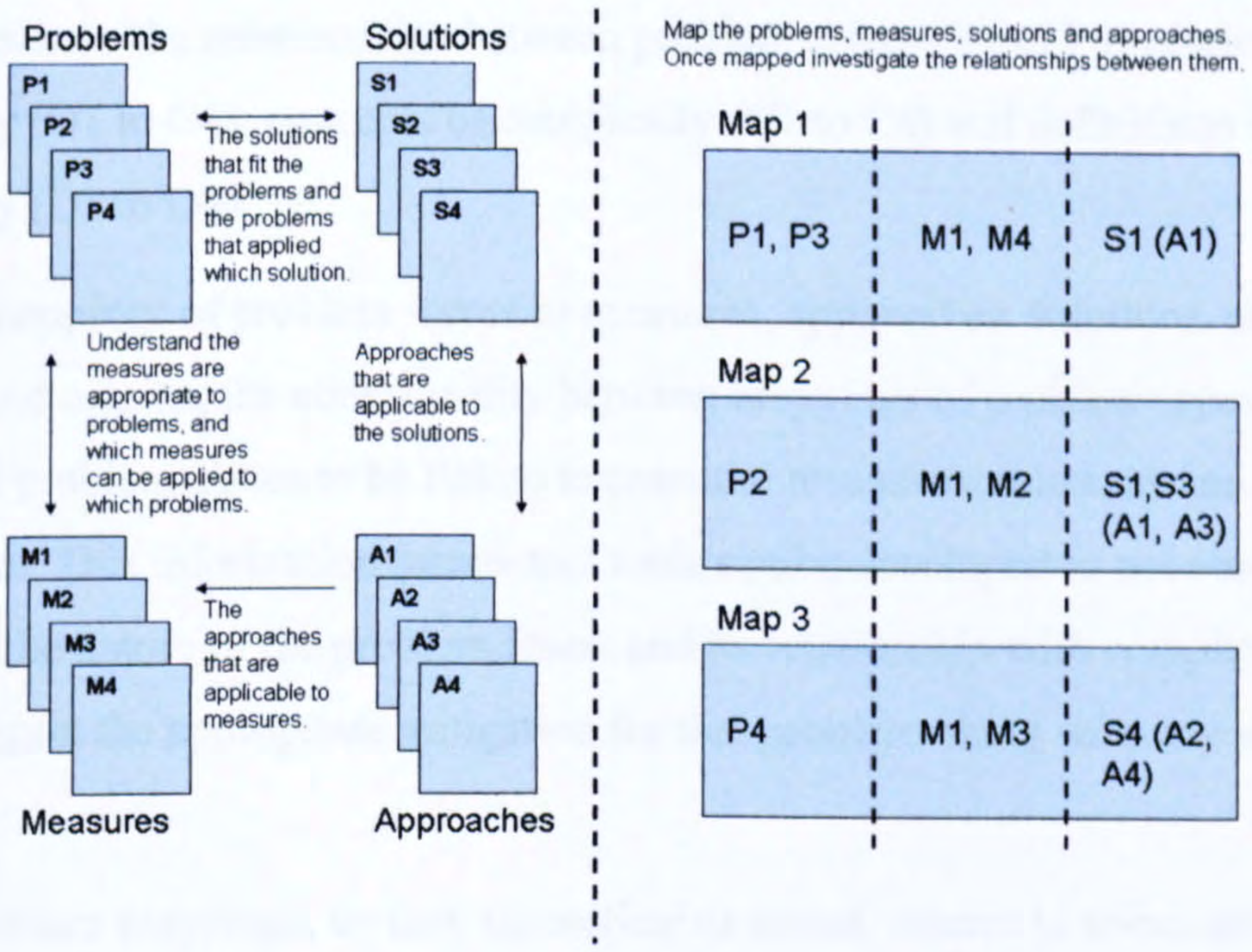


Figure 32 - Systems engineering complexity understanding.

Figure 32 shows the mapping of solutions (S1 to S4) approaches (A1 to A4) and measures (M1 to M4) to problem issues (P1 to P4), in this case P1 and P3 despite being different problems are measured in the same way and employ the same solution and approach.

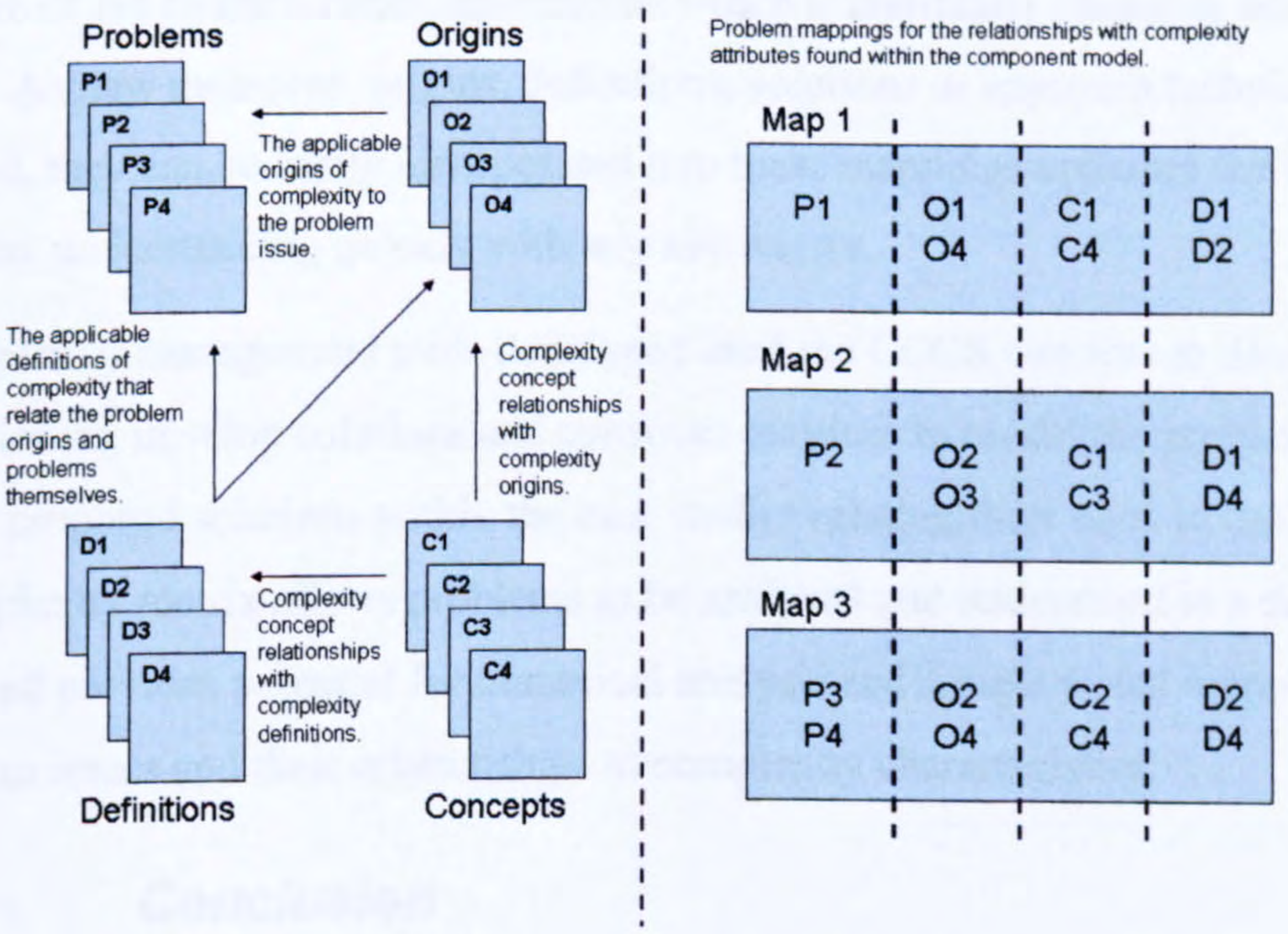


Figure 33 - Problem issues linked to definitions, concepts and origins of complexity to produce mappings.

Figure 33 shows the relationships between problem issues (P1 to P4) origins of complexity (O1 to O4), concepts of complexity (C1 to C4) and definitions of complexity (D1 to D4).

With the mappings of problem issues to measures, approaches, solutions, definitions, concepts and origins, the commonality between mappings of problem types enables theoretical problem issues to be linked to potential measures, solutions and approaches. This information means that tools can be developed to not only determine the nature of the problem issue, and its relationship with complexity, but also to suggest the appropriate mitigation for that problem along with any appropriate metrics.

All the problem mappings, be they theoretical or actual, related to solutions or origins, use the information contained within the original complexity characteristics data. The mappings are then fed back into the complexity characteristics data as an addition to the problem characteristics information. This additional information is then used to compare new problems that are linked with already existing data.

The advantage of developing models, processes or analysis tools in this manner is the live nature of the characteristics information which is constantly changing and being updated. As new measures, origins, definitions, solutions or approach techniques are developed, they can be easily incorporated into these mappings updating the whole complexity understanding quickly with any new views.

The complexity management tools developed used the CCCS contents to deconstruct problem issues, develop solutions and construct matrices to model the problem issues and their proposed solutions within the case studies relating them back to the CCCS. The complexity matrix allows problems to be analysed and understood in a simple format, and provides potential for numerical analysis and simple visual representation of problem issues and their relationships to complexity characteristics.

5.3 Conclusion

The complexity matrix enabled an understanding of complexity in problems and the common elements, components or characteristics of complexity that exist within these problems. Using the matrix it was possible to characterise the systems looked at within the case studies in terms of complexity characteristics that they may contain. This method could then be used to predict problems that may occur in the

development or operation of that system if the relationships with complexity characteristics were similar. The matrix also has the potential to suggest solutions or approaches if these approaches or solutions had been used in the past and subsequently stored within the CCCS. As the matrix increases in size, the ability of the matrix to help in assessing systems increases.

The complexity matrix provided a basic awareness of complexity in systems, it shows the what (which complexity concepts, or definitions), the why (the complexity causes, origins), the how much (potential measurements that are applicable) and what can be done (solutions that can be implemented).

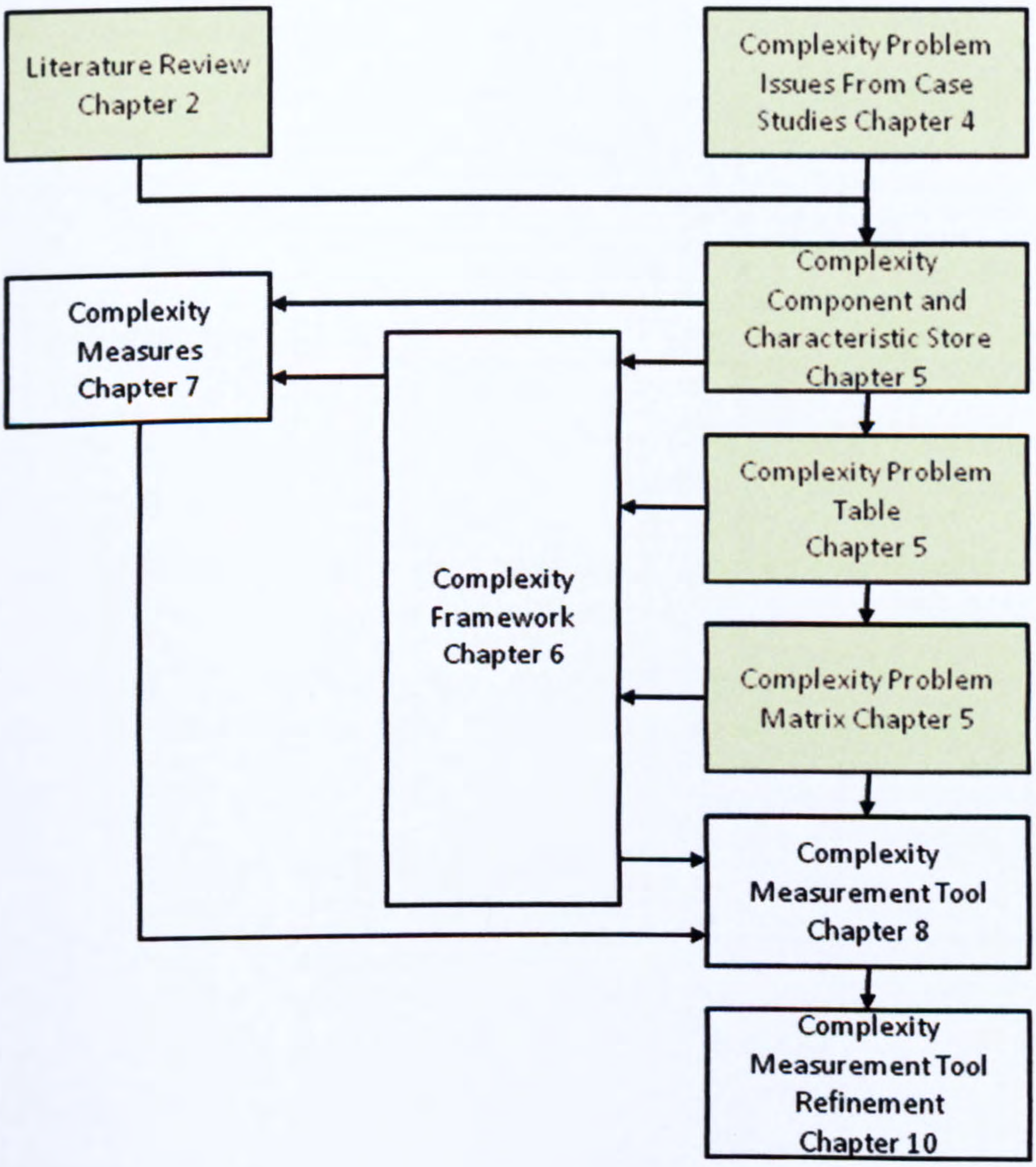


Figure 34 - The layout of the work and the thesis outputs roadmap.

The CCCS is produced using the case study outputs and also the information contained within the literature review. The focus at this point was to understand complexity within problems in more detail. This lead to the development of the Complexity Framework found in chapter 6.

6 The Characteristics of Complexity and their Inter-relations: A Complexity Framework

This chapter examines the relationships between various complexity characteristics. These characteristics were determined from the literature search in chapter 2, the case studies and the interactions within the case studies of complexity components in chapter 4. The output is a framework of complexity which can be used as an aid to creating various measurements, coping mechanisms or management approaches for complexity in engineering'

6.1 Introduction

The following section takes a look at the various components, or characteristics of complexity that have been collected from case studies and from the literature review, and combines these with concepts, measures, approach mechanisms and definitions of complexity.

The relationships between these different elements of complexity are important when developing an understanding of complexity for engineered systems. The section first looks at the different aspects of complexity and then discusses their relationships and the implications of these relationships.

Beyond just the characteristic categories are the actual characteristics themselves explored within the literature review (2 State of the Art Review on Complexity, see section 2.1), different definitions support different concepts, different measurement approaches are consistent with different problems. These relationships are explored within this section.

6.1.1 What are the Inter-relations?

The first set of inter-relations are the interactions between complexity characteristic categories. These form the basis of the inter-relations between the different specific characteristics (specific measures, specific origins, etc). The second set of inter-relations are those that exist between specific cases within each category, the links between different concepts of and origins of complexity, the links between definitions and problems relating to complexity. These links improve the understanding of complexity within systems and further enhance the understanding. Since the list of

specific characteristics is growing, the understanding of complexity is constantly enhanced with further links between specific parts.

6.1.2 Why are the Inter-Relations Important?

The inter-relations between the different complexity characteristic categories provide the basis from which specific characteristics can be linked. Problem issues can be linked to complexity origins and definitions, and those in turn linked to measures, concepts and coping mechanisms. The understanding of these linkages enables approaches or coping mechanisms to be selected for problem issues, and the reasons for that selection is traced back through the other complexity characteristics such as origins, definitions and measures.

6.2 Complexity Characteristic Relationships

The interactions between the different complexity characteristics within the Complexity Characteristics and Component Store (CCCS) are defined within Table 9. The relationships between the different information categories, and also between the problem issues found within industry help to relate items within the CCCS to their relevant problems – definitions can hence be related to problem issues, and then, perhaps, definitions to measurements. These measures may then be considered for use within the terms of particular problems.

	Complexity Origins	Complexity Measures	Complexity Problem Issues	Complexity Coping Mechanisms / Approaches	Complexity Classifications / Concepts
Complexity Definitions	Definitions relate to origins in that definitions describe some origins, irreducible, intricacy, etc.	Measures look at the attributes of the appropriate definitions and origins of complexity within the system.	Definitions may describe problem issues; too many interfaces, lack of understanding, etc.	No straight forward link between definitions and coping mechanisms.	Definitions help to describe the concepts and classifications of complexity.
Complexity Origins		Appropriate measures are selected to measure or quantify some origins of complexity.	If the problem is a complexity problem the causes of that complexity create the problem issue.	The causes of complexity in the problem issues influence the coping mechanisms that are chosen to counteract it.	The classification of complexity within a system is usually derived from appropriate origins and definitions.
Complexity Measures			No direct relationship, measures are usually selected as a result of appropriate origins, definitions and classifications.	Although measures or metrics can be used to test if a coping mechanism is successful, they do not help the selection process of a coping mechanism.	Different classifications of complexity may require different measurement approaches.
Complexity Problem Issues				The problem characteristics, and complexity characteristics within problems, will influence the choice of coping mechanism or approach to reduce it.	The classification of complexity is not subject to the problem issue, but the description of complexity that causes that issue.
Complexity Coping Mechanisms / Approaches					The classification of complexity does not really influence the coping mechanism employed to solve the problem.

Table 9 - Relationships between the complexity characteristic categories.

6.3 The Complexity Framework

The following sections outline the details surrounding the relationships between the CCCS complexity categories. These relationships form the basis of the Complexity Framework, which can be expanded and used to further understand complexity in systems.

6.3.1 Specific Complexity Characteristic Relationships

There are a number of specific relationships between complexity characteristics and the problem issues that were identified within the case studies, the details of these linkages can be found within Appendix A – First Case Study Set Details.

The full set of relationships has not been fully explored, that is every individual complexity characteristic related to every other characteristic, either positive, negative or a null relationship, the underlying concepts of the relationships can be related. Identifying the commonality between different specific complexity characteristics in each of the categories will provide common types. These characteristic types can then be inter-related more easily than every individual characteristic.

The following sections detail the types that exist within the characteristics; these types are sub-set groups within which a significant number of the individual characteristics within that category fit.

6.3.2 Complexity Definition Types

The common complexity definition types are as follows:

Complexity Definition Type	Description of Complexity Definition Type
Irreducible	The system is irreducible, it stops functioning as a result of elements or interactions being removed.
Difficulty in modelling and predictability	A common complexity definition is that of unpredictability, and the difficulty associated with modelling the system. There are however obvious links with scale and the intricacy and number of interfaces, as these attributes are those that make the system difficult to model or predict in the first place. The same goes for irreducibility as simplification is not possible when modelling the system and expecting accurate predictions.
Difficult to describe, lack of understanding	The difficulty in describing the system is very similar to the difficulty in modelling the system. Without an adequate description the plausibility of modelling or predicting the system behaviour is vastly reduced.
Scale of interfaces and sub-systems	The nature of the interfaces and the sub-systems in terms of the information type (data, Boolean, mechanical, hydraulic, pneumatic) being transmitted, and the type (data processor, pump, energy generator) of sub-system, along with the size and number of sub-systems and interfaces, along with the level of diversity.
The level of interactions and intricacies	The level of intricacy of the sub-systems is a frequently used definition which describes the level of coupling between elements within the system. The higher the coupling level the more complex the system is and perhaps the more likely unpredictable behaviour or emergence occurs.

Table 10 - Complexity definition types and their descriptions.

As shown in the literature search, a number of complexity definitions fit into one specific definition type, where as others are in fact composites of these definition types.

6.3.3 Complexity Cause Types

When creating complexity cause types (complexity origins), it is important to distinguish between those causes that are intrinsic to the system development and those that are induced.

Those causes that are intrinsic to the system may require a different approach to those that are induced. In effect, the development team must cope with the intrinsic complexity and reduce, through approaches and processes, the induced complexity.

Complexity Causes Type	Description of Complexity Cause Type
Design Optimisation	The multi-discipline design optimisation, consisting of the need to optimise highly coupled and intricate variables within design.
Immature Requirements	The lack of maturity within requirements, perhaps as a result of immature technologies or re-use of legacy equipment. The lack of requirement maturity means the system is not clearly defined, and without a clear definition it is difficult to formulate a design that will be accepted. The ambiguity means there are a lot of potential solutions (a large number of in equivalent descriptions)
Organisational Change	The changes in the organisational structure in terms of the members of staff, the tacit knowledge generated from those staff, also the introduction of mergers between business units introducing new working financial constraints.
Pre-Production Induced	Pre-production decisions or lack of consideration of product manufacture may incur additional complexities as the design may be unmanufacturable. Without adequate pre-production processes, products may all be unique introducing additional complexities when considering the support of products.
Organisational Culture	The change of the mindset of the organisation and culture within it may affect the design processes that develop the product and in turn effect the complexities of the product.
Interface and Sub-System Intricacy	The level of intricacy within the system, perhaps coupled with the lack of commonality and level of multi-disciplinary skills required make the development of the system very difficult.
System Scale	The size and scale of the system, although not a direct and necessary cause of complexity, it is possible that the scale of the product will impact the complexity of that product, if not in a sub-system and interface perspective perhaps in a scope of operation perspective.

Table 11 - Complexity cause types and their descriptions.

6.3.4 Complexity Concepts and Classification Types

The following are complexity concept or classification types:

Complexity Concept / Classification Type	Description of Complexity Concept / Classification Type
Linear or Hierarchical Complexity	A system with a linear hierarchical interface relationship structure between sub-systems. This makes for easy fault finding and system modelling due to reduced numbers of potential paths between sub-systems (spanning trees).
Non-Linear Complexity	A system with non-linear or hierarchical interfaces between sub-systems making fault diagnosis more difficult due to the large number of potential paths between sub-systems (spanning trees).
Adaptive Complexity	A system within which the interfaces between sub-systems change and adapt as a result of environmental influences.

Table 12 - Complexity concept / classification types and their descriptions.

The classifications build upon one another, and could be thought of as classifications of various levels of complexity within systems.

6.3.5 Complexity Measure Types

The following are complexity measurement types:

Complexity Measure Type	Description of Complexity Measure Type
Requirement Complexity Measures	Measures that can be used to measure various complexities within the requirement definition (number of descriptions, number of potential solutions) for products or capabilities.
System Decomposition Complexity Measures	Measures of complexities within the decomposition of systems, identifying commonality within sub-system and interface groups and components.
Interface & Sub-system Complexity Measures	Measures of complexity within interface and the sub-system components of the overall system, both internal complexities of the interfaces and the sub-systems on their own, along with a complete system understanding of the complexities of the interfaces and sub-systems components.
Management Complexity Measures	Measures of complexity within the management arena such as cognitive complexities or mindsets.

Table 13 - Complexity measure types and their descriptions.

The measurement types concentrate on different areas of complexity within systems, there are a number of measures that fit within each category but there are no clear relationships between the different measurement categories.

6.3.6 Complexity Coping Mechanism / Approach Types

The following are complexity coping mechanism / approach types:

Complexity Coping Mechanism / Approach Type	Description of Complexity Coping Mechanism Type
Evaluation / Visualisation	Able to use detailed problem evaluation or visualisation processes or tools (simulation, testing with prototypes) to provide a solution.
Numerical Analysis	Numerical analysis used to cope with the complexity, perhaps from simulation or detailed algorithmic methods to optimise multi-discipline design optimisation problems.
Process Efficiency	Improve the efficiency of various supporting processes to the development, operation or support of products in order to reduce complexity or cope with complexity better.
Improved Architectures	Improve the architectures of products directly, perhaps reducing complexity by improving the efficiency of the architecture (removing unnecessary interfaces or components, combining components, reducing the level of information transmitted between components, creating sub-systems that are less coupled with other sub-systems reducing interdependencies).
Focus On Complex Area	Focus on the complexity area within the product (if there is one) and increase the level of resource available to deal with the problem.
Preview Problems	Review the problem before it occurs, perhaps using simulation or prediction methods is an approach to complexity in systems, unfortunately in some cases complexity problem issues are no predictable, and in fact some definitions of complexity state that complex systems inherently are unpredictable.

Table 14 - Complexity coping mechanism types and their descriptions.

These types are actions taken in an attempt to minimise, reduce or eradicate detrimental complexity within systems. There is one further coping mechanism that is not included within this table, and that is to not initiate any coping mechanism or approach to managing the complexity. In some instances complexity within the system is in fact beneficial to the system.

6.3.7 Complexity Problem Types

The following are complexity problem types:

Complexity Problem Type	Description of Complexity Problem Type
Design Optimisation	Design optimisation of a number of different disciplines, components and variables which in some cases are not easily quantified (shape) make the development of systems an extremely difficult process.
Immature Requirements	Immature requirements, or in some cases technologies within the system can result in additional complexities that must be managed.
Organisational Change	Changes in the organisational structure may mean loss of knowledge, or product understanding. This effects the efficiency of the development and the
Pre-Production Induced	Lack of work carried out in the pre-production phase increases the complexity of the manufacture phase. Without adequate work being carried out initially, manufacture of the systems can be complex, or result in essentially custom builds that lack commonality.
Organisational Culture	The culture of the organisation and their understanding or expectations of the product development have an impact on the complexity of the product. Timescales, budgets and the nature of the product (upgrade, new build) all influence the processes that are used in the development of that product, and these processes and constraints will influence decisions that ultimately effect product complexity.

Table 15 - Complexity problem types and their descriptions.

Problem themes within system development which relate to complexity exist. There are patterns and commonalities between problems caused by complexity that form the categories for the problem types shown here.

6.3.8 Summary of Types

The following table provides a summary of all the characteristic categories and the types within those categories. The inter-relationships between these types form the basis of the understanding of complexity within systems and development programmes.

Definition Types	Cause Types	Concepts and Classification Types	Measurement Types	Coping Mechanism / Approach Types	Problem Types
1. Irreducible 2. Difficulty in modelling or predictability 3. Difficult to describe, lack of understanding 4. Scale of interfaces and sub-systems 5. The level of interactions and intricacies	1. Design Optimisation 2. Immature Requirements 3. Organisational Change 4. Pre-Production Induced 5. Organisational Culture 6. Interface and Sub-System Intricacy 7. System Scale	1. Linear of Hierarchical Complexity 2. Non-Linear Complexity 3. Adaptive Complexity	1. Requirement Complexity Measures 2. System Decomposition Complexity Measures 3. Interface & Sub-system Complexity Measures 4. Management Complexity Measures	1. Evaluation / Visualisation 2. Numerical Analysis 3. Process Efficiency 4. Improved Architectures 5. Focus On Complex Area 6. Preview Problems	1. Design Optimisation 2. Immature Requirements 3. Organisational Change 4. Pre-Production Induced 5. Organisational Culture

Table 16 - Complexity characteristic type summary.

Just looking at the summary of the various complexity characteristic types shown here, it is obvious that there are relationships between them, as some describe very similar things; for example, definition types 4 and 5, which describe complexity in terms of interfaces, sub-systems and their intricacy directly relate to cause type 6 and measurement type 3. If a problem then maps to one of these types, and in this case design optimisation (complexity problem type 1) may map to the intricacy of the interfaces within the system and the number of interfaces and their sub-systems, then a relationships has been formed that flow through to useful measures, concepts or classifications and coping mechanisms.

6.4 The Relationships

This section details the relationships between the various types identified shown within Table 17. These relationships are essential when developing the complexity understanding.

Complexity Characteristic Types	Reference
Definition Types	DT (then number i.e. DT1, DT2, etc)
Cause Types	CT (then number i.e. CT1, CT2, etc)
Concept and Classification Types	CCT (then number i.e. CCT1, CCT2, etc)
Measurement Types	MT (then number i.e. MT1, MT2, etc)
Coping Mechanism/Approach Types	CMT (then number i.e. CMT1, CMT2, etc)
Problem Types	PT (then number i.e. PT1, PT2, etc)

Table 17 - Reference table for the various complexity characteristic types.

The relationships are not direct, that is, problem types are not directly related to measurement types; in fact, problem types are related to causes, definitions and concepts, which are then related to coping mechanisms and measurements.

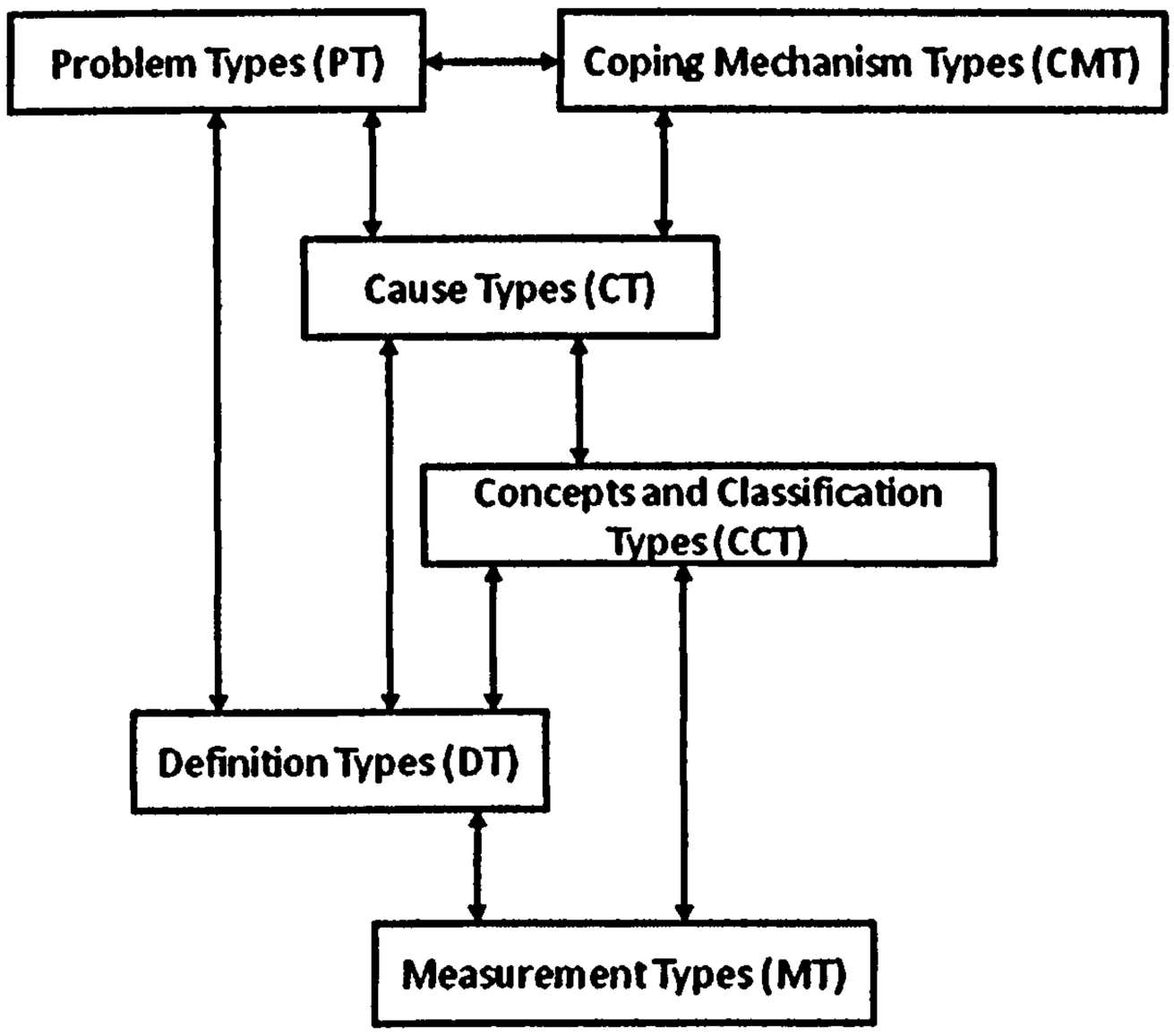


Figure 35 – Complexity Framework characteristic type relationships.

Figure 35 shows the relationships between the different complexity characteristic type categories within the Complexity Framework. Not all complexity characteristic categories relate with each other, some are directly related, and others relate indirectly through other characteristics.

The relationships outlined are shown within Table 18 below.

	DT	CT	CCT	MT	CMT	PT
DT		X	X	X		X
CT	X		X		X	X
CCT	X	X		X		
MT	X		X			
CMT		X				X
PT	X	X			X	

Table 18 - Table of relationships between complexity characteristic type categories.

All the relationships are two way relationships, and in total there are 10 definite links between complexity characteristic type sets, in notation form these are:

DT↔CT, DT↔CCT, DT↔MT, DT↔CMT, DT↔PT, CT↔CCT, CT↔CMT, CT↔PT, CCT↔MT, CMT↔PT

The following sections detail these relationships.

6.4.1 DT↔CT

The following tables detail the relationships between definitions of complexity and the causes of complexity.

Definition Type	Cause Type	Description of the Relationship
DT1	CT7	Irreducibility has a relationship with system scale, the minimum scale of the system possible.
DT2	CT1, CT6, CT7	The difficulty associated with modelling or predicting systems may be a result of a complex design optimisation process, or the scale of the system and the level of interface coupling.
DT3	CT1, CT2, CT3, CT4, CT6, CT7	Modelling and prediction of system behaviour relies on the ability of the development team to satisfactorily describe the system. As a result the relationships to system scale, coupling, intricacy and optimisation are present in this relationship as well as for predictability and modelling. Understanding may also be the result of organisational change or culture, losing knowledge or poor information transfer within the development team. Requirement immaturity is also a lack of understanding of the system, or an incomplete understanding or appreciation for what the system or capability should and should not do.
DT4	CT6, CT7	A direct relationship between the level of intricacy and scale of the system, which can in itself, be a cause.
DT5	CT6	A direct relationship exists between the system intricacies as a definition of complexity and the cause associated with that.

Table 19 - Relationships between the definition types and cause types.

	CT1	CT2	CT3	CT4	CT5	CT6	CT7
DT1							X
DT2	X					X	X
DT3	X	X	X	X		X	X
DT4						X	X
DT5						X	

Table 20 - Summary of the interactions between the definition types and cause types.

6.4.2 DT↔CCT

The following tables detail the relationship between the definitions of complexity and the concepts or classifications of complexity.

Definition Type	Concept / Classification Type	Description of the Relationship
DT1	N/A	There is no direct relationship between the classification and concepts of complexity and the definition of irreducibility.
DT2	CCT2, CCT3	Inherently systems with a non-linear or non-hierarchical structure of interactions between interfaces are more difficult to model and predict. Predictions of systems that are capable of adaptation to their environment become even more difficult still.
DT3	CCT2, CCT3	The same relationship exists here as it does for DT2.
DT4	CCT1, CCT2, CCT3	All the concepts relate to the interface structure and the intricacies, the concepts and classifications describe different levels of interaction and complexities in internal structures.
DT5	CCT1, CCT2, CCT3	The same relationship exists here as it does for DT4.

Table 21 - Relationships between the definition types and concept / classification types.

	CCT1	CCT2	CCT3
DT1			
DT2		X	X
DT3		X	X
DT4	X	X	X
DT5	X	X	X

Table 22 - Summary of the interactions between the definition types and concept / classification types.

6.4.3 DT↔MT

The following tables detail the relationships between definitions of complexity and measures of complexity.

Definition Type	Measurement Type	Description of the Relationship
DT1		There are not really any direct relationships with measures and the level of reducibility within systems.
DT2	MT2, MT3	The modelling and predictability of systems and the difficulty associated is a result of the intricacy of the interfaces and the decomposition or splitting of the system elements.
DT3	MT1, MT2, MT3, MT4	Similarly the difficulty in understanding the system or describing the system is a result of interface intricacy and system decomposition. However requirements are also a factor as they are in fact the description for the system. Difficulty in understanding may
DT4	MT2, MT3	The system decomposition and interface complexity are directly related to the definition of scale within the system interfaces and sub-systems.
DT5	MT2	The level of interaction between the sub-systems through interfaces directly relates to this definition of intricacy.

Table 23 - Relationships between the definition types and measurement types.

	MT1	MT2	MT3	MT4
DT1				
DT2		X	X	
DT3	X	X	X	
DT4		X	X	
DT5		X		

Table 24 - Summary of the interactions between the definition types and measurement types.

6.4.4 DT↔PT

The following tables detail the relationships between definitions of complexity and the problem issues associated with complexity.

Definition Type	Problem Type	Description of the Relationship
DT1		There are no real direct relationships with problem issues and definitions of irreducibility.
DT2	PT1, PT2	The difficulty in modelling or predicting system behaviour could relate to the state of the requirements and any design or design optimisation that must occur.
DT3	PT1, PT2, PT3	The understanding of the system is related to design problems through optimisation and also immaturity in requirement sets which make the product ambiguous. Organisational change is also related to the definition as it can be the cause of loss of knowledge resulting in lack of understanding.
DT4	PT1	The scale of the interfaces and sub-systems is a design optimisation issue.
DT5	PT1	As for DT4, the interactions between the elements of the system and their intricacies are a design optimisation problem.

Table 25 - Relationships between the definition types and problem types.

	PT1	PT2	PT3	PT4	PT5
DT1					
DT2	X	X			
DT3	X	X	X		
DT4	X				
DT5	X				

Table 26 - Summary of the interactions between the definition types and problem types.

6.4.5 CT↔CCT

The following tables detail the relationships between the causes of complexity and the concepts or classifications of complexity.

Cause Type	Concept / Classification Type	Description of the Relationship
CT1	CCT1, CCT2, CCT3	Design optimisation as a cause is related to the concept or classification of the systems; in a sense the concepts and classifications of the systems govern how complex the design optimisation problem will be. The systems with linear properties are easier to understand and model than those with non-linear properties. Systems with adaptive natures are even more complex and require much more intelligent modelling methods and different design techniques.
CT2		Unless the complexity cause can be associated with system interactions and the nature and structure of those interactions then there are no direct relationships between these causes and the complexity concepts or classifications, as these are a description of the system structure and its nature.
CT3		
CT4		
CT5		
CT6	CCT1, CCT2, CCT3	As for CT1 but for interfacing and intricacy.
CT7	CCT1, CCT2, CCT3	As for CT1 but for the system scale.

Table 27 - Relationships between the cause types and concept / classification types.

	CCT1	CCT2	CCT3
CT1	X	X	X
CT2			
CT3			
CT4			
CT5			
CT6	X	X	X
CT7	X	X	X

Table 28 - Summary of the interactions between the cause types and measurement types.

6.4.6 CT↔CMT

The following tables detail the relationships between the causes of complexity and the coping mechanisms for complexity in systems.

Cause Type	Coping Mechanism Type	Description of the Relationship
CT1	CMT1, CMT2, CMT4, CMT5, CMT6	Depending on the nature of the design optimisation that is the cause of complexity within the system a variety of different coping mechanisms can be employed; numerical (using algorithms to optimise systems), improving the system architecture (making optimisation easier), focusing on the area that requires the highest amount of optimisation or that has the most variables to optimise. Finally previewing the problem before actually encountering it in models may provide tools or options to counter it.
CT2	CMT3, CMT6	Improving the process efficiency may increase the speed at which requirements are matured in the system development. A preview of requirement maturity and its status will enable better management of the issue.
CT3	CMT3, CMT6	With a changing organisational structure, better focus on process and the efficiency of those processes are key when managing organisational transitions. Previews of organisational issues or the effect of the change on the programme will also enable better management of the cause of potential complexities.
CT4	CMT1, CMT3, CMT4	Improving the evaluation methods, the processes used in developing systems and manufacturing systems along with improving design architectures means that future complexities resulting in production problems can be avoided.
CT5		There are no direct relationships between organisational culture and coping mechanisms.
CT6	CMT2, CMT4, CMT5	Interfaces and their intricacy can be managed through numerical analysis of the coupling between them, improving the architecture to reduce unnecessary coupling and heavy focus of resources on those areas that have the highest levels of connectivity and intricacy.
CT7	CMT2, CMT4, CMT5	The scale of the system is managed in much the same way that interface intricacy is managed.

Table 29 - Relationships between the cause types and coping mechanism types.

	CMT1	CMT2	CMT3	CMT4	CMT5	CMT6
CT1	X	X		X	X	X
CT2			X			X
CT3			X			X
CT4	X		X	X		
CT5						
CT6		X		X	X	
CT7		X		X		

Table 30 - Summary of the interactions between the cause types and coping mechanism types.

6.4.7 CT↔PT

The following tables detail the relationships between the causes of complexity and the problem issues associated with complexity.

Cause Type	Problem Type	Description of the Relationship
CT1	PT1	Direct mapping as the problem cause is the problem issue itself.
CT2	PT2	Direct mapping as the problem cause is the problem issue itself.
CT3	PT3	Direct mapping as the problem cause is the problem issue itself.
CT4	PT4	Direct mapping as the problem cause is the problem issue itself.
CT5	PT5	Direct mapping as the problem cause is the problem issue itself.
CT6	PT1	System intricacy and coupling is a design optimisation problem, producing a system that meets the requirement with the lowest most manageable level of complexity possible, an optimised complexity against cost and resource levels.
CT7	PT1	System intricacy and coupling is a design optimisation problem, producing a system that meets the requirement with the lowest most manageable level of complexity possible, an optimised complexity against cost and resource levels.

Table 31 - Relationships between the cause types and problem types.

	PT1	PT2	PT3	PT4	PT5
CT1	X				
CT2		X			
CT3			X		
CT4				X	
CT5					X
CT6	X				
CT7	X				

Table 32 - Summary of the interactions between the cause types and problem types.

6.4.8 CCT↔MT

The following tables detail the relationships between the concepts or classifications of complexity and measures of complexity.

Concepts / Classification Type	Measurement Type	Description of the Relationship
CCT1	MT3	The concepts or classifications of complexity in systems can be associated with any type of measure. It depends on the system that is under evaluation. The concepts of complexity can apply to human, mechanical, electrical, organisational or even chemical systems; as a result the concepts are applicable to all measurement types. When attempting to discover the level of each type of complexity, the only measures appropriate is that which analyses the interfaces of that system, whatever they may be.
CCT2	MT3	
CCT3	MT3	

Table 33 - Relationships between the concept / classification types and measurement types.

	MT1	MT2	MT3	MT4
CCT1			X	
CCT2			X	
CCT3			X	

Table 34 - Summary of the interactions between the concept / classification types and measurement types.

6.4.9 CMT↔PT

The following tables detail the relationships between the coping mechanisms for complexity in systems and problem issues associated with complexity.

Coping mechanism Type	Problem Type	Description of the Relationship
CMT1	PT4	Evaluation or visual evaluations may help reduce pre-production issues.
CMT2	PT1	Numerical analysis of systems in some cases can help improve the ability of development processes in optimising designs.
CMT3	PT2, PT3, PT4, PT5	Process efficiency improves on a vast array of different problem issues and has the potential to cope with or reduce complexity in design, through coping with organisational changes and cultures, along with controlling pre-production.
CMT4	PT1, PT4	Improving the architectures can improve the design optimisation task and also pre-production issues.
CMT5	PT1, PT2	Focusing on the complexity area is a good method of focusing resources on improving requirement maturity and the optimisation process.
CMT6	PT2, PT3, PT4, PT5	Previewing problems helps when reducing complexity as a result of organisational changes or cultures, along with more technical reductions of complexity through requirement maturity and pre-production induced complexities. Previewing problems means they are realised early and the problems can be reduced with actions taken as a result.

Table 35 - Relationships between the concept / classification types and problem types.

	PT1	PT2	PT3	PT4	PT5
CMT1				X	
CMT2	X				
CMT3		X	X	X	X
CMT4	X			X	
CMT5	X	X			
CMT6		X	X	X	X

Table 36 - Summary of the interactions between the concept / classification types and problem types.

6.5 *Conclusion*

This section has highlighted the relationships that exist within the CCCS between its categories (see 4.10.2). These relationships outlined above are key when developing

an understanding of complexity and how various characteristics, components or aspects of complexity can influence each other (for example how the definition of complexity influences the use of measures).

In an ironic way, the relationships that exist within the Complexity Framework are themselves intricate (complex). But the understanding of or appreciation of these relationships is key when attempting to understand complexity with systems, in engineering, chemical, physical or biological terms.

The framework enables problems to be linked intelligently with other characteristics of complexity (concepts, definitions, measures, classifications). The framework also enables indirect linking of problems with solutions, and also provides the justification of those links.

The relationships between the complexity categories from the basic Complexity Framework is the understanding of complexity within engineered systems that industry needs, and in itself can be used as an approach mechanism to manage complexity in systems and prevent or predict problems.

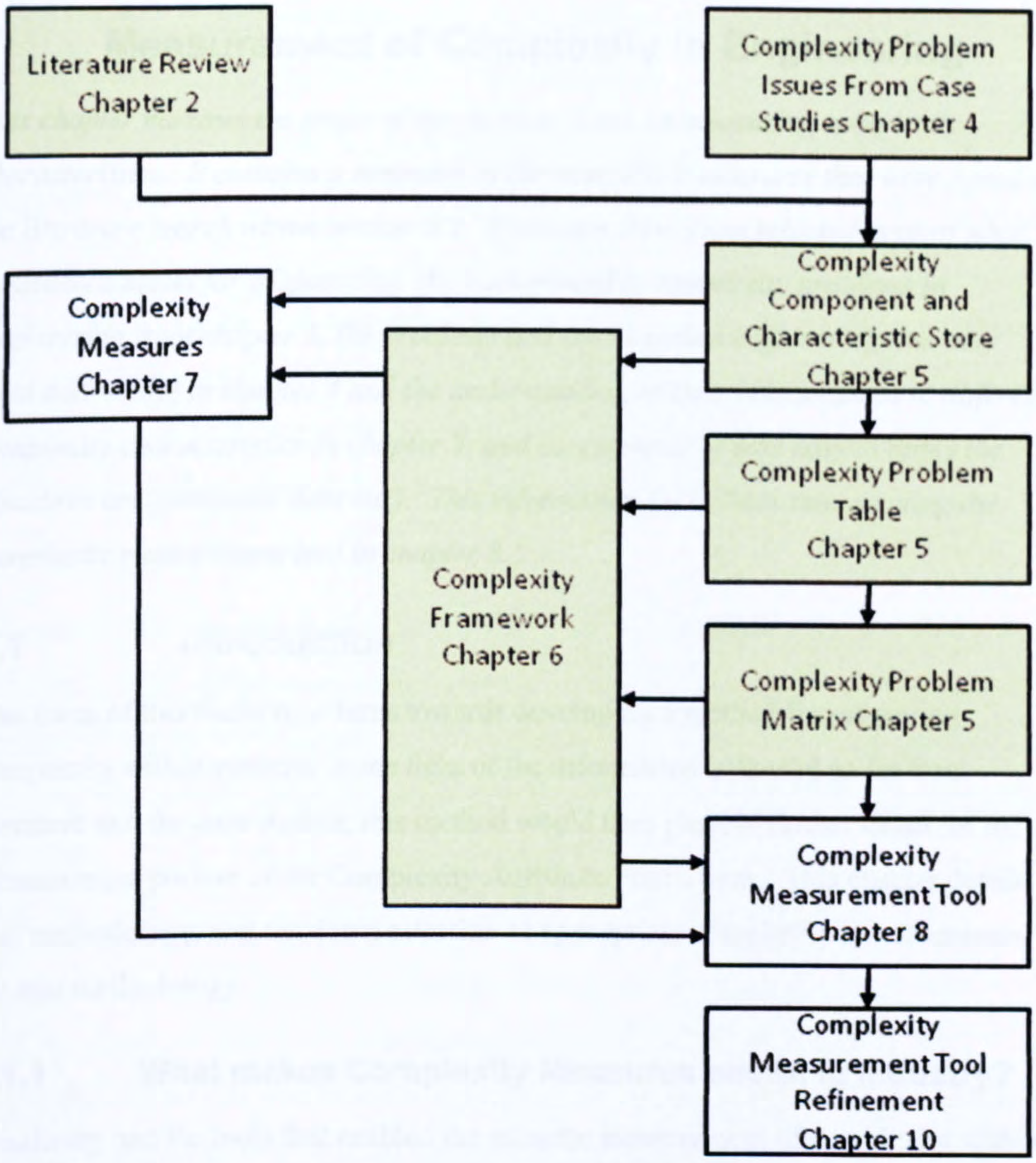


Figure 36 - The layout of the work and the thesis outputs roadmap.

Figure 39 shows the roadmap of work within the thesis, within this chapter the Complexity Framework was developed. This framework needs to be built upon and enhanced in each area (concepts, classifications, measures, etc.). To cover all areas of the Complexity Framework is beyond the scope of this thesis, and the decision was taken to enhance the understanding of complexity measurement in engineering systems, in particular within the development phase as it was of great interest to the industrial sponsor. The objective of developing measures of complexity for engineered systems is to develop an approach (and tool) to enable the measuring of system complexity within industry.

7 Measurement of Complexity in Engineering

This chapter narrows the scope of the thesis to focus on measuring complexity characteristics. It contains a summary of the complexity measures that were found on the literature search within section 2.7. These are then down selected against what is considered useful for engineering: the background to complexity problems in engineering from chapter 3, the problems and issues within engineering that may need addressing in chapter 4 and the understanding of their relationships to different complexity characteristics in chapter 5; and assessments of how easy to apply the measures are (available data etc.). This information then filters into creating the complexity measurement tool in chapter 8.'

7.1 Introduction

The focus of this thesis now turns towards developing a method for measuring complexity within systems, in the light of the information collected so far from literature and the case studies; this method would then provide further detail for the measurement portion of the Complexity Attritbute Framework. This chapter details that methodology, and the down selection of appropriate complexity measurements for that methodology.

7.1.1 What makes Complexity Measures useful to Industry?

If industry had the tools that enabled the accurate measurement of complexity within systems, or the overall complexity of programmes, it would be better equipped to assess the level of effort and budget required. The key question that industry needs to answer is how a complexity measure is of use? It is one thing to have a complexity value for a system, but unless that complexity value can be interpreted in a way that provides useful information that can be fed back into projects beneficially the measure is of no real utility. The following are a set of questions, the answers to which would be of benefit to any organisation in the process of developing a system.

In a technical or hardware sense, which area of a system under development is the cause of the complexity within the system?

A measure that can assess the complexity of a system (or sub-system) in terms of software, hardware, its location and the type of complexity that exists would provide

industry with the ability to better understand what they are developing or dealing with and give them the potential to foresee future problems.

What are the future problems that this development programme is likely to exhibit?

A complexity measure can assess both the technical and development process complexity, and could give an estimation of the number of problems likely to occur. However this measure would only provide an overall estimation, it would not offer any insight into which part of the system would cause the issues, or which phase of the development programme the problems are likely to affect.

Is the complexity manageable?

Industry needs to be able to accurately assess the risk associated with developing a complex system. If the complexity contained within that system is very large, it has the potential to be unmanageable. If the complexity is unmanageable the success of the programme is brought into question and this will impact the profit margins for that company.

How does this system compare with other systems in terms of complexity?

In order to accurately predict the required effort levels involved for future work, industry often uses previous programmes that are comparable to help with the estimation. Complexity will be a contributing factor in terms of the required resource or effort level required to produce the system. A measure of complexity will allow this contributing factor to be assessed and influence the decision regarding effort levels and feasibility. In some cases these comparisons might be with similar programmes; in other cases they may be comparisons with upgrading an existing product, or developing an entirely new one.

What effect would the introduction of another element into the system have in terms of the overall complexity?

Upgrades are often additions in terms of elements within already established systems, and industry must assess the feasibility of the upgrade and the impact on the overall characteristics the upgrade will have. The upgrade may change the overall characteristics of a system in terms of its complexity, and this could change the required resource to operate that system satisfactorily, affecting the support levels and

therefore the associated costs. Industry needs to be able to accurately predict these changes and a complexity measurement that is comparable to other systems would be a factor in this prediction.

7.1.2 What will be Measured?

Where are the key areas within a lifecycle where a complexity measure would provide benefit that answers those industrial (see section 7.1.1) questions? Where would a complexity measure application be most beneficial? What form must a complexity measure take in order for it to be useful? The following are a set of questions and answers that examine what could be measured, and how it is of benefit to industry when developing systems.

Where are the key areas within a lifecycle where a complexity measure would provide benefit? Where would a complexity measures application be most beneficial?

Measuring complexity after a system has been developed leaves no room for improvement; testing the complexity of systems before they are fully developed leaves room for improvement or changes that could reduce the levels of resource required. In this case the primary stages within development lifecycles will be the early bid stages, early development stages or design stages, and system upgrading, areas where programmes or are setup, their resources allocated, or the system is created or modified. Understanding complexity at these stages can enable informed decisions to be made and resource allocation optimised, in the hope of pre-empting or controlling complexity levels.

Upgrading or building upon current products further along in the product lifecycle presents its own problems. At which stage does building upon a product that already exists and is in service become more complex that simply designing a new product that meets all the functional requirements? A complexity measure that could identify the complexity of the product and complexity of the product once the upgrade is completed would be useful in determining the viability of the upgrade itself.

It can be said the complexity of a developed system is dependent on the environment within which that system operates, the environment will determine to some degree the behaviour of a system, the combination or interaction between the system and its

environment we will call the super system. A developed system does not always operate in a single environment at all times, but a multitude of environments defined by system operating parameters or requirements. The operating environments that can be created from these defined parameters will have an effect on the overall complexity of the super system, a more diverse environment parameter set, the higher the diversity within the super system. Understanding the nature of complexity of the super system (i.e. the developed system and the environment) allows the operating environment envelope to be selected such that the complexity of the super system never reaches an unmanageable level.

What form must a complexity measure take in order for it to be useful?

In most cases a single point value will not be sufficient - systems operate within an environment and this will have an effect on the operation of the system. If the complexity measure required is for an operational system, the different environments must be taken into account, and appropriate measures produced for each complexity environment.

The measurement must be comparable; it would also be beneficial for the measure to be easily visually represented enabling fast comparison and the information it represents to be easily conveyed.

Calculation of the measure must be achieved within a reasonable time - measures that require too much resource for calculation are less useful than those that can be calculated quickly, providing the accuracy is satisfactory.

Complexity was defined as the intricacy of the interactions of system elements, so a measure or combination of measures must provide a description of this property of a system. Complexity components within developed systems can be found within the complexity component model and it this information that will require measurement in order to assess the associated intricacy.

7.1.3 What are the Characteristics of a Measure?

To obtain an understanding of what industry wants and needs from a complexity measure, or measurement approach the next stage is to understand the various methods of measurement and types of measurement that are possible.

The classical scientific concept of measurement requires a distinct device that selectively interacts with the system being measured, and an output that has symbolic interpretation, usually numbers. That being measured can be real or theoretical, for example a physical attribute of an object or management effort metrics.

Measurements properties include units (the name of the quantity value) and scales, tailored for the value quantity being measured. Different quantity value scales allow more direct comparison of the same measure from one situation to another. Scales exist in different forms:

- Categorical = Age 18-24; 25-30; 31-40; 40+
- Ordinal = Female or Male / Yes or No
- Interval = Strongly Agree, Agree, Disagree, Strongly Disagree
- Ratio = Calendar time & date / Vulgar fractions / Length

There are requirements for useful measurements, they must be precise (a function of extent, granularity and frequency), resettable (because the measure can be repeated) and reproducible (because the measuring device must be isolated from the system being measured).

In order to apply measurement to complexity within the engineering domain, the requirements for measurement must be interpreted for the complexity problem and subsequently met. Complexity could exist within more than one scale, but the values of the different scales will change depending on which questions industry wants answering. For example, complexity within the Boolean scale has little value within an engineering environment, distinguishing between complex and simple systems in a Boolean manner provides very little useful information, where as categorical scales, intervals and ratios provide a much more detailed picture of complexity.

Scale range is also important, complexity can be thought of as a scale, from simple (or low complexity) to high (or high complexity) which could be near random behaviour. Obviously systems that exhibit different levels of complexity will need different scales, similar to the different scales available for measuring temperature, Kelvin, Celsius and Fahrenheit.

The precision in complexity measurement is more difficult, as the measure is highly subjective due to the large number of different approaches that can be taken to calculate it and the number of different forms a complexity measure can take. A set of defined measurements and measurement methods for complexity as it is in industrial systems is necessary to standardise the output and then precision can follow.

A number of measures for complexity exist in the conceptual form, for example system reducibility. In the engineering case there is sometimes scope for reduction of systems, if those systems are large and contain large numbers of elements and interactions and there is no scope for reduction it would suggest that system has a high level of complexity. However this is subjective, systems that are in production today could have scope for further reduction that is either undetectable or it is impractical. For that proportion that is undetectable the level of reducibility available is inaccurate, the system is reducible, but this reduction is overlooked. It would be very difficult and time consuming to test for reducibility in modern systems independent of the development programme. So in the abstract view reducibility is a good measure of complexity, but in reality such a measure is impractical. This is just one view or interpretation of this complexity measure - it may apply to the outputs of a system (the reducibility of the outputs) or perhaps modelling of systems (accurate modelling of behaviour with a set of reduced variables and functionalities).

There are a number of abstract or concept measured that are not rigidly defined. These measures were assessed as to whether these abstractions can be turned into measures that can be applied.

7.2 Complexity Measurement Categories

There are distinct categories of measures that can be applied to the majority of measures found within the literature search (chapter 2), however not every measure can be directly applied to these categories. The complexity measures within the literature are all different, but essentially they all seem to follow a pattern that can be categorised. It is the categories and some of the measures themselves that could form the basis for measuring and understanding complexity within systems.

These categories or complexity measure common themes found within the literature are as follows:

- Reducibility Measures – Measures that concentrate on the principle that complex systems are not reducible, and measuring the optimised system gives an estimation of the inherent complexity of that system.
- Measurement of Input/Output Complexity – Information theory measuring the ease of modelling of the input or output of a system. Usually only applicable to those systems that produce string or character outputs.
- System Specific Measures – Measures relating directly to the system or development programme itself, in terms of interfaces, people, resource, elements, and so on.
- Measure of Randomness – Measures that relate to the level of unpredictability within a system.
- Miscellaneous Measures – Measures that do not conform to any of the above categorisations.

Using this category list, and relating this list to the measures themselves taken from the literature (and created within this thesis), the categories can be related to the complexity problems studied. How the problem issues relate to specific complexities within the system will govern which categories or themes are most appropriate. If the problem issue is related to size and intricacy, the system specific measures may be most appropriate. If the system is subject to disorder in inputs or outputs, perhaps randomness measures and input output complexity measures are appropriate.

This is not the only way in which measures can be separated into categories. Based on the knowledge of measurement (see section 7.1.3), the characteristics of the complexity measures are also important. If data is limited, perhaps qualitative measures are more appropriate, perhaps data is abundant in which case quantitative measures can be calculated.

The following identifies how the characteristics of the measures can change, and it is these characteristics that may make the measures more or less viable when attempting to understand a systems complexity.

- Quantitative / Qualitative.
- Apply to; Reducibility, Input/Output, System Specific, Randomness or Miscellaneous.
- Ratio, Ordinal, Interval, or Categorical sales.
- Apply to; System Design (Interfaces, Skill Variation, Decompositions, Requirements, System States, Reducibility, Control Loops, Computation), Management (Skill Variation, Personnel Variation).

Table 37 contains the measures found within the literature search and also those created for the purpose of this research with their characteristics defined. This breakdown will help the selection process of appropriate measures that can be used for the Complexity Framework and the categorisation is based on an industrial systems engineering domain.

Measurement	Qualitative / Quantitative	Category	Scale Type	Potential Applications
IBDF - Interdependencies Between Different Factors	Qualitative	System Specific	Ratio	System Design (Interfaces)
NSI - Number of Skills Involved	Qualitative	Miscellaneous	Ratio	Management (Skill Variation)
PIA - Persons Involved in Analysis	Qualitative	Miscellaneous	Ratio	Management (Personnel Variation)
Abstract Computational Complexity	Quantitative	System Specific	Ratio	System Design (Decomposition)
Algorithmic Information Complexity	Quantitative	Input/Output, Reducibility	Ratio	System Design (Requirements)
Bennett's 'Logical Depth'	Quantitative	Input/Output, Reducibility	Ratio	System Design (Requirements)
Cognitive Complexity	Qualitative	Miscellaneous	Ordinal / Interval / Categorical	Management (Skill Variation)
Connectivity	Qualitative	System Specific	Ratio	System Design (Interface)
Cyclomatic Number	Quantitative	System Specific	Ratio	System Design (Control Loops)
Descriptive/Interpretative Complexity	Quantitative	System Specific	Ratio / Interval	System Design (Requirement)
Dimension of Attractor	Quantitative	Miscellaneous	Ratio	System Design (System States)
Ease of Decomposition	Qualitative	System Specific	Interval / Categorical	System Design (Interfaces, Decomposition)
Entropy	Quantitative	Randomness	Ratio	System Design

Measurement	Qualitative / Quantitative	Category	Scale Type	Potential Applications
				(Requirements)
External Complexity	Quantitative	System Specific	Ratio	System Design (Interfaces)
Goodman's Complexity	Qualitative	Miscellaneous	Categorical	System Design (Overall Design)
Horn Complexity	Quantitative	System Specific	Ratio / Categorical	System Design (Requirements)
Information	Qualitative	Randomness	Ratio / Categorical	System Design (Requirements)
Information Gain in Hierarchically Approximation and Scaling	Quantitative	System Specific	Ratio	System Design (Decomposition)
Interface Complexity	Quantitative	System Specific	Ratio	System Design (interfaces)
Internal Complexity	Quantitative	System Specific	Ratio	System Design (interfaces)
Irreducibility	Quantitative	System Specific	Ratio / Categorical	System Design (Requirements)
Length of Proof	Qualitative	Miscellaneous	Categorical	N/A
Link Complexity	Quantitative	System Specific	Ratio	System Design (Interfaces)
Logical Complexity/Arithmetic Hierarchy	Quantitative	System Specific	Ratio	System Design (Decomposition)
Low Probability	Quantitative	Miscellaneous	Ratio	N/A
Minimum Number of Sub Groups	Quantitative	System Specific	Ratio	System Design (Decomposition)
Minimum Size	Quantitative	Reducibility	Ratio	System Design (Decomposition)
Mutual Information	Quantitative	Randomness	Ratio	System Design (Decomposition)
Network Complexity	Quantitative	System Specific	Ratio	System Design (Reducibility)
Number of Axioms	Quantitative	System Specific	Ratio	N/A
Number of Dimensions	Quantitative	System Specific	Ratio	System Design, Management (Skill Variation, Personnel Variation, Interfaces, Decomposition, Requirements)
Number of In-equivalent Descriptions	Quantitative	System Specific	Ratio	System Design (Requirements)
Number of Internal Relations	Quantitative	System Specific	Ratio	System Design (Interfaces)
Number of Spanning Trees	Quantitative	System Specific	Ratio	System Design (Interfaces)
Number of Variables	Quantitative	System Specific	Ratio	System Design, Management (Skill Variation, Personnel Variation, Interfaces, Decomposition, Requirements)
Organised/Disorganised Complexity	Quantitative	Randomness	Categorical	System Design (Decomposition)
Scale Complexity	Quantitative	System Specific	Ratio	System Design (Scale, Interfaces)
Shannon Information	Quantitative	Input/Output	Ratio	System Design (Requirements)
Simplicity	Qualitative	Miscellaneous	Categorical	N/A
Size	Quantitative	System Specific	Ratio	System Design, Management (Skill Variation, Personnel Variation, Interfaces,

Measurement	Qualitative / Quantitative	Category	Scale Type	Potential Applications
				Decomposition)
Socio-Political Complexity	Qualitative	N/A	N/A	N/A
Super System Landscape	Quantitative	System Specific	Ratio	System Design (Overall Design)
Variety	Quantitative	System Specific	Ratio	System Design (Overall Design)
Kolmogorov Complexity	Qualitative	Randomness	Ratio	System Design (Requirements)
Rissanens Minimum Description Length (MDL)	Quantitative	Input/Output	Ratio	System Design (Requirements)
Time and Space Computational Measures	Quantitative	System Specific	Ratio	System Design (Computation)

Table 37 - Complexity measurement summary and compliance with industrial uses.

Reviewing Table 37, there are an abundance of measures applicable to certain aspects of the engineering domain, such as system specific measures; however, there are some areas that are barely covered at all, for example, reducibility measures.

The majority of measures are quantitative, and most use a ratio scale, there is a substantial use of the categorical scale but still not as significant as the ratio scale. There are a small number of qualitative measures, which may be more easily applied to the engineering domain under study.

There are a large number of interfaces, requirements and decomposition applications for the measures. Whether they are abstractions or direct applications of those measures it has yet to be determined, but this does show that the measures do suit the domain and there is sufficient scope to use them in developing a measurement structure.

There are four measures that have no obvious use within the engineering domain under study, these are Low Probability (not obvious how this could be applied to engineering), Length of Proof (a mathematical measure, systems themselves do have proofs), Simplicity (very vague, could potentially be used in a system specific manner, but there are other measures that could be applied more easily) and Number of Axioms (as simplicity). These do not correspond to any complexity or complexity abstraction that could produce useful metrics within the engineering domain. They are more relevant to the mathematical domain and their application to industrial system is difficult.

The measures in the table fit specific categories, however the complexity problem within engineering is multi-dimensional and cross category. As Martins (Martin, Pierre-Alain J. Y. 2004) measures address the internal, interface, scale, external and link complexities (with the exception of his socio-political), these are all complexities of the system intricacy and interactions. Martin focuses on the products themselves as produced, the internal complexities of those products in terms of scale and interactions, and then moving on to interface complexities and socio political complexities. However, the focus here is for a system that is already developed, a complexity that already exists and that needs quantifying, but what about those systems that are not yet produced? And what about those aspects of system development that are beyond just the elements and their interfaces? The complexity measurement process needs to provide information to business that allow an understanding of complexity within the system as the product is under development. These multi-dimensional measures must provide information that allows more accurate and reasonable estimations of the resource required and the likelihood of success and the level of manageability. In order to do that, variety, commonality, and understanding of the concepts and classifications within the system need to be quantified too.

Within the measurements that have been explored here, and the questions that have been identified, there seems to be a great potential for measurement of system complexity using the requirements for that system. As a result of this multi-dimensional approach, measures must be selected that cover the questions industry wishes to answer.

Not only does a multi-dimensional approach to complexity provide a basis for understanding the nature of the complexity within the system, what causes it and where it is within the system, it also provides a tool for the elimination of unnecessary induced complexity. It provides various measures that can all be used to get a complete picture of the systems' complexity, where it comes from, what triggers it and how the approaches to dealing with it will effect it; Do these requirements need to be this highly interactive? Are these interactions necessary? Can the interaction be reduced without cost to the required functionality? Can more commonality reduce the complexity? Is the system hierarchical? Can the system be made more hierarchical?

7.3 Measurement Selection

Before a set of measures can be selected, these measures should answer the questions (or at least most of them) that industry needs to answer (see 7.1.1 with justification for those questions in 7.1.2). This forms the basis of what these measures will cover in order to provide answers for the questions industry needs answering. Taking the questions from the previous sections that industry needs answering, the following categorisation of measures of complexity was derived.

- Requirement Complexity Measures
- System Decomposition Complexity Measures
- Interface Complexity Measures
- Element Complexity Measures
- Management Complexity Measures

The following sections will in turn explore these areas and the relevance and appropriateness of complexity measures to industry.

7.3.1 Requirement Complexity Measures

There are 13 complexity measures which either have a concept that can be applied in an abstract (that is not directly but the idea of the measure can be used to create another basic measure for systems) form or directly to engineered systems. Six areas were selected as relevant to measuring requirements complexity.

1. Information Theory Measures; potentially measuring the level of information within the requirements set.
2. Interpretation or In-equivalent Descriptions; potentially measuring the variation in the final product from the requirement set description of that product.
3. Irreducibility; potentially measuring how bloated the requirements are.
4. Logical and Algorithmic Measures; measuring the logic within the requirements.
5. Randomness; potentially measure the level of disorder in the requirement set.

6. Number of Variables and Dimensions; potentially measure the different aspects covered within the requirements (safety, design, etc.)

The following sections investigate the applicability of the measures indentified within the literature review (chapter 2) with regard to these areas. When considering complexity within requirements the following industry questions were considered:

- How small can the requirement set be made without reducing the system?
- Where are the critical complexities within the requirements?
- How coupled are the requirements?
- How large is the potential solution diversity?

7.3.1.1 Information Theory as a Requirements Complexity Measure

Information theory can show us some of the difficulties which arise when applying complexity measures. Shannon (Shannon 1948) and Kolmogorov's (Li 1997) measures for information theory (see section 2.7.2) specialise in the complication of replication (or compression) of a string output into an algorithm. Within the requirements domain there is a tenuous link between this search for structure within strings and the search for structure within requirements. However, the application of these measures in their pure form is extremely difficult to impossible. There is no value in understanding the properties of the strings of characters within a requirement set, or how that set can be replicated using an algorithm when trying to understand the system. The concept cannot easily be applied in an abstract sense either, and as a result these two measures are not going to answer many, if any, of the questions outlined above.

Understanding the potential for reduction, however, is applicable, so measures that use a different concept for reduction should be considered further. Rissaen's MDL (Barron, Rissaen et al. 1998) (see section 2.7.2) is a similar measure, but it places emphasis on minimum description length. As an abstract term this fits nicely within the requirements domain; it may be sensible to ensure that even when optimised a description of the requirements for a system is large, and it has a higher complexity than if the optimised description was low. Although the abstraction to the engineering world is simple, the application is more difficult - how is it known when the requirements are optimised? How much work is required to optimise a large

requirement set? One has to question the point of such an exercise - if a requirement set has been optimised to reveal the minimum description length in terms of its requirements, there is no room for improvement, Rissaen's measure would simply measure the complexity, at this point, in requirement document size terms. Size already exists as a measure within some companies; as a rough estimation of complexity the size of the requirements document (assumed to be optimised COSYSMO) is a rough measure of complexity, but a very poor one. Rissaen does take this a lot further, as this method would mean coding requirements into a form where the system could be optimised, but the effort involved would out way the rewards.

7.3.1.2 Interpretation and In-equivalent Description Measures as Requirements Complexity Measures

Descriptive or interpretive complexity has a strong link with complexity within engineering. In requirement terms, the more ways in which a requirement set can be interpreted or described in terms of compliant solutions the more complex the process to reach the solution due to the level of choice. If this measure is combined with the number of in-equivalent descriptions, this becomes a very valid measure of complexity within the requirement set, but how can this be applied? The application may be easier at a top level within the requirement structure, but the derived requirements may actually have a limiting effect on the level of interpretation available. A lower level estimation of requirements relating to different system elements may be sufficient to get a complexity value.

7.3.1.3 Irreducibility Measures as Requirements Complexity Measures

Irreducibility links to Rissaen's MDL measure, and has as very similar output. Understanding how reducible the requirements set is requires effort in actually reducing it to the minimum description. There is no simple way to theoretically calculate the reducibility of a requirement set without actually doing the work, and one has to question the need of a reduction prediction if the reduction has to be done in order to get the answer anyway.

Commonality within the requirements, or a high level of commonality allows potential reduction in the requirements set. The higher the commonality within the requirements set the more reduction is possible.

7.3.1.4 Logical and Algorithmic Measures as Requirements Complexity Measures

Logical and algorithmic expressions of complexity have been selected as potential measures for complexity within the engineering domain. Again, although the principles are good, and there is an application for these principles, the practical application to engineering requirements is not obvious.

Horn complexity is an example of creating a logical model of a system then re-ordering or re-writing the logical expressions so they become simpler and expressed more effectively (similar to minimisation). It has a direct application to the requirements problem, with relevance to all questions above. It may enable an understanding of the extent of requirements coupling within design, if there is a heavy referral to a system element (sub-system) within the requirement structure, critical complexities can be identified, and the effect of requirement changes or additions to the system can be understood by the effect the change has on the logical model.

7.3.1.5 Randomness as a Requirements Complexity Measure

Entropy is the measure of disorder in a system - the higher the level of disorder and randomness in a system, the higher the entropy. The entropy of the number of skill sets required and the relationships of these skill sets to the design may have a link to the complexity of the system. If these are ordered the entropy will be low, and perhaps the complexity of the system lower at least in terms of the design process.

The order of the inter-relations between different components or variables within a design is also a potential entropy measure. The higher the level of disorder within the interfaces between elements the higher the complexity of the system, it is however important to note that the complexity exhibited may not be within the design, but also within the operation of the system, the unordered interactions will take place as the system is operating.

Neither of these applications have a direct bearing on the requirements issue. The level of disorder within the requirements is difficult value to calculate, it is also hard to define requirements that are ordered and requirements which are not, or sets of requirements with no order. The skill set application of entropy is not really a measure of order or disorder, but diversity in the requirement structure. This is

perhaps where measures of this type should be focused, concentrating on the level of diversity and not the level of disorder.

7.3.1.6 Number of Variables and Dimensions Measures as Requirements Complexity Measures

Number of variables may be a good measure in requirements terms and is similar to the number of dimensions within the requirement structure. In relation to requirements these may provide some complexity answers, but the question arises what are requirement dimensions or variables?

The variables are perhaps more easily found, contained within the requirements themselves as tolerances, the number of variables however may require a large effort to ascertain the real value. There are also variables that occur as a result of derived requirements, these will need to be accounted for.

Dimensions are highly subjective, but could be a related to the skill set requirement of the system under study or applied to different sub-systems of the complete system. There is a large scope for the application of the dimensions to the requirements for complexity measurement, which makes this method a more versatile approach.

7.3.1.7 Summary of Requirement Complexity Measures

Although not mentioned as a specific complexity measure in itself, commonality should also be included as a measure to provide understanding of the nature of the complexity of the requirement set. Table 38 shows the appropriate complexity measures selected from the literature review (chapter 2) and their direct applicability to the questions outlined.

Complexity Measurement	How small can the requirement set be made without reducing the system?	Where are the critical complexities within the requirements?	How coupled are the requirements?	How large is the potential solution diversity?
Horn Complexity	X		X	
Number of Variables		X		
Number of Dimensions	X		X	
Number of In-Equivalent Descriptions				X
Descriptive/Interpretive Complexity				X
Commonality	X		X	

Table 38 - Requirement complexity measures against requirement question criteria.

7.3.2 System Decomposition Complexity Measures

There are 6 complexity measures which either have a concept that can be applied or can be directly applied to engineered systems. Four areas were selected as relevant to measuring decomposition complexity.

1. Number of Dimensions and Variables; measuring the number of different dimensions or disciplines covered within the systems.
2. Element and Interface Complexity Measurement; measuring the intricacy between elements, and the elements themselves.
3. Decomposition Measures; measuring the ease of which the system can be decomposed, or broken down.
4. Commonality Measures; measuring the level of commonality which may or may not reduce complexity within the system.

In terms of complexity within the decomposition of systems, industry has a number of different perspectives to consider. The product or system development programme, which has many different stages requiring knowledge of how different decomposition strategies will affect the complexity of the output. The product system in terms of the supply chains that support or create it and the manufacture processes. Different decomposition strategies will probably suit different stages of the system lifecycle, it is important for industry to keep the complexity caused be the decomposition to a minimum for all stages. The questions that need to be answered are:

- What is the optimum split of a system in terms of sub-systems?
- What is the optimum split of a system in terms of supply chain?
- What is the optimum split of a system in terms of manufacture?
- What is the optimum split in terms of the overall development and operation?

There are several measures that have abstractions that can be applied to the system decomposition area; however, some are more applicable than others. The following sections examine the various complexity measures and their appropriateness to system decomposition.

7.3.2.1 Number of Dimensions and Variables as Decomposition Complexity Measures

The number of dimensions and variables within the decomposition problem provides some scope for measurement. The dimension approach could be provided to test the number of skill set dimensions required for each sub-system decomposed for example (which is essentially the NSI measure), and the variable number could provide a limit in terms of what is manageable per sub-system decomposed (this could be expanded to include the required resource level which links with the PIA measure). In either case the higher the dimension or variable number per sub-system the more likely it is that the sub-systems will have a high complexity. The overall average could provide a measure of complexity within the entire system.

7.3.2.2 Element and Interface as Decomposition Complexity Measures

Martin's measurement set (interface, external, internal, link and socio-political complexity) is specific as it tackles the internal complexities within systems in terms of their decomposition. Martin created internal, external and interface complexity measures that could be applied to the chosen decomposition. Again these provide a method of comparing complexities across different system decomposition structures.

7.3.2.3 Decomposition Measures as Decomposition Complexity Measures

The ease of decomposition although as the title would suggest directly applicable, is not strictly a measure but an aid when developing products. In itself does not help in answering the questions that industry wishes to answer relating to system decomposition. However linked with scaling this becomes an important factor, depending on the scaling or information hierarchy chosen by the developer when measuring the complexity of the level of detail of the elements represented in the model will change, but the axioms of those components will remain the same.

7.3.2.4 Commonality as Decomposition Complexity Measures

Mutual information as a principle has an obvious application to system decomposition. It may be that with a specific sub-system split the complexity level is high, but changing the decomposition structure rather than the system itself may reduce the complexity. This in particular applies to the final question above, industry is interested in the mutual information that exists between all the areas of design, and

how the decomposition split between sub-systems affects the complexity of the manufacture. The complexity measure the mutual information concept offers over the whole spectrum is of great interest for industry when developing its systems. This implies that complexity should be measured independently for different aspects in design. Measures should exist that are domain specific within design, and the mutual information between them studied in order to optimise the decomposition early.

7.3.2.5 Summary of Decomposition Complexity Measures

Table 39 shows the appropriate complexity measures and their direct applicability to the questions outlined.

Complexity Measurement	What is the optimum split of a system in terms of sub-systems?	What is the optimum split of a system in terms of supply chain?	What is the optimum split of a system in terms of manufacture?	What is the optimum split in terms of the overall development and operation?
Information Gain In Hierarchically Approximation and Scaling	X	X	X	
Martins Measurement Set	X			
Mutual Information	X	X	X	X
Number of Dimensions	X	X	X	X
Number of Variables	X	X	X	X

Table 39 - Decomposition complexity measures against decomposition question criteria.

7.3.3 Interface Complexity Measures

There are 9 complexity measures which either have a concept that can be applied or can be directly applied to engineered systems. Four areas were selected as relevant to measuring interface complexity.

1. Interface Complexity; the complexities regarding the interfaces within the system.
2. Connectivity Measures; the connectivity of the interfaces within the system and how it can be measured.
3. Dimensions; the number of different dimensions to the interfaces (types, matter, etc.)
4. Size; the size of the system.

When developing systems, industry must consider the interactions within the design. These form the basis for a large number of definitions of complexity which have been explored elsewhere, one of which is quite prudent is the Evans' definition of complexity, which centres on the interactions being the cause of the intrinsic complexity within systems.

Industry must be able to control or measure the level of interaction within the systems they create in order to maintain control over the design or have confidence in the system functionality.

- What is the number of interactions within the system?
- Where are the concentrations of those interactions?
- What is the level of coupling within the system?
- How will interface changes affect the system?
- How complex are the information flows within the system?

The following sections examine the various complexity measures as measures to measure interface complexities in systems.

7.3.3.1 Interface Complexity Measures as an Interface Complexity Measure

Martin's measures are an obvious match for this specific area of complexity within systems. The details around the measurement system Martin devised are aimed directly at quantifying the complexity within the system itself (link complexity, interface complexity, scale complexity, etc.). The measure as it stands can be directly applied to a system to estimate the complexity level, and from a breakdown of the measure the questions surrounding interaction number and concentration can be answered.

Although Martin aims to understand complexity from a design point of view, his method completely ignores issues that surround information flow within systems. The number of spanning trees within systems is also a measure of complexity, an understanding of the paths information can take and an understanding of the interface layout can provide further information. Senge (1994) referred to information paths as detail or dynamic complexity for systems with hierarchical only and both lateral and

hierarchical interfaces. This can further be expanded upon to include systems where the interface structure changes over time; this too is ignored by Martin in his measurement scheme.

7.3.3.2 Connectivity Complexity Measures as an Interface Complexity Measure

Connectivity, internal relations and interdependencies between factors are measuring practically the same thing. However, connectivity measures are more quantifiable, as they are related to actual interfaces rather than interacting factors which have a degree of subjectivity associated with them. The IBDF and internal relations measures are subject to interpretation of interacting elements within the system, whatever they may be, so semantics would need identifying which define clearly these 'interacting factors'.

There are other methods or measures that are not found within the literature that may need addressing for interfaces within design when considering the connectivity or perhaps the style of interfacing, be it hierarchical or non-hierarchical. For this spanning trees and perhaps a measure that analyses the nature of the system interfaces, making a distinction between detail and dynamic complexities within the system and quantifying that in some manner.

7.3.3.3 The Number of Dimensions and Variables as an Interface Complexity Measure

The number of dimensions and variables always have an application to measurement; there is always a variable that needs measurement, and a dimension to the problem. In this problem there are potentially many different interpretations as to what a dimension exactly is in interface terms, dimensions to the interface problem could be coupling, connectivity or interface strength to name some examples. Variables within interface complexity measurement consist of much of the same, and again are applicable to all measurement as you are measuring a variable. The variables within interface measurement may be number of interfaces, size of vocabulary (Martin, Pierre-Alain J. Y. 2004), interface strength, information pathways, and so on. The variables and dimensions can be used to create other measures, but alone they are not suitable measurements for interface complexity.

7.3.3.4 Size as an Interface Complexity Measure

Size related to system interfaces are loosely linked with complexity in the system design. Unfortunately there is not a direct correlation with complexity within system interfaces and the size of the system. There are examples of systems that are massive in size but which, overall, exhibit a very low intrinsic complexity, although they can exhibit emergent properties (Kauffman, Game of Life). There may be a point at which size within interface design will increase the complexity (where the modelling or understanding of a system is hindered by its size) of a system but the link is tenuous.

7.3.3.5 Summary of Interface Complexity Measures

Additional measures that should be considered also are commonality measures between interfaces, and also measures that help define the structure, these measures are included as:

Commonality – The level of commonality within the system interfaces.

System Interface Nature – A measure that determines the nature of the connectivity, be it detail or dynamic complexity (see section 2.6).

Table 40 shows the appropriate complexity measures and their direct applicability to the questions outlined.

Complexity Measurement	What is the number of interactions within the system?	Where are the concentrations of those interactions?	What is the level of coupling within the system?	How will interface changes affect the system?	How complex are the information flows within the system?
Connectivity	X	X	X		
Martins Measurement Set	X	X		X	
Number of Internal Relations	X				
Number of Spanning Trees					X
Commonality				X	X
System Interface Nature			X	X	X

Table 40 - Interface complexity measures against interface question criteria.

7.3.4 Element Complexity Measures

There are 6 complexity measures which either have a concept that can be applied or can be directly applied to engineered systems. Three areas were selected as relevant to measuring element complexity.

1. Dimensions and Variables; the variation within the different element types.
2. Size; the size of the system and how the element number or size may affect the system.
3. Variety; a look at the commonality and variation within the system as a complexity measure.

When developing systems in engineering, both interfacing and the elements within the system need to be understood. When considering measures that can be applied to elements within systems, the questions industry wish to answer must be understood.

- What is the level of commonality within the system?
- What are the functions of the elements?
- What is the spread of functionality (which elements do what functions and where) within the elements of the system?
- How large is the system?

The following sections examine the various complexity measures as measures to measure element complexities in systems.

7.3.4.1 Dimensions and Variables as Element Complexity Measures

Dimensions and variables have a very abstract connection to element complexity, functionality, size and functionality spread. Although dimensions and variables could very vaguely represent different functionalities, it would perhaps be more difficult to include these as measures than it would be to create dedicated measures for that specific purpose.

7.3.4.2 Size as an Element Complexity Measure

Size of the system is obviously of interest to industry and the simple application of a size measure would help in determining spreads. Element counts would form a good size measure for a system (which will not be further decomposed), and perhaps sub-

systems sizes for systems that can and will benefit from further decomposition a sub-system size based on the number of elements contained within that sub-system would be one method or one dimension to the complexity view.

7.3.4.3 Variety as an Element Complexity Measure

The variety within the system is key when considering element functionality and the spread of that functionality within the system. Commonality between elements within sub-systems, and systems as a whole generally imply that the engineering process is simpler, or that other processes that support the system in operation (maintenance, and failure analysis as key examples) will become smoother.

The spread of the variety within the system is important, sub-systems with high levels of variety (low commonality) will invariably be more complex that those with very little.

7.3.4.4 Summary of Element Complexity Measures

The available complexity measures fail to address some of the key questions that industry will need to answer. Elements which take in a series of different interface types, and so some work or processing and complex actions, are invariably more complex than those that do not. Elements such as pumps for example have electrical inputs (power and signal) and physical inputs (fluid), but the interfaces are very simplistic. Computer systems may have various data inputs, signal outputs, power inputs and more, these in invariably more complex elements and the element interfaces reflect this. From a system level, at perhaps a higher level of decomposition, an aircraft fuel management system may be fluid inputs, values, data buses, power, hydraulics, compressed air and more, and the variation and number of different interfaces to this element, are a good indicator of the complexity within that element. This is not really included in the complexity measures that are outlined above, but should be to indicate element complexity in functionality terms, this will be called Element Functionality Complexity.

Table 42 shows the appropriate complexity measures and their direct applicability to the questions outlined.

Complexity Measurement	What is the level of commonality within the system?	What are the functions of the elements?	What is the spread of functionality within the elements of the system?
Dimensions			X
Variables			X
Size			X
Variety	X		
Commonality	X		X
Element Functionality Complexity		X	X

Table 41 – Element complexity measures against element question criteria.

7.3.5 Management Complexity Measures

There are 7 complexity measures which either have a concept that can be applied or can be directly applied to engineered systems. Three areas were selected as relevant to measuring management complexity.

1. Variety - the variety within the skills and work packages.
2. Organisation and Knowledge - the knowledge and skill complexities.
3. Size - the size of the project and organisation.

As with all development programmes, there are management concerns, and management activities can lead to additional complexities within the system designs. Within management the key element is to minimise the resource requirement whilst ensuring that the complexity level of the system is kept at a manageable level. A management complexity measure should aim to answer the following questions:

- How can the resource required by the development process be minimised?
- How can the complexity of the organisation be minimised?
- How can the complexity of the product be minimised?
- Is the organisational capability capable of delivering the system?

Typically larger organisations are more difficult to manage than smaller ones, not only do development organisations contain engineering resources, but supply chain resources, support resources, training resources and operational resources. When

developing large scale complex systems the complexity can emerge from any one of these aspects of the product lifecycle, exist across some or be found within all of them. The following sections examine various complexity measures and their appropriateness for management complexity measures.

7.3.5.1 Variety as a Management Complexity Measure

The variety within the skill base has a potential link to the complexity within the system; generally those systems with high diversity in terms of skill requirements are more complex and interactive than those which are not. Interactions between engineering disciplines need careful systems engineering, the higher the diversity in the disciplines the more interaction between disciplines needs control. The NSI measure attempts to quantify this qualitatively, assessing the diversity in the skill set required to produce a system. NSI may need extension to include how the skill sets change throughout the lifecycle of the product, the initial diversity may be large but, there may not be a consistency between life cycle stages. In order to understand the complexity within the organisation, engineering discipline interactions need to be understood as interfaces within the organisation.

7.3.5.2 Organisation and Knowledge as a Management Complexity Measure

Cognitive complexity is a very useful measure when understanding the nature of your workforce. It does not however provide any useful data in terms of the product complexity of the system under development. Tools that provide a detailed understanding of the workforce skill set are already in use today, BAE SYSTEMS for example use EDY (Engineering Developing You) and PDY (Project management Developing You) tools to establish the capability that is available. This area of measurement steers towards the concept of organisational competency.

7.3.5.3 Size as a Management Complexity Measure

Size can be a factor when considering complexity within development organisations, the size of the developing organisations have increased with larger supplier bases and diversity. The PIA measure supplies this information in terms of the detailed design, assessing the size of the resource required to conduct this activity. Extending this to cover the size of the resource in different areas of product development such as

manufacture, design, testing, operation and training provides an insight into the difficulty of designing and managing an organisation capability that can deliver this.

7.3.5.4 Summary of Management Complexity Measures

Table 42 shows the appropriate complexity measures and their direct applicability to the questions outlined.

Complexity Measurement	How can the resource required by the development process be minimised	How can the complexity of the organisation be minimised?	How can the complexity of the product be minimised?	Is the organisational capability capable of delivering the system?
NSI - Number of Skills Involved	X	X	X	X
PIA - Persons Involved in Analysis (Size)	X	X	X	X
Cognitive Complexity (Competency)				X

Table 42 - Management complexity measures against management question criteria.

7.4 Cross Comparison of Complexity Measures

Table 43 shows the cross comparison of the complexity measures selected for requirements, decomposition, interfaces and management areas. There is a large amount of overlap and a proportion of the measures are redundant and provide little benefit. Those measures that are redundant are given a red text and are not pursued further within this thesis.

Requirement Complexity Measures	System Decomposition Complexity Measures	Interface Complexity Measures	Element Complexity Measures	Management Complexity Measures
Algorithmic Information Complexity	Ease of Decomposition	IBDF - Interdependencies Between Different Factors	Dimensions	NSI - Number of Skills Involved
Bennett's 'Logical Depth'	Information Gain in Hierarchically Approximation and Scaling	Connectivity	Variables	PIA - Persons Involved in Analysis (Size)
Commonality	Martins Measurement Set	Ease of Decomposition	Size	Cognitive Complexity (Competency)
Descriptive/Interpretative Complexity	Mutual Information	Martins Measurement Set	Variety	Size
Entropy	Number of Dimensions	Number of Dimensions	Commonality	
Horn Complexity	Number of Variables	Number of Internal Relations	Element Functionality Complexity	
Information		Number of Spanning Trees		
Irreducibility		Number of Variables		
Number of Dimensions		Size		
Number of In-equivalent Descriptions		Commonality		
Number of Variables		System Interface Nature		
Shannon Information				
Kolmogorov Complexity				
Rissanens Minimum Description Length (MDL)				

Table 43 - Comparison table of complexity measures.

There is obviously some commonality between the measures that have been selected, this commonality needs to be looked at to ensure that the measures are independent and they are in fact measuring different complexities within the systems engineering exercise.

Table 44 is a comparison of the measures in terms of their versatility across the overall complexity measurement scope, the X marks are placed where the measure has a direct application.

Complexity Measurement	Applicability to Different Measurement Stages				
	Requirement Complexity	System Decomposition Complexity	Interface Complexity	Element Complexity	Management Complexity
Cognitive Complexity (Competency)					X
Connectivity			X		
Commonality	X		X	X	
Descriptive/Interpretative Complexity	X				
Horn Complexity	X				
Element Functionality Complexity				X	
Information Gain in Hierarchically Approximation and Scaling		X			
Martins Measurement Set		X	X		
Mutual Information		X			
NSI - Number of Skills Involved					X
Number of Dimensions	X	X			
Number of In-equivalent Descriptions	X				
Number of Internal Relations			X		
Number of Spanning Trees			X		
Number of Variables	X	X			
PIA - Persons Involved in Analysis (Size)					X
System Interface Nature			X		

Table 44 - Complexity measure to application mapping.

The results do show that the majority of the measures have only one main application. There are of course some exceptions to this, and these are:

- Commonality – Useful when considering the level of variety in a system and that can exist within requirements, elements and links.
- Martin's Measurement Set – Useful in both decomposition and interface complexities as it is a measure of system design complexity.

- Number of Dimensions – There is a possibility that dimensions could be applied to every application, but this has been limited to just requirement and decomposition stages due to its easier application.
- Number of Variables – Like dimensions, potentially variables are applicable to all applications (variables are the measured item after all) but in terms of complexity measurement again they have been limited to requirement and decomposition complexities.

There are fewer management and interface measures than for decomposition and requirement complexities, but this measurement set can be expanded later on to include derivations of measures or include new ones. For now the set has the higher focus in areas of systems engineering that are perhaps more critical when handling complexity in design.

The refined measures above require allocation and exploration within the systems engineering field. Problem case studies of complex system development programmes need to be carried out to test the measurement applications that have been suggested here. The relationships between the measures, solutions, problems and approaches will form the complexity understanding for the thesis.

There is room for expansion in the detailed applicability of the measures to the industrial domain and their feasibility. Without this, understanding what can be gained from the measures and from which lifecycle stage they can be applied is difficult. Those measures with defined relationships from the industrial examples should be expanded to include instructions of application:

- What they measure
- How to measure it
- How to interpret the result
- What the limitations of the measure are

With this information, industry has a better understanding of how complexity measures fit within their problems, development stages and how they can be applied effectively with a correct interpretation and understanding of the limitations.

7.5 Turning Selected Measures into Useful Measures

The measures selected above need to be applied to engineering systems to be of any use to industry. Some of the measures selected can not be easily calculated with the data available from engineering systems; interface details, element details, system functional diagrams, information flow. Consequently the down selected measures must be assessed in terms of the practicality of their application. In some cases concepts taken from the measures that were down selected may be applied rather than attempt a difficult direct application of the measure.

The following sections detail the assessment of practicality of the measures down selected for application within the complexity analysis tool.

7.5.1 Measures Selected

The measures down selected must be mapped against data that can be collected from the engineering programmes. Table 45 shows the down selected measures from Table 43, and analyses the difficulties in their application to systems.

Complexity Measurement	Ease of Application		Comments
	Good	Bad	
Cognitive Complexity (Competency)		X	It would be difficult to calculate a combined cognitive complexity for everyone working on a project, due to the project size and the number of people working on that project.
Connectivity	X		Connectivity of the system could be applied easily, as elements and their interfaces are well documented in engineering programmes.
Commonality	X		Commonality within systems can be easily applied as common components or components that have a similar function can be easily identified within the system from documentation.
Descriptive / Interpretative Complexity		X	Descriptive and interpretative complexity is difficult to calculate, there is no set method of calculation using data. A qualitative measure at best and application is limited. This requires the measurement of the number of requirements and potential solutions to those requirements.
Horn Complexity		X	Similar to descriptive complexity, it is hard to measure horn complexity within systems. The only measure that would be appropriate would be the number of requirements required to describe the system. The application is difficult and the benefit from the measure is also limited.
Element Functionality Complexity	X		Easily applied to systems, as a clear understanding of element functionality is known.
Information Gain in Hierarchically Approximation and Scaling	X		Can be applied to systems, by changing the level of abstraction in the system and analysing the complexity for both levels in the same manner with a comparison between the two.
Martins Measurement Set	X		Can be applied to systems and already has been applied.
Mutual Information		X	Difficult to apply to systems as measuring the level of random behaviour within a system. In engineering systems, random behaviour is usually a problem, as a result it is minimised. The abstract of the measure may be used to understand how different grouping of sub-system components and interfaces changes overall system complexity.

Complexity Measurement	Ease of Application		Comments
	Good	Bad	
NSI - Number of Skills Involved	X		The number of different skills of the members of a team working on a programme should be found within the organisational structure charts for that programme.
Number of Dimensions		X	The number of dimensions within projects is vague. Dimensions may be elements, commonality, interfaces, etc. These are calculated by other measures within the set and this is therefore redundant.
Number of In-equivalent Descriptions		X	Similar to descriptive / interpretative and horn complexity, without finding every possible method of describing a system using requirements, this measure cannot be calculated. In large scale systems consisting of 10,000 or more requirements the time taken to calculate the potential outputs from that requirements document would be large.
Number of Internal Relations	X		The number of internal relationships should be found within the systems interface specifications.
Number of Spanning Trees		X	Spanning trees can be applied to systems using the interface data and functional flow diagrams for the information that passes along those interfaces.
Number of Variables	X		Easily measured in systems, in particular within design problems such as optimisations.
PIA - Persons Involved in Analysis (Size)	X		The personnel working on a project is generally known within engineering programmes. This information should be obtainable and the measure should be calculated easily.
System Interface Nature	X		The nature of the system interfaces can be found within the system interface specifications.

Table 45 - Analysis of measure application difficulty.

Table 45 shows the down selected complexity measures and their ease of use against potential sources of information within engineering programmes that could be used to calculate them. In addition it was also recognised that there may be problems in collecting data to provide input for these measures, therefore the first set of case study measures were derived using the principles from this measurement set.

7.5.2 Derivation of Measure Set for Case Study Exploration

Measures or a measurement set needs development from the measures that are useful to industry as they answer questions about complexity in systems or provide some quantification of complexity characteristics. The measures that are appropriate and that are practically applicable to engineering systems were applied, and if not directly applicable an abstraction was made.

The measurement sets have been divided up into those measures that are appropriate to sub-systems, and those measures that are appropriate to the interfaces between those sub-systems, and are shown within the following two tables. In each case a description of the measure is outlined and the applicability or relationships to those measures outlined above are shown.

Table 46 shows those complexity measures (within the CCCS) that are applicable to the overall system and the sub-systems contained within it which have been derived from the down selection of measures within the previous section. The first column shows the name of the derived measure calculated for the case studies, the second column shows the description of that measure. The final column shows the measures within the CCCS currently, that have been used as a concept to derive the measure within column one.

Calculated Measures	Description	The Measure or Measure Concept Used
Connectivity Multi	The number of elements that are connected regardless of the direction of travel of the information, matter or energy. That is if A and B are connected and A passes information to B, a connection exists in the model between A and B, and is counted as a connection for both elements.	Connectivity, Information Gain in Hierarchically Approximation and Scaling, Martins Measurement Set, Number of Internal Relations, Number of Spanning Trees, System Interface Nature
Connectivity Single	The number of connections between elements but taken from a directional view, i.e. there is a connection in which information from A is passed to B, this will be a single connection from A to B and only counted within A.	
Connectivity Single %	The percentage of the total possible connections between elements.	
Link Number	The total number of links within the sub-system or system being analysed	
Link Complexity	The total number of links within the sub-system or system multiplied by the link complexity factor allocated.	
Number of Link Types	The number of different link types within the system or sub-system.	Commonality, System Interface Nature
Element Number	The number of elements within the system or sub-system.	Commonality, Element Functionality Complexity, Information Gain in Hierarchically Approximation and Scaling, Martins Measurement Set
Element Complexity	The number of elements within the system or sub-system multiplied by the complexity type for those elements.	
Number Of Element Types	The number of different element types within a system or sub-system.	
Skill Number	The number of different skills within the organisational body developing the system.	Commonality, NSI - Number of Skills Involved
Supplier Total	The number of suppliers within the organisational body developing the system.	Commonality
Personnel Total	The number of personnel in total working on the system within that company.	Commonality, PIA – Persons Involved in Analysis (Size)
Commonality Links	The number of link duplication within the system.	Connectivity, Commonality, Information Gain in Hierarchically Approximation and Scaling, Martins Measurement Set, Number of Internal Relations, Number of Spanning Trees, System Interface Nature
Commonality Link Types	The number of link type duplication within the system.	
Commonality Elements	The number of element duplication within the system.	Commonality, Element Functionality Complexity, Information Gain in Hierarchically Approximation and Scaling, Martins Measurement Set
Commonality Element Types	The number of element type duplication within the system.	

Calculated Measures	Description	The Measure or Measure Concept Used
Commonality Skills	The number of skill duplication within the development organisation for the system.	Commonality, NSI - Number of Skills Involved
Commonality Suppliers	The number of mutual suppliers within the development organisation for the system.	Commonality
Commonality Personnel	The number of personnel that are shared between sub-systems.	Commonality, PIA - Persons Involved in Analysis (Size)

Table 46 - Model calculated measures for complexity model sub-systems/systems.

Table 47 shows those measures that were derived for the system interfaces.

Calculated Measures	Description	The Measure or Measure Concept Used
Connectivity	The connectivity between the two sub-systems as a function of what is passed between them and the number of nodes within each sub-system. Direction of travel of the interface is ignored.	Connectivity, Information Gain in Hierarchically Approximation and Scaling, Martins Measurement Set, Number of Internal Relations, Number of Spanning Trees, System Interface Nature
Connectivity %	The connectivity between the two sub-systems as a function of what is passed between them and the number of nodes within each sub-system as a percentage of the maximum number of connections that could exist. Direction of travel is ignored.	
Connectivity % Directional	The connectivity between the two sub-systems as a function of what is passed between them and the number of nodes within each sub-system as a percentage of the maximum number of connections that could exist. Direction of the interface is taken into account.	
Link Number	The number of links between the sub-systems.	
Link Complexity	The number of links multiplied by the link factor for those links between the sub-systems.	
Number of Link Types	The number of link types between the sub-systems.	Commonality, System Interface Nature
Commonality Link Types	The number of mutual link types between the sub-systems.	Connectivity, Commonality, Information Gain in Hierarchically Approximation and Scaling, Martins Measurement Set, Number of Internal Relations, Number of Spanning Trees, System Interface Nature
Commonality Links	The number of mutual links between the sub-systems.	

Table 47 - Model calculated measures for complexity model interfaces.

These measures will be applied to a system within the complexity analysis tool and used to determine the nature of the complexity of a system in terms of complexity characteristics and their quantities.

Table 48 and Table 49 show the links each complexity measurement derived has with aspects of the CAF. The CAF shows that measures may link to concepts or classifications and to definitions of complexity.

Calculated Measures	Description	Links within the Complexity Framework
Connectivity Multi	The number of elements that are connected regardless of the direction of travel of the information, matter or energy. That is if A and B are connected and A passes information to B, a connection exists in the model between A and B, and is counted as a connection for both elements.	Definitions; Intrinsic Complexity, Intricacy, Coupling, Difficult to Predict or Model, Size (the higher the level of connectivity, the more coupling is likely between elements, and depending on the intrinsic complexities of the interfaces, the harder it is to produce an accurate model) Classifications: Hierarchical, Non-Hierarchical (those systems that have very high levels of connectivity, will more than likely exhibit non-hierarchical structures, these are generally more complex than hierarchical structures).
Connectivity Single	The number of connections between elements but taken from a directional view, i.e. there is a connection in which information from A is passed to B, this will be a single connection from A to B and only counted within A.	
Connectivity Single %	The percentage of the total possible connections between elements.	
Link Number	The total number of links within the sub-system or system being analysed	
Link Complexity	The total number of links within the sub-system or system multiplied by the link complexity factor allocated.	
Number of Link Types	The number of different link types within the system or sub-system.	
Element Number	The number of elements within the system or sub-system.	Definitions; Intrinsic Complexity, Variety, Difficult to Predict or Model, Size (the higher the number of elements and the variety within those elements, along with higher intrinsic complexities, the more difficult the system is to model and predict) Classifications; Hierarchical, Non-Hierarchical (information flows from and to elements depend on the capabilities of that element and so have an effect on the overall system structure)
Element Complexity	The number of elements within the system or sub-system multiplied by the complexity type for those elements.	
Number Of Element Types	The number of different element types within a system or sub-system.	
Skill Number	The number of different skills within the organisational body developing the system.	Definitions; Variety (the variety within skills people and suppliers has an effect on overall system complexity. The more variety, the more difficult the system to understand). Classifications: No link.
Supplier Total	The number of suppliers within the organisational body developing the system.	
Personnel Total	The number of personnel in total working on the system within that company.	
Commonality Links	The number of link duplication within the system.	
Commonality Link Types	The number of link type duplication within the system.	
Commonality Elements	The number of element duplication within the system.	
Commonality Element Types	The number of element type duplication within the system.	
Commonality Skills	The number of skill duplication within the development organisation for the system.	
Commonality Suppliers	The number of mutual suppliers within the development organisation for the system.	
Commonality Personnel	The number of personnel that are shared between sub-systems.	

Table 48 – Calculated measures for overall system and sub-systems and their links to the Complexity Framework.

Calculated Measures	Description	Links within the Complexity Framework
Connectivity	The connectivity between the two sub-systems as a function of what is passed between them and the number of nodes within each sub-system. Direction of travel of the interface is ignored.	Definitions; Intrinsic Complexity, Intricacy, Coupling, Difficult to Predict or Model, Size (the higher the level of connectivity, the more coupling is likely between elements, and depending on the intrinsic complexities of the interfaces, the harder it is to produce an accurate model) Classifications: Hierarchical, Non-Hierarchical (those systems that have very high levels of connectivity, will more than likely exhibit non-hierarchical structures, these are generally more complex than hierarchical structures).
Connectivity %	The connectivity between the two sub-systems as a function of what is passed between them and the number of nodes within each sub-system as a percentage of the maximum number of connections that could exist. Direction of travel is ignored.	
Connectivity % Directional	The connectivity between the two sub-systems as a function of what is passed between them and the number of nodes within each sub-system as a percentage of the maximum number of connections that could exist. Direction of the interface is taken into account.	
Link Number	The number of links between the sub-systems.	
Link Complexity	The number of links multiplied by the link factor for those links between the sub-systems.	
Number of Link Types	The number of link types between the sub-systems.	
Commonality Link Types	The number of mutual link types between the sub-systems.	Definitions; Variety (the variety within skills people and suppliers has an effect on overall system complexity. The more variety, the more difficult the system to understand). Classifications: No link.
Commonality Links	The number of mutual links between the sub-systems.	

Table 49 - Calculated measures for system interfaces and their links to the Complexity Framework.

The tables above show how the measures relate back to the links to the CCCS attributes, using the links between those attributes within the CAF.

7.6 Conclusion

After an extensive look at complexity measures, their attributes and relevance to complexity characteristics (that when quantified help understand complexity within developing or developed systems) it is clear that some measures are more appropriate than others. It is also clear than no one measure of complexity is good enough, and a combination of measures is required in order to provide a comprehensive understanding of the complexity that exists within the system.

Complexity cannot be measured in a manner that is meaningful if the measures are composites, in the same way that a rectangle's area cannot tell you its height and width; however, in order to understand how that rectangle fits within two dimensional space those two values are vital. Complexity is similar - connectivity alone is not enough to understand the system without knowing the complexity of the links, complexity of elements within the system is subject to commonality and maturity. As

a result a series of measures that span different aspects of the product, such as the connectivity, the commonality, the number people on the programme, the diversity as a whole help to understand the complexity characteristics of that system. But the real knowledge of the system is not just from the measures, but from the understanding of their links and relationships.

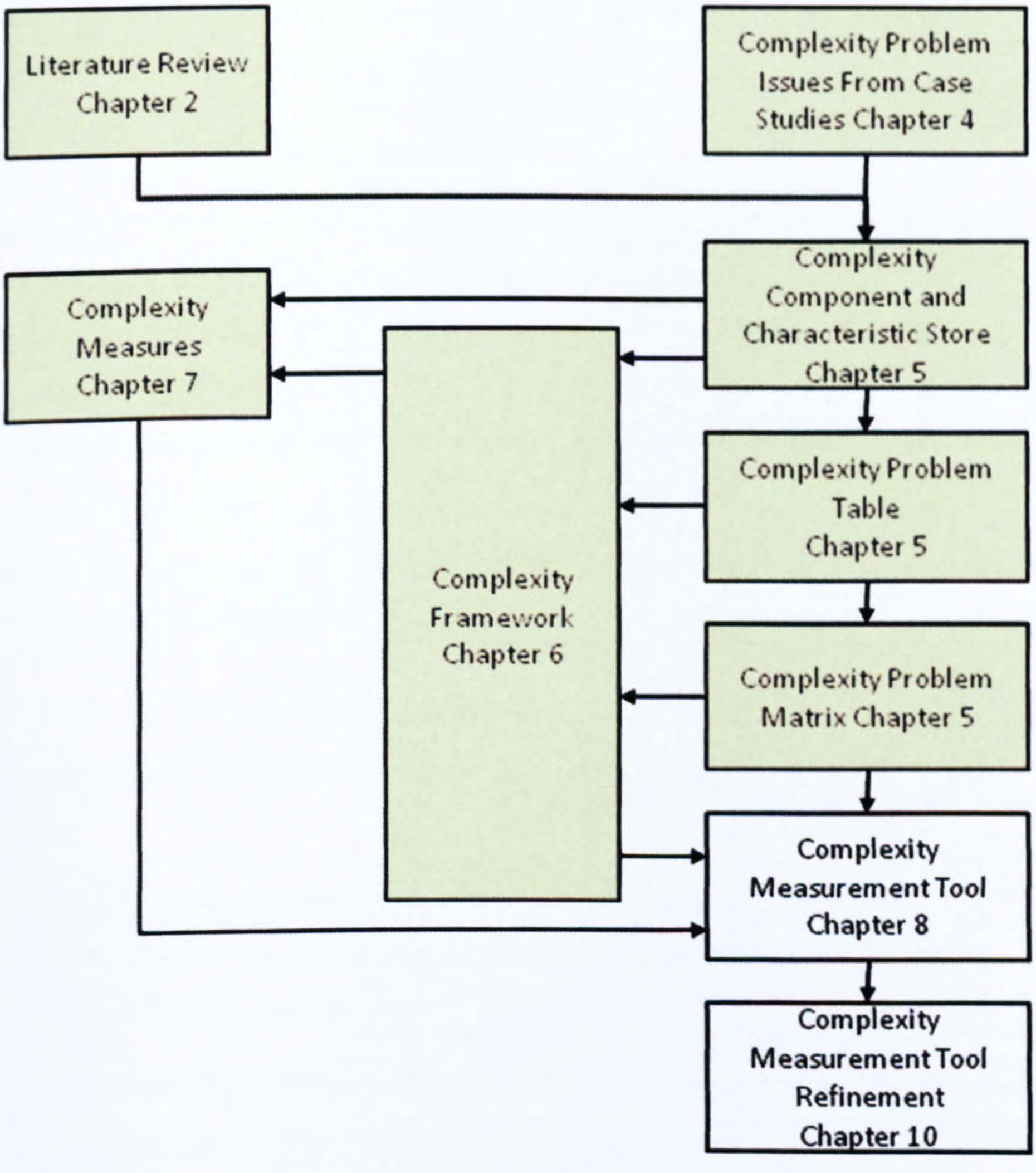


Figure 37 - The layout of the work and the thesis outputs roadmap.

As the roadmap within Figure 39 shows, measures have been identified that could be of use within engineering systems, and appropriate measures derived from these that can more easily take advantage of the data that is available within industry development programmes. These measures are able to provide a comprehensive view of the systems complexity characteristics. These measures when added to an analysis

tool provide industry with the ability to understand their system in terms of complexity characteristics and also with the links within the complexity matrix understand the potential problems that may arise within their system.

8 Industrial Case Study Data and the Complexity Measurement Tool

This chapter takes the information regarding complexity issues and their mappings to complexity characteristics in chapters 4 and 5, and then using the measures that are useful from chapter 7 creates an analysis tool for analysing complexity in systems (including the sub-systems and interfaces)'

8.1 Introduction

This chapter tests the derived measures (chapter 7) within a measurement tool using three new case studies. New case studies were necessary as the data required to populate the measurement tool was not easily obtained from those used to further understand problems in industry. These new case studies are outlined (see section 8.2) and the data collected from them also outlined along with collection techniques.

The data collected was then entered within an analysis tool. The details of the analysis tool regarding the inputs, how the tool is constructed, what the tool measures, how it is measured, and the tool outputs are found within section 8.3.

8.2 Industrial Case Studies Introduction

Three systems were analysed and data extracted from them to populate the analysis tool.

- Naval Command System Trainer – A training system consisting of a configurable number of terminals that interact with real navy command systems to improve realism.
- Naval Command System – A command system for navy ships, consisting of configurable workstations.
- Aircraft Fuel System Test Rig – A fuel pumping test rig consisting of pumps, sensors, pipe work, tanks and electronic interfaces with a computer controller.

The background details for the case studies in turn can be found below.

8.2.1 Naval Command System Trainer (NCST)

A training system for naval systems consisting of a number of training workstations (of which the number is configurable, 1 to 6 were used in this exercise) which in turn connect to a central interface connecting them to the main vessel command systems. This connection enables accurate simulations and the crew to train using the real equipment in the real environment rather than substitutions.

8.2.2 Naval Command System (NCS)

A naval command system essentially comprising of the three workstations and a network between them with an interface to the onboard systems of the vessel within which it resides. The three workstations within the system are identical but can be programmed to perform different tasks. Each workstation can be configured to *own* specific tasks and other workstations may access the information associated with this task in order to carry out their own, *owned* tasks.

8.2.3 Aircraft Fuel System Test Rig

A small system used to test problem diagnosis and health monitoring tools consisting of a power unit, water tanks, pumps, valves and sensors which can be controlled remotely using a computer or manually from a panel. There is a software platform built within Labview (National Instruments) that drives the rig and processes the output data, unfortunately for now the software models are not entirely accurate and could not be used in the analysis.

8.3 Complexity Tool Introduction and Description

The complexity analysis tool is an Excel based workbook consisting of a number of spreadsheets; input sheets, input tables, processing sheets, output summary sheets. Each analysis workbook for each case study is identical apart from the data used within it. The model calculates various measures (see chapter 7) of complexity based on the data inputted and displays the results in a summary sheet, the measures contained within the tool are detailed within section 8.3.2. The analysis tool description is detailed within g sections 8.3.1 to 8.3.5.

The tool calculates complexity measures for the system as a whole, and also for the sub-systems and interfaces between sub-systems separately to improve the understanding of complexity within the system.

8.3.1 Mechanisms to Represent Different Interfaces, Elements and Skills

The provision of more detailed information of the composition or nature of the elements, skills of the personnel, and interfaces within the engineered systems under consideration or required to produce it provide a better understanding of that system. Interfaces may be pipes for fluid, data buses or electrical power, elements may be pumps, power converters, sensors or processing units, and the skills required to develop the system may be computer programming, aerodynamic modelling or electronic design. In order to incorporate this knowledge into the model a naming system was created for system elements, interfaces and the skills required to produce the system. These names would form an input to the analysis tool and enable commonalities to be established (most of the system consisting of the same interface types, etc.).

8.3.1.1 Element Types

Names were created for various different elements within the case study systems under analysis. The names of each element reflect their 'type' and function, and are divided into two components (TYPE, FUNCTION); the first component indicates the element 'type' as follows:

- MAT indicates the element deals with matter.
- ENGY indicates the element deals with energy.
- DATA OUT, DATA IN indicates the element deals with data, either exporting that data out, or importing data in.
- DATA PROCESSING indicates the element deals with data and processes it.
- SWITCH indicates the element is a basic switch style (no second component required as it simply activates on or off).

The second component of the name indicates the element function as follows:

- DIST indicates a distribution of.
- CONV indicates a conversion of.
- CONS indicates a consumption of.
- TRANS indicates the element is a conduit for transporting.
- GEN indicates a generation of.
- BOOLEAN, VALUE, BUS, ARRAY indicate the nature of the data within the element.

An example of an element type may be MAT DIST, which means a Matter Distributor (perhaps a pump), another example is a DATA OUT VALUE, indicates the element outputs a data value (perhaps a sensor with a numerical data out indicating a temperature). Table 50 shows the complete list of element names consisting of the 'types' and their functions contained within the analysis tool.

Function	Element Name
Elements Dealing with Matter	MAT DIST
	MAT CONV
	MAT TRANS
	MAT CONS
Elements Dealing with Energy	ENGY DIST
	ENGY CONV
	ENGY GEN
Elements Dealing with Data and Data Processing	DATA IN BOOLEAN
	DATA IN VALUE
	DATA IN BUS
	DATA IN ARRAY
	DATA OUT BOOLEAN
	DATA OUT VALUE
	DATA OUT BUS
	DATA OUT ARRAY
	DATA PROCESSING BOOLEAN
	DATA PROCESSING VALUE
	DATA PROCESSING BUS
	SWITCH

Table 50 - Element names.

8.3.1.2 Interface Types

Names were created for various different interfaces within the case study systems under analysis. The names of each interface reflect their 'type' and the data nature (only applies to data interfaces), and are divided into two components (TYPE, DATA NATURE); the first component indicates the interface 'type' as follows:

MATTER, an interface transmitting matter, perhaps a fluid, or solid fuel, or hydraulic system.

ENERGY, a transfer of electrical energy.

DATA, a transmission of data.

The second component of the name indicates the interface data nature as follows:

BOOLEAN, a simple on off interface, like a switch.

VALUE, a data interface of a data value.

BUS, a data bus interface.

ARRAY, an interface consisting of an array of values.

8.3.1.3 Interface and Element Example

An example of an interface may be DATA BOOLEAN, which means a data interface consisting of data of a Boolean on/off nature. The interface may be an electrical signal from a pump telling a computer it is either running of shut down. Table 51 shows the complete list of interface names consisting of the 'types' and the data nature for data interfaces contained within the analysis tool.

Interface Type	Interface Names
Interfaces Transmitting Matter	MATTER
Interfaces Transmitting Energy	ENERGY
Interfaces Transmitting Data	DATA BOOLEAN
	DATA VALUE
	DATA BUS
	DATA ARRAY

Table 51 - Interface names.

When entering information into the analysis tool it is recognised that in most cases an element or interface cannot be labelled with just one of the names found within Table 50 or Table 51. In fact multiple interface types/data natures and element

types/functions can exist for any one system element or interface. An example might be a pump that internally has a sensor which detects pressure, speed of the pump, and transmits these on a data port. The pump also obviously pumps fluid at a demanded speed sent through a separate connection. The interfaces and elements could be described as:

The following two tables (Table 52 and Table 53) show the interfaces and elements within the pump example written in the naming formats outlined above.

Interface Name	IN/OUT	Description
DATA ARRAY	OUT	The data port transmitting the pump pressure and speed.
DATA VALUE	IN	Demanded speed signal sent to the pump.
MATTER	OUT	The fluid leaving the pump.
MATTER	IN	The fluid entering the pump.
ENERGY	IN	Power supply for the pump.

Table 52 - Pump example for interfaces.

Element Name	Comment
MAT TRANS	Moves the fluid through the pump, transmitting the fluid.
DATA IN VALUE	Takes in data values, in this case the speed demand.
DATA OUT ARRAY	Outputs a data array, in this case the speed and pressure data.

Table 53 - Pump example for elements.

Multiple interface and element types, functions or data natures may be entered for one interface or one element. In some cases multiple element and interface descriptions are required in order to assure the complete functionality of that element or interface is captured, the example above demonstrates this.

8.3.1.4 Skill Types

Skill requirements are also categorised within the analysis tool. These skills are those required to develop an element of a system, it is assumed the interfaces form part of the element development and therefore are included in the skill set to develop the element.

The skill types used are as follows: Mechanical, Electrical, Computing, Management, Systems Engineering, Aerodynamic, Control, Materials, Fluid Dynamics, Thermodynamics, and Software Engineering.

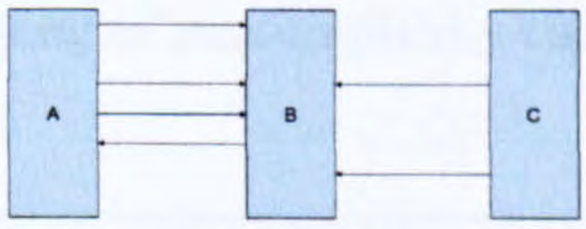
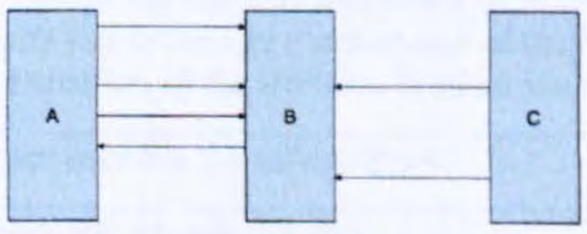
These skills were used as they represent the breadth of the disciplines required to develop engineered systems.

8.3.2 Analysis Tool Calculated Measures used within the Tool

The following section contains the measure details used within the analysis tool, and their method of calculation. Table 54 shows the measures used and their method of calculation for the system as a whole and the sub-systems contained within it. These measures were the result of a down selection process detailed within section 7 (from Table 43 and Table 44). The factors used within these measures can be found within section 8.3.3

- Table 54 describes the complexity measures used within the whole system and the sub-systems used to determine their complexity characteristics.
- Table 55 describes the complexity measures that are used in determining the complexity characteristics of the system and sub-system interfaces.

Further analysis for each sub-system is also conducted independently of the overall system. This is done so the overall complexity of the system can be understood, and the origin of that complexity and the cause (within sub-systems) of that complexity also characterised.

Calculated Measures	Description
Connectivity Multi	The number of connections between elements is recorded. Multiple links may occur between two elements in one direction, if more than one interface exists between those elements. For example if A were a pump and B were controller if A sent speed, temperature and pressure to B and B sent a demand to A, 3 links would be recorded from A to B and 1 for B to A. Example:  In this case A to B would contain 3 links, B to A would have 1 link, B to C would have no links and C to B would have 2 links.
Connectivity Single	If a connection or connections exist between two elements in a single direction this is recorded as one link. Multiple links in a single direction are considered as one link between those elements in that direction. Using the same example as above, A to B would contain 1 link, B to A would have 1 link, B to C would have no links and C to B would have 1 link.
Connectivity Single %	The percentage of the total possible directional connections between elements in the system.  Total number of elements is 3, the maximum number of directional connections is 6 (1 - AB, 2 - BA, 3 - AC, 4 - CA, 5 - BC, 6 - CB), the number of connections in this system is 3 (1 – AB, 2 – BA, 3 – CB), therefore the connectivity single % is 3 / 6 = 0.5 or 50%.
Link Number	The total number of links within the sub-system or system being analysed.
Link Complexity	The total number of links within the sub-system or system multiplied by the link complexity factor allocated.

Calculated Measures	Description
Number of Link Types	The number of different link types within the system or sub-system.
Element Number	The number of elements within the system or sub-system.
Element Complexity	The number of elements within the system or sub-system multiplied by the complexity type for those elements.
Number Of Element Types	The number of different element types within a system or sub-system.
Skill Number	The number of different skills within the organisational body developing the system.
Supplier Total	The number of suppliers within the organisational body developing the system.
Personnel Total	The number of personnel in total working on the system within that company.
Commonality Links	The number of link duplication within the system.
Commonality Link Types	The number of link type duplication within the system.
Commonality Elements	The number of element duplication within the system.
Commonality Element Types	The number of element type duplication within the system.
Commonality Skills	The number of skill duplication within the development organisation for the system.
Commonality Suppliers	The number of mutual suppliers within the development organisation for the system.
Commonality Personnel	The number of personnel that are shared between sub-systems.

Table 54 - Model calculated measures for complexity model sub-systems/systems.

Table 55 shows those measures used when determining the complexity of interfaces between elements. The factors used within these measures can be found within section 8.3.3.

As for the elements above the analysis of interfaces is conducted at a system and also sub-system level to provide an understanding of the complexity characteristics and origins of complexity within the system.

Calculated Measures	Description
Connectivity %	The connectivity between the two sub-systems as a function of what is passed between them and the number of nodes within each sub-system as a percentage of the maximum number of connections that could exist. Direction of travel is ignored.
Link Number	The number of links between the sub-systems.
Link Complexity	The number of links multiplied by the link factor for those links between the sub-systems.
Number of Link Types	The number of link types between the sub-systems.
Connectivity % Directional	The connectivity between the two sub-systems as a function of what is passed between them and the number of nodes within each sub-system as a percentage of the maximum number of connections that could exist. Direction of the interface is taken into account.
Commonality Link Types	The number of mutual link types between the sub-systems.
Commonality Links	The number of mutual links between the sub-systems.

Table 55 - Model calculated measures for complexity model interfaces.

These measures are calculated using various excel spread sheet logic, and displayed within a summary sheet (see section 8.3.5).

8.3.3 Analysis Tool Factors used in Measure Calculations

Some of the complexity measures calculated within the analysis tool employ the use of factors in their calculation. Within the analysis tool set for the case studies a common set of factors was produced for:

- Interfaces (see Table 56); as in section 8.3.1.2.
- Elements (see Table 57); as in section 8.3.1.1.
- Skills (see Table 58)

The interface factors relate to the intrinsic complexity of the interface type. A fluid flow in most cases is less complex than a data bus link, or an array link. The factors are self generated based on the perceived complexity of the interfaces and given a score between 1 and 5, 5 being high intrinsic complexity. Table 56 shows the values for the factors for each interface type within the analysis tool.

Interface Type	MATTER	ENERGY	DATA VALUE	DATA BUS	DATA BOOLEAN	DATA ARRAY
Factor	1	2	3	5	2	4

Table 56 - Interface type factors.

Table 57 shows the factors set for the element types, data handling elements generally contain some electronics and were seen as more complex than matter or energy elements. Those elements handling data and data processing are intrinsically more complex than those that handle simple Boolean or single values. As for the interfaces, the elements too are self generated based on the intrinsic complexity considered applicable to the element, they are then scored between 1 and 4, 4 being the higher level of complexity. The table shows the complete factor list for the element types used in the analysis tool for calculation of complexity measures.

Element Type	Factor
MAT DIST	1
MAT CONV	1
MAT TRANS	1
MAT CONS	1
ENGY DIST	1
ENGY CONV	1
ENGY GEN	1
DATA IN (BOOLEAN)	2
DATA IN (VALUE)	2

Element Type	Factor
DATA IN (BUS)	3
DATA OUT (BOOLEAN)	2
DATA OUT (VALUE)	2
DATA OUT (BUS)	3
DATA PROCESSING (BOOLEAN)	4
DATA PROCESSING (VALUE)	3
DATA PROCESSING (BUS)	4
SWITCH	2

Table 57 - Element type factors.

Table 58 shows the skill type factors within the analysis tool. There was no clear method of identifying which skill was more complex than the other as a result although the factors are required by the analysis tool to calculate the values of measures, they were all set to 1 to ensure they did not affect the result.

Skill Type	Factor
Mechanical	1
Electrical	1
Computing	1
Management	1
Systems	1
Aerodynamic	1
Control	1
Materials	1
Fluid Dynamics	1
Thermodynamics	1
Software	1

Table 58 - Skill type factors.

The factors were used in the calculation of various measures within the tool, they were consistent through all case studies, and are easily modified if required.

8.3.4 Analysis Tool Data for Calculations and Sources

This section details the required data the tool needs in order to calculate the various complexity measures contained within it. The model data that is required to complete the tool is shown within Table 59.

Input Data	Description
Interface Data	Interface data comprising of: • The number of interfaces between two components (system/sub-system elements). • The nature of the interfaces between the two components (system/sub-system elements); data Boolean, data value, data bus, etc. • The nature of the direction of the various data flows between components (system/sub-system elements). • The number of exact interface duplicates.
Element Data	Element data comprising of: • The number of elements within the system/sub-system. • The nature of the element within the system/sub-system; data processor, matter transfer, energy transfer, etc. • The number of exact element duplicates.
Personnel Data	Personnel data comprising of: • The number of personnel required to develop an element. • The number of personnel performing multi roles within that development. (this information so far has been difficult to collect, and detail has been omitted until this information is mature)
Skill Data	Personnel skill data comprising of: • The number of skills required to develop each sub-system element. • The amount of skill duplication.
Supplier Data	Supplier data comprising of: • The number of suppliers required to produce a system/sub-system/element. • The number of suppliers that are duplicated within the system/sub-system/element. (this information so far has been difficult to collect, and detail has been omitted until this information is mature)

Table 59 - Data required to complete the models.

The sources must provide the relevant information for the model to use in order to produce the output including:

- Interfaces, the nature of those interfaces and commonality between them.
- Elements, the nature of those elements and commonality between them.
- The system structure.
- Supplier Information
- Organisation Information

Table 60 shows how these sources can be used to collect data from the case studies to populate the analysis tool.

Data Source	Potential Use
System Functional Diagrams	Provide information regarding the functions of elements within the system so their types can be determined, along with their interactions. However functional diagrams are not necessarily the same as the structure of the actual system so care is required. Functional diagrams also enable the identification of common functions and parts. See Interface and Element Data in Table 59.
System Specifications	System specifications enable the actual construction (not just from a functional perspective) of the system to be understood and further details of elements to be understood. See Interface and Element Data in Table 59.
Interface Specifications	System interface specifications enable the interfaces between the elements to be understood and their types identified. See Interface and Element Data in Table 59.
System Diagrams	Enable the actual construction of the system to be understood in terms of elements and interfaces. See Interface and Element Data in Table 59.
Supplier Lists	Lists of suppliers provide information regarding how many suppliers there are, what they supply for the system and which of those suppliers are common. See Supplier Data in Table 59.
Organisational Charts	Enable and understanding of the skill sets and the distribution of people within the organisation so any personnel working on more than one portion of the system can be identified. See Personnel and Skill Data in Table 59.
Organisational Lists	Organisational lists show who works on which portion of the programme. See Personnel and Skill in Table 59.

Table 60 - Data sources for the analysis tool.

This data would be collected from the relevant business units and then used to build the data inputs to the analysis tool. Using this data and the inputted factors (see section 8.3.3) the tool is able to calculate the complexity measures for the overall system and the system sub-systems.

8.3.5 Analysis Tool Output

The tool generates outputs for the system as a whole and the sub-systems individually. The interfaces between the sub-systems are also analysed by the tool in more detail. The output from the tool can be divided into three key shown within Table 61

Output	Description
Overall System Complexity Characteristics	Details the complexity characteristics of the system as a whole, considering the system sub-systems as elements with their respective properties (considered as black boxes with inputs and outputs, the internal complexities of the sub-systems are ignored) and interfaces between them with their respective properties.
Sub-System Complexity Characteristics	Details the complexity characteristics of each sub-system, the elements within it and their respective properties, and the interfaces between them and their respective characteristics.
Sub-System Interface Complexity Characteristics	Details the complexity characteristics of the interfaces between the sub-systems.

Table 61 - Model outputs for complexity model.

The system overall output considers each sub-system as a black box element with a function with inputs and outputs.

8.4 Conclusion

The complexity analysis tool aims to provide answers to the questions industry wishes to answer which are outlined within section 7.1.1 and repeated below.

In a technical or hardware sense, which area of a system under development is the cause of the complexity within the system?

The tool will enable an understanding of the complexity within the system and the characteristics of that system its sub-systems and interfaces. These characteristics and their respective quantities calculated by the tool enable the user to understand the cause or causes of complexity within that system.

What are the future problems that this development programme is likely to exhibit?

The causes of the complexity coupled with the problem table and complexity matrix within sections 5.2.3 and 5.2.1 enable a translation between complexity characteristics and their prevalence, identified within the analysis tool output and potential problem issues.

Is the complexity manageable?

A difficult question to answer, even when the tool is coupled with the problem table and complexity matrix. The linking of different attributes with the complexity matrix and the problem table may provide clues, and the levels shown of the complexity characteristics provide an indication of how difficult or manageable the complexity within the system is.

How does this system compare with other systems in terms of complexity?

Comparison between complexity characteristics and their levels in systems is possible with a large enough sample. Once levels are established for different systems, or types of system then future systems can be assessed against these to estimate if the complexity is too high, or has the characteristics and quantities one would expect.

What effect would the introduction of another element into the system have in terms of the overall complexity?

This can be easily determined by the model; additional sub-systems can be added, or additional elements for sub-systems can be added, along with interfaces. The complexity characteristics can then be re-analysed and the differences quantified.

The tool created to analyse complexity within the case studies is a comprehensive tool covering most of the aspects of complexity discussed in previous chapters. The tool is highly expandable and can accommodate more measures if required. Not only is the tool expandable, the factors used can be reconfigured easily and the new results produced instantly reflect those changes.

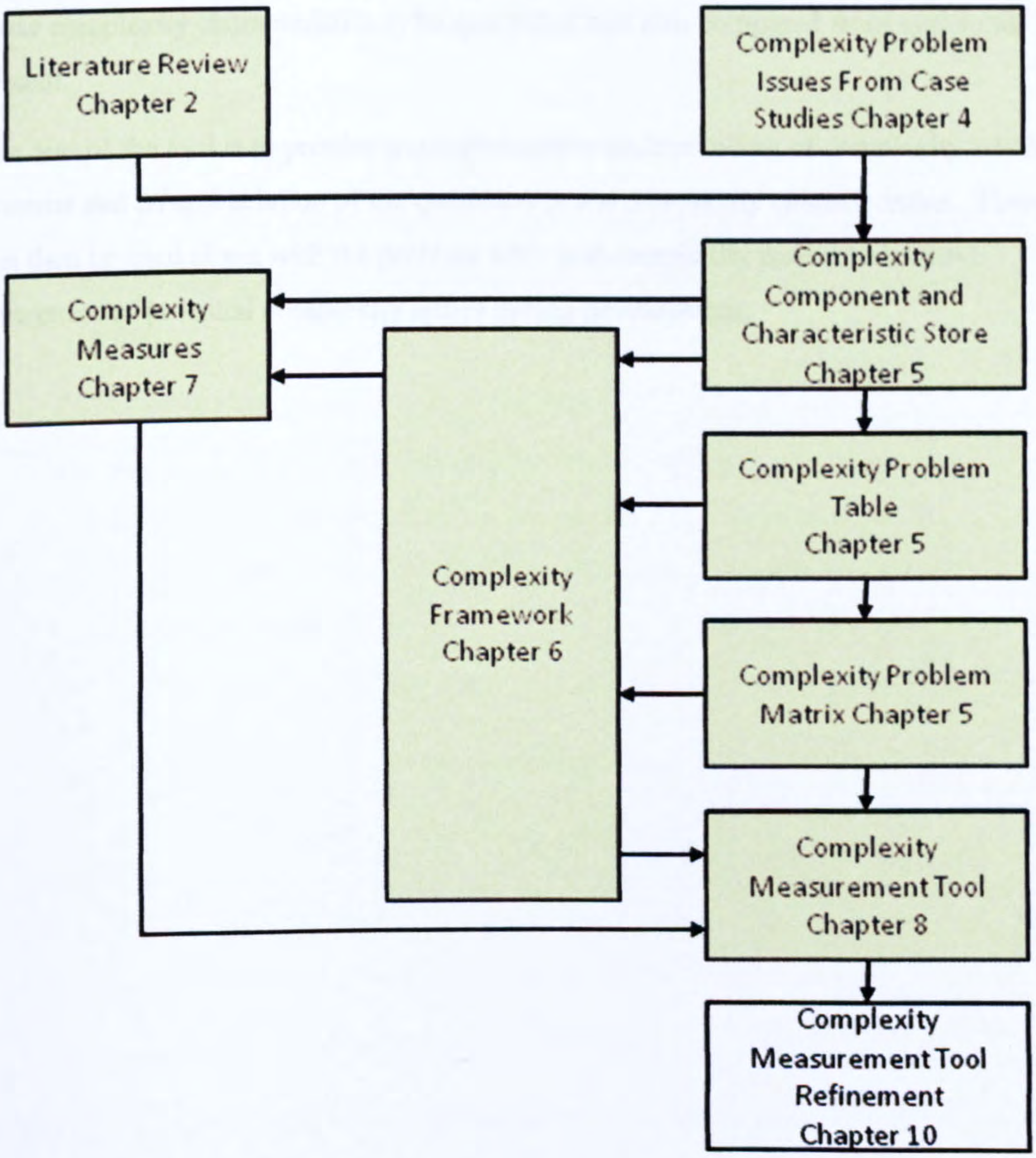


Figure 38 - The layout of the work and the thesis outputs roadmap.

Figure 38 shows the thesis roadmap and how this section fits into the overall structure, the tool uses the concepts within the Complexity Framework of chapter 6, the

knowledge collected from the problem issues using the Problem Matrix within chapter 5 and the measures down selected in chapter 7.

The sources and methodology for capturing the data are outlined and will provide the information and data to populate the analysis tool. Additionally the methodology has been outlined for the evaluation of the tool, and this will enable enhancements to be added to the tool or changes to be made that improve the output.

The output of the tool is comprehensive and creates a data set that enables an understanding of the complexity characteristics of that system. Also the tool enables those complexity characteristics to be quantified and also compared from system to system.

The aim of the tool is to provide a comprehensive understanding of complexity within systems and an appreciation of the quantities of the complexity characteristics. These can then be used along with the problem table and complexity matrix to improve awareness of potential complexity issues during development.

9 Results Analysis

This chapter details the results that were generated from the second set of case studies outlined within section 8.2 using the complexity analysis tool outlined within section 8.3. Once the results are collected and shown they are then compared and conclusions made regarding the tool and data.

9.1 Introduction

The complexity analysis tool detailed within section 8.3 was applied to three separate case studies which are detailed within section 8.2:

- Naval Command System (NCS) – A command system for navy ships, consisting of configurable workstations.
- Aircraft Fuel System Test Rig (Fuel Rig) – A fuel pumping test rig consisting of pumps, sensors, pipe work, tanks and electronic interfaces with a computer controller.
- Naval Command System Trainer (NCST) – A training system consisting of a configurable number of terminals that interact with real navy command systems to improve realism. Since the system is configurable, and workstations can be added or removed, 6 variations are made for a trainer with 1 to 6 workstations:
  - NCST1 – One workstation in operation.
  - NCST2 – Two workstations in operation.
  - NCST3 – Three workstations in operation.
  - NCST4 – Four workstations in operation.
  - NCST5 – Five workstations in operation.
  - NCST6 – Six workstations in operation.

The outputs of the tool, the findings and analysis of these results are detailed within this section. The results indicated that there were deficiencies within the measurement sets, and aspects of these deficiencies were addressed in a revision of the analysis tool. The results analysis is divided into two sections for the three case

studies. The first details the overall system complexity as a whole, and the second details the complexities of the system interfaces and sub-systems.

In practice collecting information regarding suppliers and personnel was difficult. Although the data is available the time taken to collect that data and process it would have been too long for it to have been included in the analysis tool. As a result this data was omitted from the results.

9.2 Overall System Results

The initial results collected from the analysis tool, were those that related to the system as a whole. The overall system is treated in the same way as the sub-systems, subject to those measures outlined within Table 54 within section 8.3.2.

Table 62 shows the results for the complexities of the overall systems for each case study.

Measures	NCS	FUEL RIG	NCST 1	NCST 2	NCST 3	NCST 4	NCST 5	NCST 6
Connectivity Multi	6	37.33	5.333	8	9.6	10.667	11.429	12
Connectivity Single	1.714	6.889	1.333	2	2.4	2.667	2.857	3
Connectivity Single %	57.14	86.11	66.667	66.667	60	53.333	47.619	42.857
Link Number	42	168	16	32	48	64	80	96
Link Complexity	186	346	64	128	192	256	320	384
Number of Link Types	2	4	1	1	1	1	1	1
Element Number	7	9	3	4	5	6	7	8
Element Complexity	175	50	67	94	121	148	175	202
Number Of Element Types	9	9	11	11	11	11	11	11
Skill Number	3	4	3	3	3	3	3	3
Commonality Links	1	1	1	1	1	1	1	1
Commonality Link Types	1	1	1	1	1	1	1	1
Commonality Elements	0.857	0	0	0.5	0.6	0.667	0.714	0.75
Commonality Element Types	1	0.91	0.923	0.972	0.978	0.982	0.984	0.987
Commonality Skills	1	1	1	1	1	1	1	1

Table 62 - Overall system data for all systems analysed.

The analysis of the results within the table has been divided into three sections:

- Connectivity and Link Complexities (see section 9.2.1)
- Element Complexities (see section 9.2.2)
- System Commonality (see section 9.2.3)

Each section will show the relevant results, conduct a cross comparison and analysis of the output.

9.2.1 Connectivity and Link Complexities

This section will analyse the connectivity and link complexity results within the analysis tool output. Table 63 is an extract from Table 62 taking only the connectivity and link complexity information.

Measures	NCS	FUEL RIG	NCST 1	NCST 2	NCST 3	NCST 4	NCST 5	NCST 6
Connectivity Multi	6	37.33	5.333	8	9.6	10.667	11.429	12
Connectivity Single	1.714	6.889	1.333	2	2.4	2.667	2.857	3
Connectivity Single %	57.14	86.11	66.667	66.667	60	53.333	47.619	42.857
Link Number	42	168	16	32	48	64	80	96
Link Complexity	186	346	64	128	192	256	320	384
Number of Link Types	2	4	1	1	1	1	1	1

Table 63 - Connectivity and Link complexities for all systems.

Connectivity is measured in three different ways and there are subtle differences between each measurement method (see Table 54). Each different method provides different information regarding the complexity of the system in terms of connectivity.

Higher levels of connectivity indicate a higher chance of dynamic complexity (Senge 1994) in the system. In systems with just detail complexity (Senge 1994) there cannot be any interfaces that are bi-directional in nature, otherwise there would be potential loops for information energy or matter within the system structure. If this is the case the number of interfaces within the system for hierarchical structures (systems with no loops) can range from:

$$Interfaces = Elements - 1$$

To

$$Interfaces = Elements \times (Elements - 1)$$

This means there is in fact a lower and upper bound to the possibility of a hierarchical system structures. The first equation means a hierarchical structure is only possible, and beyond the second equation a non-hierarchical structure is only possible.

Table 64 (lower bound) and Table 65 (upper bound) translate these bounds into values for Connectivity Single % which can then be applied to the results to determine how likely, or to what extent the systems tested exhibit non-hierarchical structure characteristics.

Elements	Single Max	Number for Hierarchical Structures (elements – 1)	Connectivity Single %
1	0	0	N/A
2	2	1	50.00%
3	6	2	33.33%
4	12	3	25.00%
5	20	4	20.00%
6	30	5	16.67%
7	42	6	14.29%
8	56	7	12.50%
9	72	8	11.11%
10	90	9	10.00%
11	110	10	9.09%
12	132	11	8.33%

Table 64 - Element number relationships to the minimum number of interfaces required for a hierarchical system structure with the corresponding connectivity single % values.

Elements	Single Max	Number for Hierarchical Structures (elements * (elements – 1)	Connectivity Single %
1	0	0	N/A
2	2	1	50.00%
3	6	6	50.00%
4	12	12	50.00%
5	20	20	50.00%
6	30	30	50.00%
7	42	42	50.00%
8	56	56	50.00%
9	72	72	50.00%
10	90	90	50.00%
11	110	110	50.00%
12	132	132	50.00%

Table 65 - Element number relationships to the maximum number of interfaces to support a hierarchical system structure with the corresponding connectivity single % values.

The bounds are quite clear, the upper bound of Connectivity Single % is always 50%, and the lower depends on the number of elements within the system. The more elements within the system the lower the Connectivity Single % value needs to be.

Table 66 shows the results for the systems tested. Clearly all the systems show the potential for non-hierarchical structures. Some are close to the upper bound of hierarchical structures and as a result it is likely parts of the systems have loops. Interestingly no system tested exhibits a purely hierarchical structure.

Systems	Connectivity Single % Lower Bound for Non-Hierarchical Structures	Connectivity Single % System Value	DET / DYN Connectivity Single % Over / Under	Connectivity Single % Over Hierarchical Structure
NCS	14.29%	57.14%	35.71% DET / DYN 7.14% DYN	42.85%
FUEL RIG	11.11%	86.11%	38.89% DET / DYN 28.88% DYN	75%
NCST 1	33.33%	66.667%	16.67% DET / DYN 16.67% DYN	33.33%
NCST 2	25.00%	66.667%	25% DET / DYN 16.67% DYN	41.67%
NCST 3	20.00%	60%	30% DET / DYN 10.00% DYN	40%
NCST 4	16.67%	53.333%	33.33% DET / DYN 3.333% DYN	36.663%
NCST 5	14.29%	47.619%	33.33% DET / DYN	33.329%
NCST 6	12.50%	42.857%	30.36% DET / DYN	30.357%

Table 66 - System element numbers and their respective connectivity single % values for detail complexity structures.

Table 66 provides an indication as to the nature of the system interface structure. The majority of the systems (NCS, Fuel Rig, NCST 1 to 4) tested indicate a definite non-hierarchical nature with definite interface loops. Since NCST 1 to 4 have definite non-hierarchical structures, it is very likely NCST 5 and 6 do also.

The NCST systems with one and two workstations exhibit the highest proportion non-hierarchical (dynamic) interface structure of all the systems tested. NCS and the Fuel Rig have the highest levels of potential non-hierarchical structures.

These numbers by themselves are not sufficient when attempting to understand the complexity characteristics within these systems. Systems may exhibit structures that are very non-hierarchical in nature (in proportion to the system size), but those

structures may be small, or consist of interfaces with lower complexities (link complexity). In order to appreciate how complex these system structures really are, the various connectivity values must be considered along with the number of elements, interfaces, their associated complexities and the degree of commonality within the system. With all this information an accurate picture of the system complexity characteristics can be established.

Measures	NCS	FUEL RIG	NCST 1	NCST 2	NCST 3	NCST 4	NCST 5	NCST 6
Link Number	42	168	16	32	48	64	80	96
Link Complexity	186	346	64	128	192	256	320	384
Link Complexity Average	4.43	2.05	4.00	4.00	4.00	4.00	4.00	4.00
Number of Link Types	2	2	1	1	1	1	1	1

Table 67 - Number of links in the systems and their complexities.

Table 67 shows the number of links, number of link types, the link complexity and the average link complexity within the systems as a whole. This coupled with the level of non-hierarchical interface structure provides a good appreciation of interface complexities within the systems.

It is clear from Table 67 that the Fuel Rig has the lowest average link complexity, but a very high number of links. This is because a vast majority of the links within the Fuel Rig system are in fact matter movements, of the fluid throughout the rig, and even though there are more interfaces within this system than all the others, the other systems consist of links which contain various forms of data.

Variance overall is low in terms of link types, in fact NCST and NCS appear to have 1 or 2 link types. In reality this is certainly not correct for NCST, as there will be power supplies required to transport energy to power the various system components, as well as the electrical interfaces between components. Unfortunately the data required was not available when these systems were analysed.

Although the fuel rig is very non-hierarchical, the intrinsic complexity within the interfaces is lower. This is a factor when considering the overall complexity of the system. The average link complexity of NCS is the highest coupled with a very high of connectivity within which a high proportion demonstrates potential for non-hierarchical structures and loops.

Although the NCST systems become less non-hierarchical as the systems increase in size (workstations are added), the number of links and their overall complexity increases. NCST 6 demonstrates the highest overall link complexity of all the systems.

9.2.2 Element Complexities

This section will analyse the element and element complexity results within the analysis tool output. Table 68 is an extract from Table 62 taking only the element and element complexity.

Measures	NCS	FUEL RIG	NCST 1	NCST 2	NCST 3	NCST 4	NCST 5	NCST 6
Element Number	7	9	3	4	5	6	7	8
Element Complexity	175	50	67	94	121	148	175	202
Average Element Complexity	25	5.55	22.33	23.5	24.2	24.667	25	25.25
Number Of Element Types	9	9	11	11	11	11	11	11

Table 68 – Element complexities for all systems.

Clearly NCST 6, NCST 5 and NCS have the highest levels of element complexity within the systems tested. The average element complexity is high for all systems with the exception of the Fuel Rig. This system consists predominately of basic sensors, pumps and tanks; as a result the overall element complexity is lower than NCS or the NCST systems which consist of electronic processing hardware.

All systems have a large number of element types, with the NCST systems having 11 different types present. This means the systems have a lot of diversity within their element components and this diversity constitutes a higher complexity.

The NCST systems predictably (since they consist of additional workstations which are identical) increase in a linear fashion in terms of element complexity and element number. There is little difference between the complexity of NCST 1 and the complexity of NCST 2 in terms of the average complexity for each element. It is only the overall element complexity which changes. Also predictably, since NCST 1 contains all the components used to create NCST 2 to 6, they add no additional element types.

Like the links the individual complexity metrics for the elements are of little use, they must be used together in order to better understand the system.

Clearly NCST 6 is the most complex it terms of elements, however it is not the system with the most elements, but due to the system size and the nature of the elements within it, and it is the most complex.

9.2.3 System Commonality

This section will analyse the connectivity and link complexity results within the analysis tool output. Table 69 is an extract from Table 62 taking only the connectivity and link complexity information.

Measures	NCS	FUEL RIG	NCST 1	NCST 2	NCST 3	NCST 4	NCST 5	NCST 6
Skill Number	3	5	3	3	3	3	3	3
Commonality Links	1	1	1	1	1	1	1	1
Commonality Link Types	1	1	1	1	1	1	1	1
Commonality Elements	0.857	0	0	0.5	0.6	0.667	0.714	0.75
Commonality Element Types	1	0.91	0.923	0.972	0.978	0.982	0.984	0.987
Commonality Skills	1	1	1	1	1	1	1	1

Table 69 – Commonality and skill complexities for all systems.

Clearly some of the measures calculated here are of very little use. The commonality between links and link types shows 100% commonality throughout the systems. So no interface within the systems tested is bespoke.

The useful metrics taken are the commonality metrics for the element data for each system (common elements and element types). At a system level no commonality is observed between elements within the Fuel Rig of NCST 1. This is because no fuel rig sub-systems are the same, tanks are different and pump configurations are different. Similarly NCST 1 consists of three very different sub-systems, a central interface, a network and a workstation. NCS shows a high level of commonality within the system element types - this is more than likely because the system itself consists of 3 workstations which are identical.

The commonality of elements increases as would be expected as another workstation is added, but not linearly as the network sub-system also increases in size with additional common components.

The link type commonality shows a very high proportion of commonality for all systems, all are about 80%, with the majority about 90%. This is consistent with the makeup of the systems. NCS consists of 2 common components, with additional software components. The identical hardware components will contain the same element types and the software consists of a large number of data handling elements and so exhibits a high level of commonality (in this case 100% for element types). There is a similar case for the Fuel Rig and the NCST systems, each consisting of very similar components that will exhibit the same element type properties.

Skill commonality and skill number tell very little in terms of adding to the complexity understanding of the system. The element information for elements and their types would provide any skill information required and this makes any skill commonality redundant.

The Fuel Rig in this case has the lowest level of commonality, closely followed by NCST 1. The systems with the higher levels of commonality are understandably NCST 6, as it consists of 6 identical workstations and NCS as it consists of 3 pairs of identical workstation components.

9.3 System Sub-Systems and Interfaces

The following section details the results for the sub-systems within each system and the interfaces between each sub-system. These results are taken in turn and in the following order:

- Naval Command System (NCS), see 9.3.1.
- Fuel Rig, see 9.3.2.
- Naval Command System Trainer with 1 to 6 Workstations (NCST 1 to 6), see 9.3.3.

For each of the systems tested, firstly the individual sub-systems are analysed followed by the interfaces between them. The interfaces and their metrics are expanded in the following order:

- Interface Connectivity Single %
- Interface Link Number

- Interface Link Complexity
- Interface Connectivity Multi
- Number of Common Link Types
- Number of Common Links

This is repeated for all the systems tested.

9.3.1 NCS

The following sections detail the results for the Naval Command System (NCS); they are split into two sections. The first contains the sub-system results and the second contains the interface results for the system.

9.3.1.1 NCS Sub-System Complexities

Table 70 shows the overall system view for the NCS system that was tested.

Measures	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
Connectivity Multi	3	2.416	3	2.416	3	2.416	31.1
Connectivity Single	2.842	2.083	2.842	2.083	2.842	2.083	3.7
Connectivity Single %	31.579	37.87	31.579	37.87	31.579	37.87	25.517
Link Number	57	29	57	29	57	29	933
Link Complexity	258	129	258	129	258	129	3732
Number of Link Types	2	2	2	2	2	2	1
Element Number	19	12	19	12	19	12	30
Element Complexity	285	171	287	171	287	171	603
Number Of Element Types	9	9	9	9	9	9	9
Skill Number	2	2	2	2	2	2	2
Commonality Links	0.596	0.828	0.596	0.828	0.596	0.828	1
Commonality Link Types	1	1	1	1	1	1	1
Commonality Elements	0	0	0	0	0	0	1
Commonality Element Types	1	1	1	1	1	1	1
Commonality Skills	1	1	1	1	1	1	1

Table 70 - NCS sub-system complexities.

As for the overall systems, the NCS sub-system complexities have been divided into sections.

- Connectivity and Link Complexities (see section 9.3.1.1.1)
- Element Complexities (see section 9.3.1.1.2)
- System Commonality (see section 9.3.1.1.3)

Each section will show the relevant results, conduct a cross comparison and analysis of the output.

9.3.1.1.1 NCS Connectivity and Link Complexities

Table 71 shows the results for the connectivity and link complexities within the sub-systems of the NCS system.

Measures	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
Connectivity Multi	3	2.416	3	2.416	3	2.416	31.1
Connectivity Single	2.842	2.273	2.842	2.273	2.842	2.273	3.7
Connectivity Single %	31.579	45.455	31.579	45.455	31.579	45.455	25.517
Link Number	57	29	57	29	57	29	933
Link Complexity	258	129	258	129	258	258	3732
Link Complexity Average	4.5	4.4	4.5	4.4	4.5	4.4	4
Number of Link Types	2	2	2	2	2	2	1

Table 71 –Connectivity and link complexities for NCS.

The sub-system output shows very clearly that the software system has the highest complexity in terms of interfaces and connectivity within the NCS system. The Link Complexity value and Link Number is much higher for the software sub-system within NCS. The average link complexity is lowest for the software sub-system, but not significantly lower than the other sub-systems (IPU's and DPU's).

The connectivity of the software system is very high, when compared with to the other sub-systems. With multiple connections the software system has a connectivity of 31 which is over 10 fold that of any other sub-system within the NCS system.

Link types vary very little within each sub-system, and the software system only consists of a single link types The other two sub-systems, the IPU 1 to 3 and DPU 1 to 3 are identical and therefore the results within the sub-systems data is the same. Clearly, without any detail analysis the software sub-system has the likelihood of being the most complex component of the NCS system.

Analysis of the structures for each sub-system is required to further improve the understanding of the complexity. Table 72 shows the nature of the sub-system interface structures.

Sub System	Element Number	Connectivity Single % Lower Bound for Non-Hierarchical Structures	Connectivity Single % System Value	DET / DYN Connectivity Single % Over / Under	Connectivity Single % Over Hierarchical Structure
IPU1	19	5%	31.50%	26.50% DET / DYN	26.50%
DPU1	11	11%	45.40%	33.40% DET / DYN	33.40%
IPU2	19	5%	31.50%	26.50% DET / DYN	26.50%
DPU2	11	11%	45.40%	33.40% DET / DYN	33.40%
IPU3	19	5%	31.50%	26.50% DET / DYN	26.50%
DPU3	11	11%	45.40%	33.40% DET / DYN	33.40%
Software	30	3..4%	25.50%	22.10% DET / DYN	22.10%

Table 72 – NCS system element numbers and their respective connectivity single % values for detail complexity structures.

Proportionally to their size, the IPU and DPU components exhibit the highest potential for non-hierarchical interface structures, where as the software system the least. This is due to the software system having such a large number of components with few links between them.

9.3.1.1.2 NCS Element Complexities

This section details the results for the element complexities for the sub-systems within the NCS system.

Measures	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
Element Number	19	12	19	12	19	12	30
Element Complexity	285	171	287	171	287	171	603
Average Element Complexity	15	14.25	15	14.25	15	14.25	20.1
Number Of Element Types	9	9	9	9	9	9	9

Table 73 - Element complexities for NCS.

Again like the links the software has the highest number of elements with a very large element complexity, and significant element complexity average. The average for the other components, the IPUs and DPUs are very similar, but these sub-systems are in

fact very similar in construction, as a result they contain very similar hardware and the results for average complexity for their elements is therefore very much the same.

The software consists of data processing and handing modules, these are very different to the hardware elements within the DPUs and IPUs and there are a lot more of them. Due to an increase in the number of software processing components with a high intrinsic complexity factor within the analysis tool, the software system has a much larger overall element complexity and element complexity average.

The spread in terms of element types, however, is consistent throughout the sub-systems, although the software contains different element types when compared with the IPUs and DPUS.

It is clear that in terms of element complexity, the software is the most complex sub-system within the NCS.

9.3.1.1.3 NCS System Commonality

The commonality of the NCS sub-systems is shown below within Table 74.

Measures	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
Skill Number	2	2	2	2	2	2	2
Commonality Links	0.596	0.828	0.596	0.828	0.596	0.828	1
Commonality Link Types	1	1	1	1	1	1	1
Common Elements	0	0	0	0	0	0	1
Commonality Element Types	1	1	1	1	1	1	1
Commonality Skills	1	1	1	1	1	1	1

Table 74 – Commonality and skill complexities for NCS.

There is a good proportion of commonality within element types, skills, and link types. There is a very obvious lack of duplication with no common elements within the IPU or DPU. The software however consists of predominantly common elements and common links.

The IPU sub-system differs from the DPU sub-systems in that it contains a much lower level of commonality between its elements.

9.3.1.2 NCS Interface Complexities

Table 75 shows the system interfaces that exist within the NCS system.

	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
IPU1		X					X
DPU1	X						
IPU2				X			X
DPU2			X				
IPU3						X	X
DPU3					X		
Software	X		X		X		

Table 75 - NCS interface table.

The interface structure shown provides enables loops to be identified, and the system structure is such that every interface is a bi-directional interface and therefore there are a number of potential loops, meaning a non-hierarchical structure. This was reflected in Table 66, which clearly identified the NCS system containing a non-hierarchical or dynamic complexity structure.

	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
IPU1		3					4
DPU1	3						
IPU2				3			4
DPU2			3				
IPU3						3	4
DPU3					3		
Software	4		4		4		

Table 76 - NCS interface link number.

The number of links between sub-systems is shown within Table 78. There are relatively few connections between sub-systems, with the software containing the most connections overall. The nature of the connections would suggest that the DPU and IPU units seem to function as pairs and the software is the integrating component of the system, as it links all the DPU units together.

	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
IPU1		15, 5					16, 4
DPU1	15, 5						
IPU2				15, 5			16, 4
DPU2			15, 5				
IPU3						15, 5	16, 4
DPU3					15, 5		
Software	16, 4		16, 4		16, 4		

Table 77 - NCS interface link complexity.

Table 77 shows the Link Complexity for the interfaces between components along with the average Link Complexity (Link Complexity, average Link Complexity). Overall the complexities of the links are comparable, with a link complexity of 15 of IPU, DPU interfaces and an overall complexity of 16 for IPU, Software interfaces. The average complexity of the links is high within all the sub-system links, for the software the links show a complexity of 4 and for the links between the IPU and DPU units a complexity of 5 is shown. These links are data transfers through arrays or values and subsequently show high levels of complexity.

	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
IPU1		1					1
DPU1	1						
IPU2				1			1
DPU2			1				
IPU3						1	1
DPU3					1		
Software	1		1		1		

Table 78 - NCS interface number of link types.

Table 78 shows the variation in the link types within the sub-systems. The links are all of one type for each component. This is because the software and hardware systems are separated, the data connections exist within the software system, and the electrical connections and components that support the software systems are within the IPU and DPU units. As result of there being only one link type per interface, the link type commonality is obviously 100% for all links and therefore is not shown as it provides no benefit to understanding the complexity within the system.

	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
IPU1		0.097					0.082
DPU1	0.097						
IPU2				0.1			0.082
DPU2			0.1				
IPU3						0.1	0.082
DPU3					0.1		
Software	0.082		0.082		0.082		

Table 79 - NCS interface connectivity multi.

Table 79 shows the multiple connectivity between the sub-system components of the NCS. This shows the level of connectivity when all interfaces are included between two sub-systems, and this level of connectivity is around 0.1 for all sub-systems with

little variation. The software sub-system component appears to have the lowest level of connectivity when individual interfaces with other sub-systems are considered, but overall the software connects to the most components within the system as the sub-system integrator.

There was no commonality of the links themselves within the system, and subsequently this table was not included. The links themselves are not common as they carry very different information from different elements within the sub-systems. The commonality of the links in terms of absolute duplication exists between the common sub-system interfaces. All interfaces between the IPU and DPU are repeated, along with Software interfaces with the IPU are also repeated. This is shown at a system level, but the commonality at sub-system level is 0.

9.3.2 Fuel Rig

The following data shows the overall system view for the fuel rig system.

Measures	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
Connectivity Multi	25.8	2.344	0.667	2	1.167	1.909	1.605	1.714	1.769
Connectivity Single	1	1.625	0.667	2	1.167	1.818	1.605	1.714	1.769
Connectivity Single %	25	5.242	16.667	100	46.667	17.316	8.677	8.362	9.312
Link Number	129	75	6	6	7	42	61	72	69
Link Complexity	292	150	11	6	9	44	71	94	87
Number of Link Types	2	2	2	2	2	2	3	3	3
Element Number	5	32	9	3	6	22	38	42	39
Element Complexity	516	42	15	4	8	28	52	55	51
Number Of Element Types	4	4	3	2	3	4	4	4	4
Skill Number	2	2	3	2	3	3	4	4	3
Commonality Links	1	0.96	0.833	0.667	1	1	1	1	1
Commonality Link Types	1	1	0.833	0.667	1	1	1	1	1
Commonality Elements	0.6	0.625	0.222	0	0	0.773	0.789	0.810	0.872
Commonality Element Types	1	0.969	1	0.667	0.833	0.955	1	1	1
Commonality Skills	1	0.970	1	1	0.909	1	0.986	0.988	1

Table 80 - Fuel Rig sub-system complexities.

As for the overall systems, the Fuel Rig sub-system complexities have been divided into sections.

- Connectivity and Link Complexities (see section 9.3.2.1.1)
- Element Complexities (see section 9.3.2.1.2)
- System Commonality (see section 9.3.2.1.3)

Each section will show the relevant results, conduct a cross comparison and analysis of the output.

9.3.2.1.1 Fuel Rig Connectivity and Link Complexities

Table 81 shows the results for the connectivity and link complexities within the sub-systems of the Fuel Rig system.

Measures	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
Connectivity Multi	25.8	4.688	0.667	2	1.167	1.909	1.605	1.714	1.769
Connectivity Single	1	1.625	0.667	2	1.167	1.818	1.605	1.714	1.769
Connectivity Single %	50	10.484	16.667	100	46.667	17.316	8.677	8.362	9.312
Link Number	129	75	6	6	7	42	61	72	69
Link Complexity	292	150	11	6	9	44	71	94	87
Link Complexity Average	2.26	2	1.83	1.00	1.29	1.05	1.16	1.31	1.26
Number of Link Types	2	2	2	2	2	2	3	3	3

Table 81 - Connectivity and link complexities for Fuel Rig.

Since in most cases the connectivity multi and single are in fact the same, it would indicate that the interfaces within those sub-systems usually just consist of a maximum of one bi-directional connection. This would suggest that the sub-system interface structures for these sub-systems are, in fact, predominantly hierarchical in nature, and have few if any loops within information, matter or energy transfers.

The PC Harness sub-system stands out as being one of the most complex systems in terms of interfacing. Although other sub-systems have higher numbers of interfaces, or connectivity, the other data for those sub-system components shows that the interfaces themselves are low in complexity, or there is in fact a very low number of interfaces.

The link complexities and their averages show clearly that the majority of the sub-systems have very low complexity links. The PC harness has the highest level of complexity within the links.

The link diversity is low for the majority of the sub-system components. Most sub-systems consist of just 2 link types, with the Collector, Main and Wing tanks consisting of 3 types. The main reason for this is that the Main, Wing and Collector tanks consist of tanks and pipes like the other components, but also sensors that transmit data back through the PC Harness.

The PC harness, although having fewer link types, consists of Boolean and Value data links, as it connects the various sensors within the Fuel Rig to the PC platform.

Sub System	Element Number	Connectivity Single % Lower Bound for Non-Hierarchical Structures	Connectivity Single % System Value	DET / DYN Connectivity Single % Over / Under	Connectivity Single % Over Hierarchical Structure
PC Rig Harness	5	20.00%	50%	30.0% DET / DYN	30.0%
ADF Power	32	3.13%	10.484%	7.4% DET / DYN	7.4%
Engine Tank	9	11.11%	16.667%	5.6% DET / DYN	5.6%
Leak Sim	3	33.33%	100%	16.67% DET / DYN 50% DYN	66.7%
Flow Recon	6	16.67%	46.667%	30.0% DET / DYN	30.0%
Fuel Trans	22	4.55%	17.316%	12.8% DET / DYN	12.8%
Collector Tank	38	2.63%	8.677%	6.0% DET / DYN	6.0%
Main Tank	42	2.38%	8.362%	6.0% DET / DYN	6.0%
Wing Tank	39	2.56%	9.312%	6.7% DET / DYN	6.7%

Table 82 – Fuel Rig system element numbers and their respective connectivity single % values for detail complexity structures.

The majority of the system consists of components that have an interface structure close to hierarchical in nature. The levels of connectivity within the sub-system components are low, often below 15% with three exceptions, the Link Simulator, Flow Reconfiguration and PC Harness sub-systems.

The Link Simulator system is an exception, as it has very few components (only two) and links, but due to the low number the system is shown to have a completely non-hierarchical structure. In this situation however, the interfaces are small in number, low in complexity and subsequently this will mean the sub-system is not difficult to design.

9.3.2.1.2 Fuel Rig Element Complexities

Table 83 shows the element complexities for the Fuel Rig system.

Measures	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
Element Number	5	32	9	3	6	22	38	42	39
Element Complexity	516	42	15	4	8	28	52	55	51
Average Element Complexity	103.2	1.3	1.67	1.33	1.33	1.27	1.37	1.31	1.31
Number Of Element Types	4	4	3	2	3	4	4	4	4

Table 83 – Element complexities for Fuel Rig

Like the link complexities for the Fuel Rig sub-systems the high levels of complexity are found within the PC Harness, with a element complexity of 516 and a huge average element complexity of 103. These figures would suggest that the components of the PC Harness are very complex when compared to the other components. In reality the PC Harness is made up of input and output cards which take data and turn this into information that can be processed by a PC platform. In comparison none of the other sub-systems have element complexities anywhere near this level, as they are simple pipe, sensor, pump and tank configurations.

Overall the diversity of sub-system components is high, with most sub-systems containing 4 different element types.

9.3.2.1.3 Fuel Rig Commonality

The commonality of the Fuel Rig sub-systems is shown below within Table 84.

Measures	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
Skill Number	2	2	3	2	3	3	4	4	3
Commonality Links	1	0.96	0.833	0.667	1	1	1	1	1
Commonality Link Types	1	1	0.833	0.667	1	1	1	1	1
Commonality Elements	0.6	0.625	0.222	0	0	0.773	0.789	0.810	0.872
Commonality Element Types	1	0.969	1	0.667	0.833	0.955	1	1	1
Commonality Skills	1	0.970	1	1	0.909	1	0.986	0.988	1

Table 84 – Commonality and skill complexities for Fuel Rig

The commonality within the Fuel Rig sub-systems differs quite substantially from sub-system to sub-system. Generally most sub-systems have common components, some with identical elements. In order to better understand the commonality within Fuel Rig, the link commonalities must be considered against the number of link types and the number of links, and likewise for the elements.

Measures	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
Link Number	129	150	6	6	7	42	61	72	69
Number of Link Types	2	2	2	2	2	2	3	3	3
Commonality Links	1	0.96	0.833	0.667	1	1	1	1	1
Commonality Link Types	1	1	0.833	0.667	1	1	1	1	1

Table 85 - Fuel Rig system link commonalities.

Table 85 shows the commonalities associated with the sub-system links. The low levels of commonality are found within sub-system components with few internal links. As a result the lack of commonality is potentially less serious. Those systems with very high levels of commonality if not 100%, are predominately the larger sub-systems which contain a high number of links. Commonality within these large systems makes design easier and if a lack of commonality were observed, the variation may make design more difficult.

Measures	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
Element Number	5	32	9	3	6	22	38	42	39
Number Of Element Types	4	4	3	2	3	4	4	4	4
Commonality Elements	0.6	0.625	0.222	0	0	0.773	0.789	0.810	0.872
Commonality Element Types	1	0.969	1	0.667	0.833	0.955	1	1	1

Table 86 - Fuel Rig system element commonalities.

Table 86 shows the commonality within the sub-system elements. Exact duplication of elements is in some cases high, even with a large number of elements within that sub-system (main tank, wing tank, collector tank). This commonality will make design easier. The same systems showing a high level of exact element duplication also show 100% for element type duplication. This is unsurprising since a vast majority of the elements are identical and therefore share the same common element types.

The sub-systems showing the least common elements, in either type or exact duplication are relatively small in size. It may be that this size is in fact the cause of the lack of commonality. It would not be unusual for a sub-system containing perhaps 2 or 3 components (elements) to exhibit no common components. Many small sub-systems will not have common components.

Overall commonality within the Fuel Rig is high, exact element duplication is at quite high levels for the larger sub-systems (over 60%) and it is these sub-systems that will impact design and development more than the others.

9.3.2.2 Fuel Rig Interface Complexities

Table 87 shows the system interfaces that exist within the Fuel Rig system.

	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
PC Rig Harness		X	X	X	X	X	X	X	X
ADF Power	X		X	X	X	X	X	X	X
Engine Tank	X	X		X	X	X	X	X	X
Leak Sim	X	X	X		X	X			
Flow Recon	X	X	X	X		X	X		
Fuel Trans	X	X	X	X	X		X	X	X
Collector Tank	X	X	X		X	X		X	X
Main Tank	X	X	X			X	X		X
Wing Tank	X	X	X			X	X	X	

Table 87 - Fuel Rig interface table.

The interfaces show the system has a non-hierarchical structure, as there are a large number of potential interface loops. This is reflected within Table 66, within which the Fuel Rig is shown to have a structure with over 50% connectivity, indicating a definite non-hierarchical (dynamic) structure.

Table 88 shows the number of links that exist between the sub-system components. The high levels of linking between sub-systems appear to be between the PC Harness and the other system components. This is understandable, as the PC Harness is the main interface with the PC platform taking data from the Fuel Rig. As a result all the sensor information passes through the PC Harness, and the high numbers of connections consist of the links required for various sensors, and control signals.

	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
PC Rig Harness		0	5	1	0	5	5	12	11
ADF Power	1		2	2	2	2	2	2	2
Engine Tank	12	0		2	0	0	1	0	0
Leak Sim	4	0	0		0	2			
Flow Recon	1	0	1	1		0	1		
Fuel Trans	13	0	1	0	1		1	1	1
Collector Tank	17	0	1		1	0		1	1
Main Tank	20	0	1			1	0		1
Wing Tank	24	0	1			0	1	1	

Table 88 - Fuel Rig interface link number.

Table 89 shows the total link complexity for all the links between the various sub-systems of the Fuel Rig along with their averages (Link Complexity, Link Complexity Average). As expected the PC Harness interfaces contain the highest levels of link complexity within the sub-system interfaces. This will be, the result of the data transmissions through the PC Harness, as these have an intrinsically higher complexity than simple fluid flows within the Rig. The average complexities of the links between sub-systems are also reflective of the higher complexity associated with links between the PC Harness and other system components. The majority of the links between the sub-systems are in fact basic fluid flows, and the link complexity totals, and averages reflect this with a value of 1.

	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
PC Rig Harness		0,0	11, 2.2	3, 3	0,0	12, 2.4	13, 2.6	26, 2.167	24, 2.182
ADF Power	2, 2		4, 2	4, 2	4, 2	4, 2	4, 2	4, 2	4, 2
Engine Tank	26, 2.167	0, 0		2, 1	0, 0	0, 0	1, 1	0, 0	0, 0
Leak Sim	10, 2.5	0, 0	0, 0		0, 0	2, 1			
Flow Recon	2, 2	0, 0	1, 1	1, 1		0, 0	1, 1		
Fuel Trans	29, 2.31	0, 0	1,1	0, 0	1, 1		1, 1	1, 1	1, 1
Collector Tank	41, 2.41	0, 0	1, 1		1, 1	0, 0		1, 1	1, 1
Main Tank	44, 2.2	0, 0	1, 1	0, 0	0, 0	1, 1	0, 0		1, 1
Wing Tank	52, 216	0,0	1, 1			0, 0	1, 1	1, 1	

Table 89 - Fuel Rig interface link complexity.

Table 90 shows the number of different link types, and the variation occurs predominately within the interfaces between sub-systems and the PC Harness. These consist of primarily data transmission of values or Boolean commands. The rest of the sub-systems are mainly matter flows of the fluid around the Fuel Rig, with the exception of the ADF Power, which is a power supply sub-system with energy interfaces between the other sub-systems.

	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
PC Rig Harness		0	2	1	0	2	2	2	2
ADF Power	1		1	1	1	1	1	1	1
Engine Tank	2	0		1	0	0	1	0	0
Leak Sim	2	0	0		0	1			
Flow Recon	1	0	1	1		0	1		
Fuel Trans	2	0	1	0	1		1	1	1
Collector Tank	2	0	1		1	0		1	1
Main Tank	2	0	1			1	0		1
Wing Tank	2	0	1			0	1	1	

Table 90 - Fuel Rig interface number of link types.

Table 91 shows the level of connectivity between the sub-systems. The highest levels of connectivity again are between the PC Harness and the other sub-systems. Again, this is because the PC Harness is the main interface between all the components.

	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
PC Rig Harness		0	2.747	1.786	0	0.712	0.277	0.555	0.581
ADF Power	0.075		0.122	0.168	0.142	0.070	0.041	0.037	0.040
Engine Tank	6.593	0		1.515	0	0	0.046	0	0
Leak Sim	7.143	0	0		0	0.333			
Flow Recon	0.910	0	0.476	1.389		0	0.053		
Fuel Trans	1.852	0	0.108	0	0.132		0.028	0.025	0.027
Collector Tank	0.941	0	0.046		0.053	0		0.016	0.017
Main Tank	0.925	0	0.039			0.025	0		0.015
Wing Tank	1.268	0	0.044			0	0.017	0.015	

Table 91 - Fuel Rig interface connectivity multi.

Table 92 shoes the commonality within the link types, this shows that the all of the links between sub-systems either have 100% commonality or 0%. The 0% commonality is in fact due to there only being a single interface in most cases - as a result commonality is not possible.

	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
PC Rig Harness		0	0	0	0	1	1	1	1
ADF Power	0		1	1	1	1	1	1	1
Engine Tank	1	0		0	0	0	0	0	0
Leak Sim	1	0	0		0	0			
Flow Recon	0	0	0	0		0	0		
Fuel Trans	1	0	0	0	0		0	0	0
Collector Tank	1	0	0		0	0		0	0
Main Tank	1	0	0			0	0		0
Wing Tank	1	0	0			0	0	0	

Table 92 - Fuel Rig interface common link types.

The number of exactly duplicated links was zero, and as a result the table offered no additional information as to the interface complexities and was not included.

The interface data from the sub-systems clearly shows the complexity of the sub-systems is predominantly found within the interfaces between the PC Harness and the other components, however it also shows that despite the high number of interfaces, and connectivity, the interfaces themselves are not of a very high complexity as they are data types Boolean or value in nature.

The other interfaces within the system are very simple in nature as they are fluid flows or simple power distribution (the ADF). The overall complexity of the system interfaces as a result is generally low and manageable when it is compared to the other systems tested.

9.3.3 NCST

The following a number of data sets that have been produced for the NST system with various different configurations. These configurations are for a system with a single workstation to a system with six in single workstation increments.

Many of the components of the overall system are common despite the number of workstations. The CTS Interface remains the same, the workstations remain the same but the Ethernet LAN interfaces function differently. Subsequently the different LAN configurations are shown within the sub-system analysis but the analysis of the Workstations and CTS Interface only occur once.

Table 93 shows the overall system view for the NCST systems (1 workstation configuration to 6 workstation configurations) that were tested. The LAN is the only component that changes depending on the number of workstations and subsequently is the only component that is repeated for each of the potential configurations.

Measures	CTS Interface	WS	LAN (1 WS)	LAN (2 WS)	LAN (3 WS)	LAN (4 WS)	LAN (5 WS)	LAN (6 WS)
Connectivity Multi	4.077	4.167	1.333	1.5	1.6	1.667	1.714	1.75
Connectivity Single	4.077	4.167	1.333	1.5	1.6	1.667	1.714	1.75
Connectivity Single %	16.308	24.51	66.667	50	40	33.33	28.531	25
Link Number	106	75	4	6	8	10	12	14
Link Complexity	421	308	16	24	32	40	48	56
Number of Link Types	2	3	1	1	1	1	1	1
Element Number	26	18	3	4	5	6	7	8
Element Complexity	458	182	42	56	70	84	38	112
Number Of Element Types	10	10	6	6	6	6	6	6
Skill Number	3	3	0	0	1	0	0	1
Commonality Links	1	1	1	1	1	1	1	1
Commonality Link Types	1	1	1	1	1	1	1	1
Commonality Elements	0	0.111	0.667	0.75	0.8	0.667	0.571	0.875
Commonality Element Types	0.989	0.986	1	1	1	1	1	1
Commonality Skills	1	1	1	1	1	1	1	1

Table 93 - NCST 1 to 6 workstation sub-system complexities.

As for the overall systems, the NCST sub-system complexities have been divided into sections.

- Connectivity and Link Complexities (see section 9.3.3.1.1)
- Element Complexities (see section 9.3.3.1.2)
- System Commonality (see section 9.3.3.1.3)

Each section will show the relevant results, conduct a cross comparison and analysis of the output.

9.3.3.1.1 NCST Connectivity and Link Complexities

Table 94 shows the results for the connectivity and link complexities within the sub-systems of the NCST systems.

Measures	CTS Interface	WS	LAN (1 WS)	LAN (2 WS)	LAN (3 WS)	LAN (4 WS)	LAN (5 WS)	LAN (6 WS)
Connectivity Multi	4.077	4.167	1.333	1.5	1.6	1.667	1.714	1.75
Connectivity Single	4.077	4.167	1.333	1.5	1.6	1.667	1.714	1.75
Connectivity Single %	16.308	24.51	66.667	50	40	33.33	28.531	25
Link Number	106	75	8	12	16	20	24	28
Link Complexity	421	308	16	24	32	40	48	56
Link Complexity Average	3.97169811	4.1066667	2	2	2	2	2	2
Number of Link Types	2	3	1	1	1	1	1	1

Table 94 - Connectivity and link complexities for NCST.

The sub-system components with the high levels of connectivity appear to be the Workstation and CTS Interface. With levels of connectivity at over 4, however the connectivity single % is much higher for the LAN sub-system components, because the LAN sub-systems have far fewer components and as a result a much lower maximum number of theoretical connections.

The number of links is significantly higher within the workstations and the CTS Interface than within the LAN sub-systems. This is understandable since the workstations and CTS contain far more components than the LAN sub-systems, and also have a much higher diversity in link types.

Table 95 shows the nature of the interface structure within the systems. The CTS Interface and Workstation values do not change from one system configuration to another, however the LAN changes as more workstations are added.

Sub System	Element Number	Connectivity Single % Lower Bound for Non-Hierarchical Structures	Connectivity Single % System Value	DET / DYN Connectivity Single % Over / Under	Connectivity Single % Over Hierarchical Structure
CTS Interface	26	3.85%	16.31%	16.31% DET / DYN	12.46%
WS	18	5.56%	24.51%	24.51% DET / DYN	18.95%
LAN (1 WS)	3	33.33%	66.67%	16.67% DET / DYN 16.67% DYN	33.33%
LAN (2 WS)	4	25.00%	50.00%	50.00% DET / DYN	25.00%
LAN (3 WS)	5	20.00%	40.00%	40.00% DET / DYN	20.00%
LAN (4 WS)	6	16.67%	33.33%	33.33% DET / DYN	16.66%
LAN (5 WS)	7	14.29%	28.53%	28.53% DET / DYN	14.25%
LAN (6 WS)	8	12.50%	25.00%	25.00% DET / DYN	12.50%

Table 95 – NCST system element numbers and their respective connectivity single % values for detail complexity structures.

Generally the system exhibits characteristics of a hierarchical structure. The LAN for the single workstation configuration shows a high level of non-hierarchical interfacing, indicating a high potential for interface loops. Although this is the case from the results, the main reason this figure is so high for the LAN sub-system with a single workstation is due to the low number of elements within this LAN sub-system. As the LAN system increases in size to cope with the greater number of Workstations, the level of connectivity reduces, the non-hierarchical nature of the sub-system component also reduces.

9.3.3.1.2 NCST Element Complexities

Table 96 shows the element complexities for the NCST systems.

Measures	CTS Interface	WS	LAN (1 WS)	LAN (2 WS)	LAN (3 WS)	LAN (4 WS)	LAN (5 WS)	LAN (6 WS)
Element Number	26	18	3	4	5	6	7	8
Element Complexity	458	182	42	56	70	84	38	112
Average Element Complexity	17.6	10.1	14	14	14	14	14	14
Number Of Element Types	10	10	6	6	6	6	6	6

Table 96 - Element complexities for NCST.

The average element complexity is quite high across all the sub-systems. The system in general consists of a number of data handling units. These units intrinsically are more complex than simple energy or fluid transfers and as a result generate a high average element complexity.

There are much higher numbers of elements within the CTS Interface and Workstations than within the LAN. This is to be expected as the CTS Interface is responsible for transferring information to and from the ship systems. The components of the CTS handle and change this data accordingly.

A much higher level of variety within the element types exist within the Workstations and CTS Interface than within the LAN. This is understandable as the LAN essentially consists of communication cards which are all identical no matter how large the LAN sub-system. These all have the same element characteristics and subsequently the element type number never changes from one LAN to another.

9.3.3.1.3 NCST Commonality

The commonality of the NCST sub-systems is shown below within Table 97.

Measures	CTS Interface	WS	LAN (1 WS)	LAN (2 WS)	LAN (3 WS)	LAN (4 WS)	LAN (5 WS)	LAN (6 WS)
Skill Number	3	3	1	1	1	1	1	1
Commonality Links	1	1	1	1	1	1	1	1
Commonality Link Types	1	1	1	1	1	1	1	1
Commonality Elements	0	0.111	0	0.5	0.6	0.667	0.714	0.875
Commonality Element Types	0.989	0.986	1	1	1	1	1	1
Commonality Skills	1	1	1	1	1	1	1	1

Table 97 – Commonality and skill complexities for NCST.

The commonality within the systems overall is very high. Element types are over 98% common, skills 100% common for all sub-systems, links and link types are also 100% common. The number of duplicated elements however differs substantially from sub-system to sub-system. The LAN common elements increases as the number of workstations increase, this will be due to the increased number of identical interface cards required to accommodate the extra workstations.

9.3.3.2 NCST Interface Complexities

The following sections detail the interface complexities of the system under test for each of the different NCST configurations. In some cases the interface data is the same for all configurations of the system, if this is the case this is stated and the results shown.

9.3.3.2.1 NCST Interface Structures.

The following tables show the interface structures within the NCST systems starting with the single workstation and then incrementing the workstation number up until the six workstation configuration.

	CTS Interface	Ethernet LAN	Workstation 1
CTS Interface			X
Ethernet LAN			X
Workstation 1	X	X	

Table 98 - NCST 1 workstation interface table.

	CTS Interface	Ethernet LAN	WS 1	WS 2
CTS Interface			X	X
Ethernet LAN			X	X
WS 1	X	X		
WS 2	X	X		

Table 99 - NCST 1 to 2 workstation interface table.

	CTS Interface	Ethernet LAN	WS 1	WS 2	WS 3
CTS Interface			X	X	X
Ethernet LAN			X	X	X
WS 1	X	X			
WS 2	X	X			
WS 3	X	X			

Table 100 - NCST 1 to 3 workstation interface table.

	CTS Interface	Ethernet LAN	WS 1	WS 2	WS 3	WS 4
CTS Interface			X	X	X	X
Ethernet LAN			X	X	X	X
WS 1	X	X				
WS 2	X	X				
WS 3	X	X				
WS 4	X	X				

Table 101 - NCST 1 to 4 workstation interface table.

	CTS Interface	Ethernet LAN	WS 1	WS 2	WS 3	WS 4	WS 5
CTS Interface			X	X	X	X	X
Ethernet LAN			X	X	X	X	X
WS 1	X	X					
WS 2	X	X					
WS 3	X	X					
WS 4	X	X					
WS 5	X	X					

Table 102 - NCST 1 to 5 workstation interface table.

	CTS Interface	Ethernet LAN	WS 1	WS 2	WS 3	WS 4	WS 5	WS 6
CTS Interface			X	X	X	X	X	X
Ethernet LAN			X	X	X	X	X	X
WS 1	X	X						
WS 2	X	X						
WS 3	X	X						
WS 4	X	X						
WS 5	X	X						
WS 6	X	X						

Table 103 - NCST 1 to 6 workstation interface table.

The pattern of interfacing between the sub-system components is repeated for all configurations. All workstations interface with the CTS Interface and LAN sub-system components. The LAN interfaces the data transmission between workstations on a single bus, and the CTS interface carries information from the workstations to the vessel.

The structure progressively becomes less and less non-hierarchical in nature as the system increases in size. This is reflected within the overall system results as the NCST systems become less and less non-hierarchical in nature as they increase in size (see section 9.2.1).

Table 104 shows the link numbers between the various sub-systems of the NCST system configurations. Since all the workstation link numbers were the same, only one entry has been included and this is the same for all system configurations. The main links exist between the LAN sub-system and the Workstations. This is due to the way the workstations communicate with each other through a LAN connection. Additionally to these main interfaces information is transmitted through to the CTS system, however the CTS system requires no additional hardware components to deal with 6 workstations than it does for 1 and.

	CTS Interface	Ethernet LAN	Workstation 1 to 6
CTS Interface			1
Ethernet LAN			7
Workstation 1 to 6	1	7	

Table 104 - NCST 1 to 6 workstation interface link number.

Table 105 shows the link complexities of the links outlined. The complexities again were all common throughout all the NCST configurations (1 to 6 workstations) as a result just one entry is included in the table. Since there are many more links between the workstations and the LAN the link complexity for this interface is much higher. However all share the same average complexity of 4, indicating they are all substantial data links (data array links see Table 56)

	CTS Interface	Ethernet LAN	Workstation 1 to 6
CTS Interface			4, 4
Ethernet LAN			28, 4
Workstation 1 to 6	4, 4	28, 4	

Table 105 - NCST 1 to 6 workstation interface link complexity.

Table 106 shows the number of link types, since there is only one link type for every interface, it would appear all interfaces are DATA ARRAY links, which is consistent with the information and nature of the components and the system under test.

	CTS Interface	Ethernet LAN	Workstation 1
CTS Interface			1
Ethernet LAN			1
Workstation 1	1	1	

Table 106 - NCST 1 to 6 workstation interface number of link types.

Table 107 shows the detail of the connectivity between the sub-system components for each configuration of the NCST systems.

	CTS Interface	Ethernet LAN	Workstation 1 to 6
CTS Interface			NCST 1 to 6 - 0.023
Ethernet LAN			NCST 1 - 0.333 NCST 2 - 0.318 NCST 3 - 0.304 NCST 4 - 0.292 NCST 5 - 0.28 NCST 6 - 0.269
Workstation 1 to 6	NCST 1 to 6 - 0.023	NCST 1 - 0.333 NCST 2 - 0.318 NCST 3 - 0.304 NCST 4 - 0.292 NCST 5 - 0.28 NCST 6 - 0.269	

Table 107 - NCST 1 to 6 workstation interface connectivity multi.

The connectivity between the workstations and the CTS interface remains constant for all the system configurations; however the connectivity between the Workstations and the LAN sub-systems reduce as the systems increase in size. This is consistent with the way the sub-systems are linked together.

Table 108 shows commonality between the links within the LAN and Workstation interface in terms of link types. There could be no commonality between the CTS interface and Workstation interface as there is only one link. There were no common links for any of the sub-system interfaces, as a result this table has been omitted.

	CTS Interface	Ethernet LAN	Workstation 1 to 6
CTS Interface			0
Ethernet LAN			1
Workstation 1 to 6	0	1	

Table 108 - NCST 1 to 6 workstation interface common link types.

The interface data shows that there are a low number of links, but the links are fairly common, share the same link type and follow a pattern. The complexity lies within the links themselves and not so much within the intricacy or number of links.

9.3.4 Cross Comparison of Sub-System and Interface Results

This section details a quick cross comparison of the sub-system and interface results collected for the NCS, Fuel Rig and NCST systems. A cross comparison of the

results for the sub-systems and the interfaces will enable the complexity of the systems to be compared.

9.3.4.1 System and Sub-System Complexity Assessment

Overall it is clear that multiple measures are required in order to get an accurate appreciation of the complexity in the systems. It is also clear that complexity at an overall system level is not necessarily a definitive measure of the true extent of the complexity at a sub-system level, a very good example of this is the NCS system. The overall assessment of the NCS system indicates it has a high complexity but is in fact lower in complexity that other systems in terms of connectivity, link complexity, element complexity. In reality the NCS system contains the most complex sub-system analysed, and that is the software component. For the NCS software component complexities can be seen that far exceed even overall system complexities of the Fuel Rig and the NCST configurations. This component is by far the most complex sub-system within this analysis, and subsequently makes the NCS the most complex system within the analysis.

This is a good example of information change through hierarchical scaling; at system level the NCS metrics mask the true extent of the software component complexity, but at a sub-system level the level of information vastly increases to reveal the true nature of NCS.

If this were an investigation into the development of the NCS system, it would be worth conducting an assessment on the Software system alone to determine the complexity distribution within the software sub-system.

The element complexity of the Fuel Rig PC Harness needs addressing. It has a very high element complexity, with an average element complexity of 103 and a total of 516. This is very inflated number; essentially the PC Harness is a conduit for information to pass to a PC platform. The complexity is made up of high numbers of Boolean and value data inputs and outputs. The complexity of the links within the PC Harness are much lower than those within the software despite it having higher element complexities due to the volume of Boolean and value data in and outs. It is this intricacy (Evans 1987) that makes the software of the NCS system more complex.

The complexities of the NCST systems increase as the system configurations get larger. Overall the NCST system element complexity averages and link complexity averages exceed those for the Fuel Rig. This makes NCST the second most complex system.

9.3.4.2 System Interface Complexity Assessment

The system interface complexities vary from system to system; the Fuel Rig has a relatively low interface complexity when compared to the other systems. Most of the Fuel Rig interfaces are fluid transfers between tanks. The interfaces within NCS and the NCST configurations consist mainly of data transfer interfaces, and this increases the average link complexity for those systems.

The high levels complexity within the system interfaces can be found within the NCS system once again, with average link complexities of 4 and 5 indicating data buses existing between the NCS sub-systems. The NCST system too has high levels of link complexity again consisting of interfaces of data arrays.

Although there a number of interfaces between the PC Harness of the Fuel Rig and other sub-system components, those interfaces are of much lower complexity value, and subsequently do not contribute to the level of intricacy of the system as much as the data transfers within the NCSTs and NCS.

Generally all systems have quite high levels of commonality within their interface types. The diversity for all system interfaces is also quite low - this may change if the systems under test had a much larger scope. These systems have been very much workstation orientated apart from the Fuel Rig which is a test bed for fluid flow.

9.3.4.3 Overall Assessment

It appears that although at an overall level the NCS system does not appear to be the most complex system. The software component within the NCS system has a level of complexity far higher than any other system element in NCS, the Fuel Rig or the NCST configurations.

Overall the lowest interface complexity is found within the Fuel Rig, despite the high levels of complexity within the PC Harness, the low levels of complexity within the links and the link complexity reduce the intricacy of the system quite heavily.

9.4 Evaluation of the Measures

The questions that need to be asked are essentially:

A) How easy/difficult is the system to design and integrate?

B) How easy/difficult is the system to develop and manufacture?

C) How easy/difficult is it to predict the system behaviour?

Not all of the measures will answer all of these questions, as discussed before different measures are required to give an overall perspective view of the complexities within the system.

9.4.1 Evaluation of Complexity Measures

Table 46 shows the evaluation and comments regarding the interface, system and sub-system measures.

Calculated Measures	Description	A	B	C	Comments and Evaluation
Connectivity Multi	The number of elements that are connected regardless of the direction of travel of the information, matter or energy. That is if A and B are connected and A passes information to B, a connection exists in the model between A and B, and is counted as a connection for both elements.	X		X	Connectivity has a coupling with both integration and behavioural prediction. The higher the coupling and connectivity between system elements the more likely emergent properties are to exist as modelling becomes more difficult. Integration of systems with higher levels of coupling and unpredictable behaviour is more difficult than systems with low levels of coupling and highly predictive behaviour. As a result the higher the level of connectivity, the more effort and potentially cost is associated with the development and integration activity.
Connectivity Single	The number of connections between elements but taken from a directional view, i.e. there is a connection in which information from A is passed to B, this will be a single connection from A to B and only counted within A.	X		X	
Connectivity Single %	The percentage of the total possible connections between elements.	X		X	
Link Number	The total number of links within the sub-system or system being analysed	X		X	Link numbers, link complexity and link types like the connectivity affects the ease of integration, but only the link number has an effect on the behavioural prediction. The more links the higher the chances of emergent properties within the system. But those links do not necessarily have to be complex links, emergent properties are in a strange was both independent of link complexity in some cases and a direct result of it in others. The link types is a component of the link complexity and is an appreciation of diversity within the system being viewed, this offers a more informed view of the integration activity from a variety of interface type perspective. Although the interface type number is useful, the spread of the types is perhaps more useful, there may be wide variation but if the overall system is for the most part one type that does reduce the integration workload.
Link Complexity	The total number of links within the sub-system or system multiplied by the link complexity factor allocated.	X			
Number of Link Types	The number of different link types within the system or sub-system.	X			
Element Number	The number of elements within the system or sub-system.	X		X	Element number, diversity and complexity are factors in both integration and the likelihood of emergence. The number of elements within the system is key to emergence, with the more elements the higher the
Element	The number of elements within the system or sub-system multiplied by	X			

Calculated Measures	Description	A	B	C	Comments and Evaluation
Complexity	the complexity type for those elements.				chances of emergent properties and the more difficult it is to predict system behaviour. The number of elements, their diversity in type and their complexity is an indication of the integration issues. Although if these are COTS products the integration issues are restricted to interfaces only and the complexity of elements is somewhat unseen.
Number Of Element Types	The number of different element types within a system or sub-system.	X			
Skill Number	The number of different skills within the organisational body developing the system.	X			Skill diversity within systems is a reflection of the diversity of the engineering required to develop the system. This is a reflection of the complexity of that system, this variety and diversity makes the integration of a system more difficult.
Supplier Total	The number of suppliers within the organisational body developing the system.		X		The supplier diversity within the system would be useful when considering manufacturing. The more suppliers the more complex the management activity to support and co-ordinate the manufacturing and spares supply chains. This is not really a measure of complexity as such though, more a manufacture metric.
Personnel Total	The number of personnel in total working on the system within that company.				There is no clear mapping of the personnel properties with complexities in terms of development, manufacture, or behaviour prediction.
Commonality Links	The number of link duplication within the system.	X	X		Mutual links and link types does not necessarily effect the difficulty of behaviour prediction. Systems with only one interface style have been known to exhibit complex emergent behaviour, such as the game of life, or flocks. The repetition of links and link types does make integration easier and manufacture simpler too.
Commonality Link Types	The number of link type duplication within the system.	X	X		
Commonality Elements	The number of element duplication within the system.	X	X		Element repetition may reduce complexity within systems in terms of manufacture and integration, there is however no real concrete connection between element duplication and complexity reduction.
Commonality Element Types	The number of element type duplication within the system.		X		
Commonality Skills	The number of skill duplication within the development organisation for the system.	X			The less diversity within the skill set required to produce the system one would expect the system to be less in terms of its complexity. Although this is not strictly true, as high levels of complexity can exist in a single domain, in particular software.
Commonality Suppliers	The number of mutual suppliers within the development organisation for the system.		X		The more duplication in the supply chain, the supply chain may be less robust, but the complexity involved in manufacture is certainly reduced.
Commonality Personnel	The number of personnel that are shared between sub-systems.				There is no clear mapping of the personnel properties with complexities in terms of development, manufacture, or behaviour prediction.

Table 109 – Evaluation of calculated measures for complexity model sub-systems/systems.

The connectivity measures are all proportional and there is little difference in the complexity characteristics of the system each shows. Although without an understanding of the link complexity, connectivity on its own offers very little in terms of understanding. Looking at the results collected here in the sub-system and system analysis the majority of results for connectivity are around the same area. Larger systems with more links and elements could well have the same connectivity as a small system with fewer links and elements. Therefore there needs to be an additional factor that must influence the picture of the coupling and also give an appreciation for the scale. Highly coupled large systems will be more complex than

small systems with low coupling. Link complexity and connectivity must exist together in order to get this appreciation. Understanding the likelihood of behavioural emergency is only possible with a full understanding of connectivity, the links and the link complexity.

Commonality complexity measures provide support for understanding the manufacture complexities and the other complexity measures. With these measures commonality can be appreciated within the system, and this has an effect on the element and link complexity interpretations. If the commonality is high within a system, then

A better measure here may be to tell distributions of elements that are identical and the distribution of element types as well as links and link types. These distributions may be more appropriate and informative than the mutual information measures shown in the systems here.

9.5 Conclusion

The complexity measures were successfully applied to the systems tested. The data showed clear distinctions between the systems in terms of their complexity make up. It showed that despite having a high values in one type of complexity (e.g. connectivity), the other complexities did not necessarily follow suit, demonstrating that a multi-dimensional approach is necessary to fully understand complexity within systems.

It is clear that some of the measures are not required, several of the connectivity measures do not yield interesting results and the only two that are useful are those measures dealing with connectivity within single links, and the percentage of single links that exist within the system compared to the maximum number possible, and even this requires modification.

There is a gap within the results when considering the skill set included within the system or sub-systems. Like links and element data, a spread is required to fully appreciate the affect that skill distributions or variation of skill requirements within systems or sub-systems may have on the complexity of the system. Relating to the concept of variety and its inherent links to complexity (the more variance within the system the more likely the system is to exhibit complex behaviour or characteristics).

With the lack of personnel data it is unclear how useful a skill spread would be, without knowing the numbers of personnel involved. For this reason it has been omitted.

A further gap within the results, personnel data and records could not easily be found for the projects tested. It may be possible for new projects to obtain better records and account for personnel within the results. Without the addition of the personnel data within the results it is unclear as to how this may affect the complexity of the system.

In terms of content, several of the measures do not hold true to their design and they need to be modified to reflect the design. Some of the measures can be removed from the model as they do not yield valid results, or if they do they are directly proportional to another measure already within the set.

Maturity within the elements and interfaces is an issue that is not addressed within this data. The information used to calculate various complexities does not take into account that although the interface itself may have a high complexity in some areas, the maturity of that interface means that it is very easy to implement.

Overall the output from the tool enables an appreciation of the complexity within the systems, however this tool needed improvement and reduction in the areas outlined within this section.

10 Changes to the Tool

This chapter details the changes that were made to the complexity measurement tool as a result of the first set of results and the analysis within chapter 9. The conclusions formed from those results lead to the introduction of some other metrics, the removal of some metrics. The new results from the new metrics have been included here, and these metrics are; maturity, increased statistics and spreads for interface types and commonality.

10.1 Introduction

This section details the changes that have been made to the complexity analysis tool. These changes help the model produce results that can be better related to complexity within the system and enhance the outputs correlation with effort and cost predictions by introducing new metrics that broaden the complexity understanding of the systems.

10.1.1 Commonality

The commonality measures have been edited to include spreads of commonality within the systems. Systems would have commonality results of 100%, meaning that every element type within the system was duplicated at some point. But without spread information the type duplication is almost always 100%, and provides a meaningless figure to the user. Much better to understand and appreciate the commonality in the form of % spread of the various element and link types, at both a system and sub-system level.

For example, in the current model, a system could have 2 of type one, 50 of type 2, and the result for the commonality would be 100%. Another system could have 2 of type one, 3 of type two and 2 of type three, this would yield the exact same 100% result. Nothing is learned about the system, even when coupled with the element number, information is lost that is contained within the models. This information is required to get a better perspective on commonality.

10.1.2 Connectivity

The connectivity measures have been reduced to just two of the original set, the connectivity single and connectivity single %. Connectivity multi offers no additional understanding of the system.

10.1.3 Maturity

Maturity measures have been added to the tool outputs. Measuring maturity within engineering systems has been done using technology readiness levels (TRL). These levels are a metric measure of maturity and vary from level 1, deemed the lowest maturity level to 9 the highest (Mankins 1995).

Technology Readiness Level (TRL)	Description
9. Actual Technology System qualified through reliability and maintainability demonstration in service.	Application of the technology in its final form and under mission conditions, such as those encountered in operational test and evaluation and reliability trials. Examples include using the system under operational mission conditions.
8. Actual technology system completed and qualified through test and demonstration.	Technology has been proven to work in its final form and under expected conditions. In almost all cases, this TRL represents the end of Demonstration. Examples include test and evaluation of the system in its intended weapon system to determine if it meets design specifications, including those relating to supportability.
7. Technology system prototype demonstration in an operational environment.	Prototype near or at planned operational system. Represents a major step up from TRL 6, requiring the demonstration of an actual system prototype in an operational environment, such as in an aircraft or vehicle. Information to allow supportability assessments is obtained. Examples include testing the prototype in a test bed aircraft.
6. Technology system/subsystem model or prototype demonstration in a relevant environment.	Representative model or prototype system, which is well beyond the representation tested for TRL 5, is tested in a relevant environment. Represents a major step up in a technology's demonstrated readiness. Examples include testing a prototype in a high fidelity laboratory environment or in simulated operational environment.
5. Technology component and/or basic sub-system validation in relevant environment.	Fidelity of sub-system representation increases significantly. The basic technological components are integrated with realistic supporting elements so that the technology can be tested in a simulated environment. Examples include "high fidelity" laboratory integration of components.
4. Technology component and/or basic technology sub-system validation in laboratory environment.	Basic technology components are integrated. This is relatively "low fidelity" compared to the eventual system. Examples include integration of "ad hoc" hardware in a laboratory.
3. Analytical and experimental critical function and/or characteristic proof-of-concept.	Analytical studies and laboratory studies to physically validate analytical predictions of separate elements of the technology are undertaken. Examples include components that are not yet integrated or representative.
2. Technology concept and/or application formulated.	Invention begins. Once basic principles are observed, practical applications can be postulated. The application is speculative and there is no proof or detailed analysis to support the assumptions. Examples are still limited to paper studies.
1. Basic principles observed and reported.	Lowest level of technology readiness. Scientific research begins to be evaluated for military applications. Examples might include paper studies of a technology's basic properties.

Table 110 - MoD TRL definitions.

What is the effect of maturity on system complexity? It could be argued that highly mature technologies do not contribute to increased complexities within systems. As discussed before, maturity will have a profound impact on the complexity of the system, and in particular the induced complexities within a development programme. Induced complexity may well be due to inappropriate selection of immature technologies or the lack of appropriate management of the steady development of the maturity levels as the programme goes ahead.

Maturity within the elements and interfaces is an issue that was not initially addressed within the data. The information used to calculate various complexities did not take

into account that although the interface itself may have a high complexity in some areas, the maturity of that interface means that it is very easy to implement. Ethernet is a good example of this, although there is a high level of connectivity within the Ethernet sub-system of NCST and also some attributes of the NCS system exhibiting high levels of complexity for the Ethernet components, in reality this interface is very simple to incorporate, as it is standard and widely used with great success.

A system of applying maturity levels and factors has been included within the interfaces and component data. The levels have been estimated based on how the development required for the component or the interface between the components to see how complexity of the system is affected.

10.1.4 Summary of the New Measures

Table 111 shows the additional measures created for the sub-system and overall system complexity metrics analysis. These measures have been calculated using the same data available for the calculation of the results within section 9.

Additional Calculated Measures	Description
Element Maturity Level	The maturity level of the elements within the sub-systems or system components. This level is from 1 to 9 and the selection is made in accordance with Table 110 - MoD TRL definitions.
Link Type Spread	Provide detail regarding the composition of the interfaces within a system or sub-system.
Link Complexity Spread	Provide detail regarding the spread of complexity within the interfaces within a system or sub-system.
Element Type Spread	Provide detail regarding the composition of the elements within a system or sub-system.
Element Complexity Spread	Provide detail regarding the spread of complexity within the elements within a system or sub-system.

Table 111 - Measures added to the complexity analysis tool for the system and sub-systems.

Table 112 shows the additional interface measures that have been calculated using the same raw data as the results calculated within section 9.

Additional Calculated Measures	Description
Interface Maturity Level	The maturity level of the interfaces between the sub-systems. This level is from 1 to 9 and the selection is made in accordance with Table 110 - MoD TRL definitions.
Link Type Numbers	Provide detail regarding the composition of the interfaces within a system or sub-system.
Link Complexity Spread	Provide detail regarding the spread of complexity within the interfaces within a system or sub-system.

Table 112 - Measures added to the complexity analysis tool for the system interfaces.

These measures and their results can be found within the summary of results section.

10.1.5 Results from Added New Measures

The following section is a summary of the results generated from the new measures and is laid out similarly to the results section within chapter 9. Section 10.2 includes the new link and element distribution measures along with element maturity.

The Interface Maturity measures for the systems are found within the interface sections of each system in turn (see 10.3). The distribution of the links, elements, the link complexity averages and interface and element maturities are found within each system section within section 10.3.

The results are discussed as they are shown and an evaluation conducted for the new measurement approaches.

10.2 Overall System Results

The following sections detail the results for the complete systems in terms of the link and element distributions of numbers and complexities, along with interface and element maturity levels.

10.2.1 Link Spreads

Table 113 shows the spread of the link types within the systems overall.

	Matter	Energy	Data (Value)	Data (Bus)	Data(Boolean)	Data (Arrays)
NCS				42.86%		57.14%
FUEL RIG	7.72%	5.03%	22.15%		65.1%	
NCST 1						100%
NCST 2						100%
NCST 3						100%
NCST 4						100%
NCST 5						100%
NCST 6						100%

Table 113 - Spread of link types for overall systems.

The spread of the link types within the systems is shown within Table 113. The NCST systems only have a single link type, the DATA ARRAY link type. Whereas the Fuel Rig and NCS have a combination. The Fuel Rig is the most diverse of all the systems, but the spread is dominated by the DATA BOOLEAN interfaces and DATA VALUE interfaces. Additionally there are some Power and Matter links within the system.

	Matter	Energy	Data (Value)	Data (Bus)	Data(Boolean)	Data (Arrays)
NCS				48.39%		51.61%
FUEL RIG	3.6%	4.69%	30.99%		60.72%	
NCST 1						100%
NCST 2						100%
NCST 3						100%
NCST 4						100%
NCST 5						100%
NCST 6						100%

Table 114 - Spread of link complexity for overall systems.

The spread of the complexity within Table 114 over the different types differs only slightly from the spread of the numbers of links. There are some changes, the spread of complexity within the DATA VALUE of the Fuel Rig is significantly higher than its number share and the complexity spread between the DATA BUS and DATA ARRAY shows differences, as most of the complexity is contained within the fewer DATA BUS links.

10.2.2 Element Spreads

Table 115 shows the distribution of element types within the systems.

	NCS	FUEL RIG	NCST 1	NCST 2	NCST 3	NCST 4	NCST 5	NCST 6
MAT DIST		18.00%						
MAT CONV								
MAT TRANS		1.80%						
MAT CONS								
ENGY DIST		2.80%						
ENGY CONV		1.60%						
ENGY GEN								
DATA IN (BOOLEAN)	14.55%	22.40%	14.34%	14.29%	14.29%	14.29%	14.29%	14.29%
DATA IN (VALUE)	14.68%	10.20%	13.21%	12.54%	12.14%	11.87%	11.67%	11.52%
DATA IN (BUS)	8.31%	1.20%	7.92%	7.87%	7.86%	7.85%	7.84%	7.83%
DATA OUT (BOOLEAN)	14.55%	21.00%	15.47%	16.03%	16.43%	16.70%	16.90%	17.05%
DATA OUT (VALUE)	15.19%	16.20%	14.34%	14.29%	14.29%	14.29%	14.29%	14.29%
DATA OUT (BUS)	9.35%	1.20%	9.06%	9.62%	10.00%	10.26%	10.45%	10.60%
DATA PROCESSING (BOOLEAN)	7.92%	1.20%	12.08%	11.37%	10.95%	10.66%	10.45%	10.29%
DATA PROCESSING (VALUE)	7.92%	1.20%	12.08%	11.37%	10.95%	10.66%	10.45%	10.29%
DATA PROCESSING (BUS)	7.53%	1.20%	1.51%	2.04%	2.38%	2.62%	2.79%	2.92%
SWITCH				0.58%	0.71%	0.80%	0.87%	0.92%

Table 115 - Spread of element types for overall systems.

The results show the Fuel Rig has the largest variety. The system contains almost all the possible element types within the analysis tool (apart from MAT CONV, ENGY GEN). The main reasons for this are that the NCST and NCS systems are predominantly software / electronic systems where as the Fuel Rig actually distributes fluids through a series of tanks, valves and pipes.

The NCS has an even spread of element types. This is predominantly because the elements within the NCS system perform very similar processing functions and require the same element types to perform these functions.

	NCS	FUEL RIG	NCST 1	NCST 2	NCST 3	NCST 4	NCST 5	NCST 6
MAT DIST		9.33%						
MAT CONV								
MAT TRANS		6.67%						
MAT CONS								
ENGY DIST		1.33%	1.49%	1.06%	0.83%	0.68%	0.57%	0.50%
ENGY CONV		1.33%						
ENGY GEN								
DATA IN (BOOLEAN)	8.00%	5.33%	8.96%	8.51%	8.26%	8.11%	8.00%	7.92%
DATA IN (VALUE)	8.00%	8.00%	8.96%	8.51%	8.26%	8.11%	8.00%	7.92%
DATA IN (BUS)	12.00%	4.00%	13.43%	12.77%	12.40%	12.16%	12.00%	11.88%
DATA OUT (BOOLEAN)	8.00%	18.67%	8.96%	8.51%	8.26%	8.11%	8.00%	7.92%
DATA OUT (VALUE)	8.00%	26.67%	8.96%	8.51%	8.26%	8.11%	8.00%	7.92%
DATA OUT (BUS)	12.00%	4.00%	13.43%	12.77%	12.40%	12.16%	12.00%	11.88%
DATA PROCESSING (BOOLEAN)	16.00%	5.33%	11.94%	12.77%	13.22%	13.51%	13.71%	13.86%
DATA PROCESSING (VALUE)	12.00%	4.00%	8.96%	9.57%	9.92%	10.14%	10.29%	10.40%
DATA PROCESSING (BUS)	16.00%	5.33%	11.94%	12.77%	13.22%	13.51%	13.71%	13.86%
SWITCH		9.33%						

Table 116 - Spread of element complexity for overall systems.

Table 116 shows the spread of complexity within the system elements. The spread of complexity differs from the spread of element types as some element types are considered by the model to be more intrinsically complex than others. As a result the PROCESSING element types and DATA element types, in particular those BUS types have a much higher proportional share of the element complexity. Despite there the spread of complexity within the elements is varied across all the systems with some lower complexities existing for power distribution and fluid distribution (Fuel Rig) within the systems.

10.2.3 System Maturity

The maturity information is an average maturity of the system elements and links.

This information is shown for the overall systems within Table 117.

	NCS	FUEL RIG	NCST 1	NCST 2	NCST 3	NCST 4	NCST 5	NCST 6
Element Maturity	8.61	8.951	8.575	8.584	8.59	8.593	8.596	8.598
Link Maturity	8.43	8.679	8.875	8.875	8.875	8.875	8.875	8.875

Table 117 - Overall system maturities.

Overall the maturity of the system elements and their links are very high. This is primarily because the systems are actually operating currently and at the end of their development cycles. The technologies within the systems are mostly matured technologies, the elements of reduced maturity are usually those that were under development such as new hardware, or software elements.

10.3 System Sub-Systems and Interfaces

The following sections detail the spreads for system interface and element, types and complexities for the systems tested. These results are taken in turn and in the following order:

- Naval Command System (NCS), see 10.3.1.
- Fuel Rig, see 10.3.2.
- Naval Command System Trainer with 1 to 6 Workstations (NCST 1 to 6), see 10.3.3.

For each of the systems tested, firstly the individual sub-systems are analysed followed by the interfaces between them.

10.3.1 NCS

The following sections detail the NCS results in terms of the spreads of link types and complexities and the element types and complexities.

10.3.1.1 Link Spreads

Table 118 shows the spread of link types within the NCS system. The type numbers are close to evenly spread between the DATA BUS and ARRAY interfaces within the

IPU and DPU system elements. The software consists only of DATA ARRAY transfers, indicating a high level of commonality within the system.

	Matter	Energy	Data (Value)	Data (Bus)	Data(Boolean)	Data (Arrays)
IPU1				52.63%		47.37%
DPU1				44.83%		55.17%
IPU2				52.63%		47.37%
DPU2				44.83%		55.17%
IPU3				52.63%		47.37%
DPU3				44.83%		55.17%
Software						100%

Table 118 - Spread of link types for NCS.

Table 119 shows the spread of link complexity within the interfaces, due to the lower level of intrinsic complexity associated with DATA ARRAY interfaces, the complexities for these links is lower than that of the DATA BUS links despite the higher numbers.

	Matter	Energy	Data (Value)	Data (Bus)	Data(Boolean)	Data (Arrays)
IPU1				58.14%		41.86%
DPU1				50.39%		49.61%
IPU2				58.14%		41.86%
DPU2				50.39%		49.61%
IPU3				58.14%		41.86%
DPU3				50.39%		49.61%
Software						100%

Table 119 - Spread of link complexity for NCS.

The software information provides little additional information, since there is only a single interface type, all the complexity of those interfaces is contained there.

10.3.1.2 Element Spreads

The spread of the element complexities and types for the NCS system are shown within the following tables.

	IPU1	DPU1	IPU 2	DPU 2	IPU 3	DPU 3	Software
MAT DIST							
MAT CONV							
MAT TRANS							
MAT CONS							
ENGY DIST							
ENGY CONV							
ENGY GEN							
DATA IN (BOOLEAN)	15.04%	17.14%	15.04%	16.67%	15.04%	16.67%	11.31%
DATA IN (VALUE)	15.04%	17.14%	15.04%	16.67%	15.04%	16.67%	11.76%
DATA IN (BUS)	7.96%	5.71%	7.96%	6.06%	7.96%	6.06%	11.31%
DATA OUT (BOOLEAN)	15.93%	17.14%	14.16%	16.67%	14.16%	16.67%	11.76%
DATA OUT (VALUE)	15.93%	17.14%	15.93%	16.67%	15.93%	16.67%	12.22%
DATA OUT (BUS)	8.85%	5.71%	10.62%	6.06%	10.62%	6.06%	11.76%
DATA PROCESSING (BOOLEAN)	7.08%	7.14%	7.08%	7.58%	7.08%	7.58%	9.95%
DATA PROCESSING (VALUE)	7.08%	7.14%	7.08%	7.58%	7.08%	7.58%	9.95%
DATA PROCESSING (BUS)	7.08%	5.71%	7.08%	6.06%	7.08%	6.06%	9.95%
SWITCH							

Table 120 - Spread of element types for NCS.

Table 120 shows the spread of element types within the NCS sub-systems. Within all the sub-system components there are a number of different element types present. The spread within the software sub-system is relatively even throughout, however the IPU and DPU sub-systems seem to have areas of density and areas of reduced density in element types. DATA PROCESSING element types are lower in number, along with DATA BUS IN and OUT elements. There is a much higher concentration of BOOLEAN and VALUE data element types within these system components.

	IPU1	DPU1	IPU 2	DPU 2	IPU 3	DPU 3	Software
MAT DIST							
MAT CONV							
MAT TRANS							
MAT CONS							
ENGY DIST							
ENGY CONV							
ENGY GEN							
DATA IN (BOOLEAN)	11.93%	14.04%	11.85%	14.04%	11.85%	14.04%	8.29%
DATA IN (VALUE)	11.93%	14.04%	11.85%	14.04%	11.85%	14.04%	8.62%
DATA IN (BUS)	9.47%	7.02%	9.41%	7.02%	9.41%	7.02%	12.44%
DATA OUT (BOOLEAN)	12.63%	14.04%	11.15%	14.04%	11.15%	14.04%	8.62%
DATA OUT (VALUE)	12.63%	14.04%	12.54%	14.04%	12.54%	14.04%	8.96%
DATA OUT (BUS)	10.53%	7.02%	12.54%	7.02%	12.54%	7.02%	12.94%
DATA PROCESSING (BOOLEAN)	11.23%	11.70%	11.15%	11.70%	11.15%	11.70%	14.59%
DATA PROCESSING (VALUE)	8.42%	8.77%	8.36%	8.77%	8.36%	8.77%	10.95%
DATA PROCESSING (BUS)	11.23%	9.36%	11.15%	9.36%	11.15%	9.36%	14.59%
SWITCH							

Table 121 - Spread of element complexity for NCS.

Table 121 shows the spread of complexity within the NCS sub-systems. Again, as with the overall systems, the distribution of the complexities of the various elements are different to the numbers. When considering the intrinsic complexities of the system elements and their natures, the distribution of complexity within the elements of the sub-systems evens overall, with most intrinsic complexities of the element types ranging between 6% and 15%, and most around 11% or 12%. This shows that despite a difference in the number of elements within each element type, overall the complexity is relatively even.

10.3.1.3 Sub-System and Link Maturity

The following tables show the maturity of the sub-systems and the interfaces between them for the NCS system.

	IPU1	DPU1	IPU 2	DPU 2	IPU 3	DPU 3	Software
Element Maturity	8.68	8.75	8.68	8.75	8.68	8.75	8
Link Maturity	8.56	8.72	8.56	8.72	8.56	8.72	8

Table 122 - NCS sub-system maturity.

Table 122 shows the element and link maturity of the sub-systems within the NCS system. The maturities are very high throughout the system, as this system is

currently in use and has been for some time. The majority of the technology used within the sub-systems were COTS products, and subsequently had high levels of maturity before they were made part of the system. The link maturity is also high as most of the links are clearly defined links that were mature before the system was developed.

	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
IPU1		9					8
DPU1	9						
IPU2				9			8
DPU2			9				
IPU3						9	8
DPU3					9		
Software	8		8		8		

Table 123 - NCS sub-system interface maturity.

Table 123 shows the maturity of the links within the system. The links overall are less mature between the software and the other units, due to the software being a new component and developed purely for the NCS system this will affect the maturity of the links.

10.3.1.4 Interface Link Spreads

The following tables show the spreads of various complexities within the links between the sub-systems. Only two link types exist between the sub-systems within the NCS system, and these are the DATA BUS link and the DATA ARRAY link. As a result these are the only two links that are shown within this section. The link complexities are displayed in an interface table in the following format for each system; Number of Links, % of Total Links, % of Total Link Complexity.

	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
IPU1		3, 7.1%, 8.1%					
DPU1	3, 7.1%, 8.1%						
IPU2				3, 7.1%, 8.1%			
DPU2			3, 7.1%, 8.1%				
IPU3						3, 7.1%, 8.1%	
DPU3					3, 7.1%, 8.1%		
Software							

Table 124 - NCS DATA BUS link number.

Table 124 shows the spread of the complexity within the DATA BUS links between the sub-systems. The spread is even and when considered in conjunction with the DATA ARRAY data within Table 125 the distribution of intrinsic interface complexity is even between the two types, however the DATA ARRAY links have a higher level of complexity.

	IPU1	DPU1	IPU2	DPU2	IPU3	DPU3	Software
IPU1							4, 9.5%, 8.6%
DPU1							
IPU2							4, 9.5%, 8.6%
DPU2							
IPU3							4, 9.5%, 8.6%
DPU3							
Software	4, 9.5%, 8.6%		4, 9.5%, 8.6%		4, 9.5%, 8.6%		

Table 125 - NCS DATA ARRAY link number.

It is clear from both tables that the different interface types handle different parts of the system. The DATA ARRAY interfaces are contained within the software system and its links with the IPU sub-systems, where as the DATA BUS architectures are between the IPU and DPU sub-systems.

10.3.2 Fuel Rig

The following sections detail the Fuel Rig results in terms of the spreads of link types and complexities and the element types and complexities.

10.3.2.1 Link Spreads

The following two tables detail the spread of the link types within the Fuel Rig sub-systems and also the spread of intrinsic complexity of those links. Table 126 shows the spread of the number of different link types within the system.

	Matter	Energy	Data (Value)	Data (Bus)	Data(Boolean)	Data (Arrays)
Rig PC Harness Sub-System			26.36%		73.64%	
ADF Power Sub-System		96.00%			4.00%	
Engine Tank Sub-System	16.67%	83.33%				
Leak Simulator Sub-System	50.00%	50.00%				
Flow Reconfiguration Sub-System	71.43%	28.57%				
Fuel Transfer Sub-System	95.24%	4.76%				
Collector Tank Sub-System	90.16%	3.28%	6.56%			
Main Tank Sub-System	76.39%	16.67%	6.94%			
Wing Tank Sub-System	76.81%	20.29%	2.90%			

Table 126 - Spread of link types for Fuel Rig.

The system consists mainly of MATTER and ENERGY links, these are the power systems and fluid transfer pipes within the sub-systems. Other components relay information in the form of DATA VALUE or DATA BOOLEAN links.

	Matter	Energy	Data (Value)	Data (Bus)	Data(Boolean)	Data (Arrays)
Rig PC Harness Sub-System			34.93%		65.07%	
ADF Power Sub-System		96.00%			4.00%	
Engine Tank Sub-System	9.09%	90.91%				
Leak Simulator Sub-System	33.33%	66.67%				
Flow Reconfiguration Sub-System	55.56%	44.44%				
Fuel Transfer Sub-System	90.91%	9.09%				
Collector Tank Sub-System	77.46%	5.63%	16.90%			
Main Tank Sub-System	58.51%	25.53%	15.96%			
Wing Tank Sub-System	60.92%	32.18%	6.90%			

Table 127 - Spread of link complexity for Fuel Rig.

Table 127 shows the spread of the intrinsic complexities of the links within each sub-system. The DATA VALUE link type despite its low number of links in comparison to MATTER and ENERGY takes quite a significant proportion of the link complexity within the sub-system interface for the Collector, Main and Wing Tank Sub-Systems. Also it is apparent that the ENERGY links are more complex than the MATTER links and the distribution shows this.

10.3.2.2 Element Spreads

The spread of the element complexities and types for the Fuel Rig system are shown within the following tables.

	Rig PC Harness Sub-System	ADF Power Sub-System	Engine Tank Sub-System	Leak Simulator Sub-System	Flow Reconfiguration Sub-System	Fuel Transfer Sub-System	Collector Tank Sub-System	Main Tank Sub-System	Wing Tank Sub-System
MAT DIST			33.33%	66.67%	50.00%	63.64%	50.00%	60.98%	64.10%
MAT CONV									
MAT TRANS					16.67%	9.09%	5.56%	4.88%	5.13%
MAT CONS									
ENGY DIST		43.75%							
ENGY CONV		25.00%							
ENGY GEN									
DATA IN (BOOLEAN)	41.09%								
DATA IN (VALUE)	17.05%	3.13%							
DATA IN (BUS)									
DATA OUT (BOOLEAN)	32.56%		22.22%			4.55%	11.11%	9.76%	10.26%
DATA OUT (VALUE)	9.30%	28.13%	44.44%	33.33%	33.33%	22.73%	33.33%	24.39%	20.51%
DATA OUT (BUS)									
DATA PROCESSING (BOOLEAN)									
DATA PROCESSING (VALUE)									
DATA PROCESSING (BUS)									
SWITCH									

Table 128 - Spread of element types for the Fuel Rig.

The distribution of element types within the sub-systems is shown within Table 128. The distribution shows clearly that there are no PROCESSING elements, and those that handle data are all DATA VALUE/BOOLEAN IN and OUT elements. It is also clear which sub-systems handle power and which form part of the fuel system moving fluid around the rig, with each tank system mostly consisting of MAT DIST and MAT TRANS elements.

	Rig PC Harness Sub-System	ADF Power Sub-System	Engine Tank Sub-System	Leak Simulator Sub-System	Flow Reconfiguration Sub-System	Fuel Transfer Sub-System	Collector Tank Sub-System	Main Tank Sub-System	Wing Tank Sub-System
MAT DIST			20.00%	50.00%	37.50%	50.00%	34.62%	45.45%	49.02%
MAT CONV									
MAT TRANS					12.50%	7.14%	3.85%	3.64%	3.92%
MAT CONS									
ENGY DIST		33.33%							
ENGY CONV		19.05%							
ENGY GEN									
DATA IN (BOOLEAN)	41.09%								
DATA IN (VALUE)	17.05%	4.76%							
DATA IN (BUS)									
DATA OUT (BOOLEAN)	32.56%		26.67%			7.14%	15.38%	14.55%	15.69%
DATA OUT (VALUE)	9.30%	42.86%	53.33%	50.00%	50.00%	35.71%	46.15%	36.36%	31.37%
DATA OUT (BUS)									
DATA PROCESSING (BOOLEAN)									
DATA PROCESSING (VALUE)									
DATA PROCESSING (BUS)									
SWITCH									

Table 129 - Spread of element complexity for the Fuel Rig.

The distribution of the element complexity within Table 129 shows that the high levels of intrinsic complexity within the elements are found within the DATA handling elements, inputting and outputting Boolean or value information.

10.3.2.3 Sub-System and Link Maturity

The following tables show the maturity of the sub-systems and the interfaces between them for the Fuel Rig system. Table 130 shows the maturity of the elements and links within the sub-systems of the Fuel Rig.

	Rig PC Harness Sub-System	ADF Power Sub-System	Engine Tank Sub-System	Leak Simulator Sub-System	Flow Reconfiguration Sub-System	Fuel Transfer Sub-System	Collector Tank Sub-System	Main Tank Sub-System	Wing Tank Sub-System
Link Maturity	9	8.563	9	9	9	9	9	9	9
Element Maturity	9	8.28	9	9	9	9	9	9	9

Table 130 - Fuel Rig sub-system maturity.

Overall the maturity is very high with most of the components and subsequent links within them being COTS products which were mature before the system was developed.

Table 131 shows the maturity within the interfaces between the sub-systems. There is a divide between the maturity of the interfaces between elements of the system that are new (ADF Power and PC Rig Harness) and those that are well known. The links between the various tanks and flow controlling sub-systems are mainly fluid links and these have been deemed thoroughly understood and therefore carry the high levels of maturity shown.

	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
PC Rig Harness			8	8		8	8	8	8
ADF Power	8		8	8	8	8	8	8	8
Engine Tank	8			9			9		
Leak Sim	8					9			
Flow Recon	8		9	9			9		
Fuel Trans	8		9		9		9	9	9
Collector Tank	8		9		9			9	9
Main Tank	8		9			9			9
Wing Tank	8		9				9	9	

Table 131 - Fuel Rig sub-system interface maturity.

The links between the ADF Power and PC Harness although very mature, as the systems consist of COTS products have a lower maturity as they are new in the design; however the nature of the interface is not new.

10.3.2.4 Interface Link Spreads

The following tables show the spreads of various complexities within the links between the sub-systems. Only four link types exist between the sub-systems within the Fuel Rig system; MATTER, ENERGY, DATA VALUE and DATA BOOLEAN. As a result these are the only links that are shown within this section. The link complexities are displayed in an interface table in the following format for each system; Number of Links, % of Total Links, % of Total Link Complexity.

	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
PC Rig Harness									
ADF Power									
Engine Tank				2, 0.67%, 0.3%			1, 0.3%, 0.15%		
Leak Sim						2, 0.67%, 0.3%			
Flow Recon			1, 0.3%, 0.15%	1, 0.3%, 0.15%			1, 0.3%, 0.15%		
Fuel Trans			1, 0.3%, 0.15%		1, 0.3%, 0.15%		1, 0.3%, 0.15%	1, 0.3%, 0.15%	1, 0.3%, 0.15%
Collector Tank			1, 0.3%, 0.15%		1, 0.3%, 0.15%			1, 0.3%, 0.15%	1, 0.3%, 0.15%
Main Tank			1, 0.3%, 0.15%			1, 0.3%, 0.15%			1, 0.3%, 0.15%
Wing Tank			1, 0.3%, 0.15%				1, 0.3%, 0.15%	1, 0.3%, 0.15%	

Table 132 - Fuel Rig MATTER link number.

Table 132 shows the distribution of the MATTER links between the sub-systems, which shows a very even spread throughout the interfaces.

	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
PC Rig Harness									
ADF Power	1, 0.3%, 0.3%		2, 0.67%, 0.63%	2, 0.67%, 0.63%	2, 0.67%, 0.63%	2, 0.67%, 0.63%	2, 0.67%, 0.63%	2, 0.67%, 0.63%	2, 0.67%, 0.63%
Engine Tank									
Leak Sim									
Flow Recon									
Fuel Trans									
Collector Tank									
Main Tank									
Wing Tank									

Table 133 - Fuel Rig ENERGY link number.

Table 133 shows that the ENERGY links are all provided by the ADF Power sub-system of the Fuel Rig. This component supplies the power for all the other components on the Fuel Rig so the configuration is predictable.

	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
PC Rig Harness			1, 0.3%, 0.46%	1, 0.3%, 0.46%		2, 0.67%, 1%	3, 1%, 1.4%	2, 0.67%, 1%	2, 0.67%, 1%
ADF Power									
Engine Tank	2, 0.67%, 1%								
Leak Sim	2, 0.67%, 1%								
Flow Recon									
Fuel Trans	3, 1%, 1.4%								
Collector Tank	7, 2.3%, 3.3%								
Main Tank	4, 1.3%, 1.9%								
Wing Tank	4, 1.3%, 1.9%								

Table 134 - Fuel Rig DATA VALUE link number.

Table 134 shows the DATA VALUE links between the sub-systems, the PC Rig Harness is the primary interface with the PC software and workstation (not included within this study due to data limitations). The Collector Tank appears to be the one sub-system that transmits and receives the most information from the PC Harness with 7 links to the PC Harness, 3 from it, and a total of 4.5% of the overall link complexity is within this interface.

	PC Rig Harness	ADF Power	Engine Tank	Leak Sim	Flow Recon	Fuel Trans	Collector Tank	Main Tank	Wing Tank
PC Rig Harness			4, 1.3%, 1.25%			3, 1%, 0.9%	2, 0.67%, 0.63%	10, 3.4%, 3.2%	9, 3%, 2.8%
ADF Power									
Engine Tank	10, 3.4%, 3.2%								
Leak Sim	2, 0.67%, 0.63%								
Flow Recon	1, 0.3%, 0.3%								
Fuel Trans	10, 3.4%, 3.2%								
Collector Tank	10, 3.4%, 3.2%								
Main Tank	16, 5.4%, 5%								
Wing Tank	20, 6.7%, 6.3%								

Table 135 - Fuel Rig DATA BOOLEAN link number.

Table 135 shows the transmission of DATA BOOLEAN data to the PC Harness. There is a significantly higher proportion of links of this nature within the system than any other type, and as a result the combined intrinsic complexity of these links is higher. The Wing Tank alone has 29 different DATA BOOLEAN links with the PC Harness, and the Main Tank has 26. It would appear the majority of the complexity within the interfaces is within the DATA BOOLEAN links, however, their structure is very hierarchical in nature, perhaps reducing the overall complexity of the links.

10.3.3 NCST

The following sections detail the NCST results in terms of the spreads of link types and complexities and the element types and complexities.

10.3.3.1 Link Spreads

The following two tables detail the spread of the link types within the NCST sub-systems and also the spread of intrinsic complexity of those links. Table 136 shows the spread of the number of different link types within the system.

	Matter	Energy	Data (Value)	Data (Bus)	Data(Boolean)	Data (Arrays)
CTS Interface			2.83%			97.17%
Ethernet LAN (1 to 6 WS)						100.00%
Workstation 1 to 6			8.00%	18.67%		73.33%

Table 136 - Spread of link types for NCST configurations.

The link type distribution within the system shows that the system handles data, and mainly handles data within arrays (DATA ARRAY). The data passing within the Ethernet system consists completely of the DATA ARRAY type.

	Matter	Energy	Data (Value)	Data (Bus)	Data(Boolean)	Data (Arrays)
CTS Interface			2.14%			97.86%
Ethernet LAN (1 to 6 WS)						100.00%
Workstation 1 to 6			5.84%	22.73%		71.43%

Table 137 - Spread of link complexity for NCST configurations.

Table 137 shows the distribution of the intrinsic complexities associated with the links within the sub-systems. There is very little difference between the intrinsic complexity of the links within sub-system components of the NCST sub-systems and the number distribution of those links.

All the NCST sub-systems contained the same distribution of links for the Ethernet sub-system, this is the only sub-system that changes between configurations and

subsequently was only included once within the tables. All the workstations were identical and so one entry within the table was required for all the NCST configurations (1 workstation to 6 workstations).

10.3.3.2 Element Spreads

The number of elements within the Ethernet sub-systems within the NCST configurations depends on the number of workstations within the configuration. The Ethernet LAN component therefore is shown for each workstation configuration; however the Workstations and CTS Interfaces for each configuration remain the same.

	CTS Interface	LAN 1	LAN 2	LAN 3	LAN 4	LAN 5	LAN 6	WS
MAT DIST								
MAT CONV								
MAT TRANS								
MAT CONS								
ENGY DIST	0.56%							
ENGY CONV								
ENGY GEN								
DATA IN (BOOLEAN)	14.04%	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	14.08%
DATA IN (VALUE)	14.04%	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	9.86%
DATA IN (BUS)	7.30%	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	7.04%
DATA OUT (BOOLEAN)	14.04%	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	18.31%
DATA OUT (VALUE)	14.04%	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	14.08%
DATA OUT (BUS)	7.30%	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	11.27%
DATA PROCESSING (BOOLEAN)	14.04%							9.86%
DATA PROCESSING (VALUE)	14.04%							9.86%
DATA PROCESSING (BUS)	0.56%							4.23%
SWITCH								1.41%

Table 138 - Spread of element types for NCST configurations.

Table 138 shows that despite the change in composition of the Ethernet LAN sub-system within the NCST between configurations that the sub-system element type distributions remain the same.

	CTS Interface	LAN 1	LAN 2	LAN 3	LAN 4	LAN 5	LAN 6	WS
MAT DIST								
MAT CONV								
MAT TRANS								
MAT CONS								
ENGY DIST	0.22%							
ENGY CONV								
ENGY GEN								
DATA IN (BOOLEAN)	10.92%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	10.99%
DATA IN (VALUE)	10.92%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	7.69%
DATA IN (BUS)	8.52%	21.43%	21.43%	21.43%	21.43%	21.43%	21.43%	8.24%
DATA OUT (BOOLEAN)	10.92%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%
DATA OUT (VALUE)	10.92%	14.29%	14.29%	14.29%	14.29%	14.29%	14.29%	10.99%
DATA OUT (BUS)	8.52%	21.43%	21.43%	21.43%	21.43%	21.43%	21.43%	13.19%
DATA PROCESSING (BOOLEAN)	21.83%							15.38%
DATA PROCESSING (VALUE)	16.38%							11.54%
DATA PROCESSING (BUS)	0.87%							6.59%
SWITCH								1.10%

Table 139 - Spread of element complexity for NCST configurations.

Table 139 shows the distribution of the intrinsic complexity within the elements for each sub-system. The spread within the Ethernet LAN sub-system in complexity terms is not even, despite the number of element types being equal. Those elements with DATA BUS characteristics exhibit a much higher intrinsic complexity than those with just DATA BOOLEAN or VALUE elements. Those PROCESSING elements also contain a high proportion of the intrinsic complexities.

10.3.3.3 Sub-System and Link Maturity

The following tables show the maturity of the sub-systems and the interfaces between them for the NCST systems. The maturity within the different configurations was consistent for all the configurations, and subsequently the information displayed has been reduced.

Table 140 shows the maturity within the links and elements for each sub-system of the NCST configurations. Most of the NCST components consist of mature COTS technology, and subsequently the overall maturity is high.

	CTS Interface	Ethernet LAN	Workstation 1 to 6
Link Maturity	8.115	9	8.611
Element Maturity	8.019	9	8.611

Table 140 - NCST sub-system maturity.

The Ethernet LAN component of the configurations is a fully matured and tested technology and as a result shows the highest level.

	CTS Interface	Ethernet LAN	WS 1	WS 2	WS 3	WS 4	WS 5	WS 6
CTS Interface			8	8	8	8	8	8
Ethernet LAN			9	9	9	9	9	9
WS 1	8	9						
WS 2	8	9						
WS 3	8	9						
WS 4	8	9						
WS 5	8	9						
WS 6	8	9						

Table 141 - NCST sub-system interface maturity.

Table 141 shows the maturity within the interfaces between the sub-systems. Overall the links with the Ethernet LAN component and Workstation components are at the maximum, as the technology is fully matured and the interface protocol fully matured and understood. The interfaces between the Workstations and CTS Interface, although mainly COTS interfaces, are shown as less mature since these portions of the system were developed and designed rather than purely purchased as COTS products.

10.3.3.4 Interface Link Spreads

The following tables show the spreads of various complexities within the links between the sub-systems. Only one link type exists within this system, and that is the DATA ARRAY link. The link complexities are displayed in an interface table in the following format for each system; Number of Links, % of Total Links, % of Total Link Complexity.

	CTS Interface	Ethernet LAN	WS 1	WS 2	WS 3	WS 4	WS 5	WS 6
CTS Interface			1, 1.04%, 1.04%	1, 1.04%, 1.04%	1, 1.04%, 1.04%	1, 1.04%, 1.04%	1, 1.04%, 1.04%	1, 1.04%, 1.04%
Ethernet LAN			7, 7.3%, 7.3%	7, 7.3%, 7.3%	7, 7.3%, 7.3%	7, 7.3%, 7.3%	7, 7.3%, 7.3%	7, 7.3%, 7.3%
WS 1	1, 1.04%, 1.04%	7, 7.3%, 7.3%						
WS 2	1, 1.04%, 1.04%	7, 7.3%, 7.3%						
WS 3	1, 1.04%, 1.04%	7, 7.3%, 7.3%						
WS 4	1, 1.04%, 1.04%	7, 7.3%, 7.3%						
WS 5	1, 1.04%, 1.04%	7, 7.3%, 7.3%						
WS 6	1, 1.04%, 1.04%	7, 7.3%, 7.3%						

Table 142 - NCST 6 workstation DATA ARRAY table.

Table 142 shows the distributions for the interfaces within the NCST systems. The interface distributions were repeated for each configuration, subsequently only the 6 workstation configuration is shown. The interface structure and distributions show that the Ethernet LAN sub-system of the system has the most complex interfaces and the highest proportion of the intrinsic complexity within the system. This will be because the Ethernet LAN component acts as a central point for information which can then be sent to the CTS Interface by the Workstations. The interface with the CTS Interface is much less complex, perhaps due to most of the processing being carried out within the workstations, and only small packets of basic information being transmitted to the ship systems through it.

10.4 Evaluation of the Measures

The questions that need to be asked are essentially:

A) How easy/difficult is the system to design and integrate?

B) How easy/difficult is the system to develop and manufacture?

C) How easy/difficult is it to predict the system behaviour?

Not all of the measures will answer all of these questions, as discussed before different measures are required to give an overall perspective view of the complexities within the system.

10.5 Evaluation of Complexity Measures

Table 143 shows the evaluation and comments regarding the new system and sub-system measures added to the complexity analysis tool.

Calculated Measures	Description	A	B	C	Comments and Evaluation
Element Maturity Level	The maturity level of the elements within the sub-systems or system components. This level is from 1 to 9 and the selection is made in accordance with Table 110 - MoD TRL definitions.	X	X	X	The maturity within the design enables an appreciation for which parts of the system are new and the level of understanding the developers have of those components. This helps from a design, development and manufacture perspective and also the link between maturity and understanding may help in gauging the accuracy of behaviour predictions.
Interface Maturity Level	Provide detail regarding the composition of the interfaces within a system or sub-system.	X	X	X	
Link Type Spread	Provide detail regarding the spread of complexity within the interfaces within a system or sub-system.	X	X		The spread of the link types and the spread of the complexity help the developer understand where the complexities are within the system interfaces. The spreads show how the intrinsic complexity within the system is distributed. This intrinsic complexity may also help during manufacture phases, as those links that are more complex may be more difficult to manufacture.
Link Complexity Spread	Provide detail regarding the composition of the elements within a system or sub-system.	X	X		
Element Type Spread	Provide detail regarding the spread of complexity within the elements within a system or sub-system.	X	X		The spread of the element types and the spread of the complexity help the developer understand where the complexities are within the system elements. The spreads show how the intrinsic complexity within the system is distributed. This intrinsic complexity may also help during manufacture phases, as those links that are more complex may be more difficult to manufacture.
Element Complexity Spread	The maturity level of the interfaces within the sub-systems or system components. This level is from 1 to 9 and the selection is made in accordance with Table 110 - MoD TRL definitions.	X	X		

Table 143 – Evaluation of added calculated measures for complexity model sub-systems/systems.

The spread of link and element types and complexities within the systems is very useful from a developmental perspective. The tables show clearly which parts of the system and sub-systems contain the higher levels of intrinsic complexity. If these elements or links were added due to ease of implementation over a more pure solution, they also show the complexities that are induced by this. The spreads provide the additional information about the systems that enhance the understanding sufficiently to see how complexity is distributed, and provides a basis from which to focus resources.

The maturity measures may be used to gauge the level of understanding of components and interfaces. If the technologies within those elements or links are new and not understood, this will be shown with low maturity levels. With the spread of maturity shown for all the sub-systems, this enables the developer to see which parts of the system are exposed to higher risk of complexity that is not understood.

10.6 Conclusion

There are a number of measures that were taken forward to the improved analysis tool, changes of the commonality information measures to indicate the spread of types and identical elements. The connectivity measures were reduced to just 2 measures, indicating connectivity of directional links and also the percentage of the total links, thus giving an appreciation of connectivity and coupling. Element numbers, types and complexity enable further understanding of the system, but require understanding of the commonality and coupling. In short, there is in effect a coupling of complexity measures, they interact with each other to give the real picture, and without all the components system complexity can be distorted.

The model changes were only applied to a single system due to time constraints. NCS was chosen because system had high levels of commonality and yet a degree of diversity in its makeup, subsequently the effectiveness of the complexity spread, maturity and commanlity information could be seen.

Issues generated by the revised tool hinge around the level of work required to generate a value from the model, and ensure that value is meaningful. A tool within industry should show a lot from a small amount of information.

The complexity measures contained within the first pass of the complexity measurement tool did not provide a detailed enough picture of the complexity characteristics and their associated quantities within the systems. The added metrics showing the distribution of numbers of link and element types along with their factored intrinsic complexity provides the developer with a more detailed appreciation of the complexity within the system.

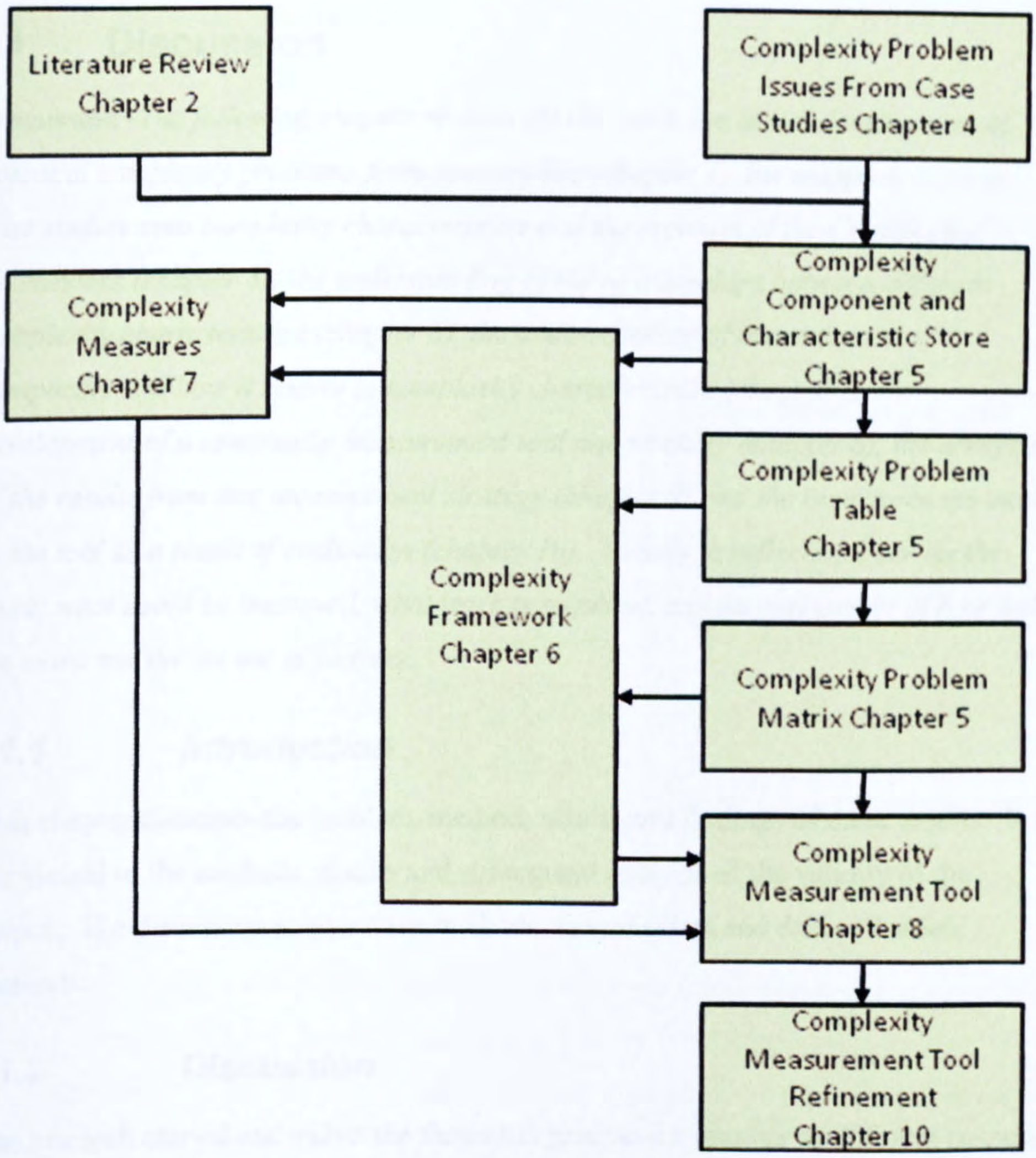


Figure 39 - The layout of the work and the thesis outputs.

Figure 39 shows the roadmap through the thesis, this chapter concludes the refinement of the tool as a result of the initial results and conclusions, and shows the development structure.

11 Discussion

Discussion -*The following chapter reviews all the work, the initial development of potential complexity problems from case studies (chapter 4), the mapping of those case studies onto complexity characteristics and the creation of the Complexity Framework (chapter 5), the understanding of the relationships between different complexity characteristics (chapter 6), the understanding of measurement of complexity and how it relates to complexity characteristics (chapter 7), the development of a complexity measurement tool and strategy (chapter 8), the analysis of the results from that measurement strategy (chapter 9) and the improvements made to the tool as a result of evaluation (chapter 10). Finally, a reflection back on the work, what could be improved, what work is required, and an assessment of how well the work met the set out objectives.*

11.1 Introduction

This chapter discusses the problem, method, results and findings of those results. It is a criticism of the methods, results and subsequent analysis of the validity of the output. The discussion analyses the methods, then the data and data collection methods.

11.2 Discussion

The research carried out within the thesis has produced a number of different outputs.

- **The Complexity Characteristics and Components Store** – A collection of complexity information, characteristics and components that can be related to real world issues or scenarios.
- **The Complexity Framework** – A way of looking at complexity within engineering and the factors that both effect and are affected by it.
- **The mapping of complexity attributes to engineering situations** – A method that allows complexity characteristics found within the framework to be mapped to real engineering problems or situations.

- **The Complexity Measurement Tool** – A method of measuring and quantifying complexity within systems in a way that can be related to the framework.
- **The complexity analysis tool results** – The results generated from the analysis tool for the different systems tested.

The following sections will discuss each of these outputs in turn.

11.2.1 The Complexity Characteristics and Component Store

The CCCS forms the basis for the derivation of a 'complexity understanding' with which industry can better deal with complexity within its engineered products. It is a product of the case studies useful output; it contains a collection of information that can then be related to real problem issues with use of the Complexity Framework.

The CCCS itself is just complexity information, without the Complexity Framework surrounding that information it is simply a literature review or collection of data. With the framework however, it becomes a much more powerful tool, industry can understand the nature of the complexities they are dealing with, and with tools created using the framework as a basis, be able to select appropriate courses of action; choose appropriate coping mechanisms, measures, or descriptions of the complexity within the system. This combined with the live nature of the CCCS, means it can be constantly updated with new information and new appropriate categories to further enhance industrial understanding of complexity in systems. This enhanced understanding is a start in improving the ability of industry to manage the development of large complex systems.

11.2.2 The Complexity Framework

The CAF allows an understanding of complexity in terms of the characteristics of complexity that exist within systems. It is an enhancement of the CCCS knowledge by adding relationships to each component of the CCCS to further enhance understanding (i.e. definitions to classifications etc). The elements from the CCCS and their relationships can then be used to focus efforts on the development of tools, processes, measurement techniques, and coping mechanisms or approaches.

The framework identifies various different themes within the different complexity components of the CCCS, for example hierarchical system structures, non-hierarchical system structures within classifications of systems. The framework identifies relationships between these themes, such as definitions of complexity relating to irreducibility, and measurement techniques that attempt to measure the reducibility of systems, or those definitions that describe intricacy and coupling, which can be related to different classifications of complexity. These interactions between complexity components or characteristics form another very important part of the complexity understanding in systems engineering that this thesis has attempted to develop. It is this understanding that lead to the development of a complexity analysis tool that attempts to identify complexity characteristic components within systems.

It would not be realistic to suggest that the framework on its own addresses most of the issues regarding complexity within engineering systems, but is a beginning. Without this understanding or knowledge of the key complexity components and their interactions, complexity in systems is difficult to deal with. The framework fills a gap within industry when engineering systems. Currently industry struggles with complexity in systems and there is no common understanding or approach to its understanding. The framework provides this link between engineering issues and complexity characteristics found within literature and theory. The framework provides a method of managing that information, and linking it back to real tangible problems forming a basis from which approaches can be developed.

However, the framework is not static; it can always be expanded to incorporate more complexity characteristics or components. This live nature of the framework makes it even more effective in engineering as it is not limited to the 'current' understanding, but can be expanded to include new industrial experience, academic theory, or other information.

The understanding gleamed from using the framework to understand the complexity attributes within the CCCS provides a way of managing the development of various tools, applications or management techniques that can help the engineering process. Within this thesis the framework identifies links between measures and definitions or

concepts, and problems issues and definitions or concepts. These links can then be used to found the basis for the measurement tool which has been created.

Other projects will fit within an element or within a relationship of the framework, and the framework can again provide an understanding to support that work. For example links between problem issues and coping or approach mechanisms. And subsequent to this research there is a new item within the CCCS, and that is the tool developed within this thesis, which essentially is a tool which aids industry in coping with or approaching complexity within design.

11.2.3 The Mapping of Complexity Characteristics to Engineering Issues

The detailed look at various industrial problems and their relationships to complexity further improved the understanding of complexity within engineering. This is an enhancement of the complexity problem issue component within the framework and CCCS. The mapping provides a bridge between components of the Complexity Framework; problem issues, coping mechanisms, definitions, origins and measures of complexity, all based on real industrial issues.

This mapping identified that most of the problems that are found within industry, are in fact induced within the programmes by industry itself, and also a significant proportion of the complexity within developed systems is induced by the development process. Although in some cases this induced complexity is actually beneficial to development, as it introduces mature technologies that can be easily integrated; in some cases it is in fact detrimental and also goes undetected as there is no current method of quantifying or monitoring it.

The mapping of complexity components to known problem issues provides a basis from which analysis can take place. The key complexity components that often are at the centre of industrial problems repeat themselves regularly within the case studies. The framework then provides the additional relationships that these key complexity components have with the other components of complexity. Using these common relationships between complexity characteristics, and methods of quantifying them, problems can be predicted within new development programmes by comparing them against old programmes with known problems.

11.2.4 The Complexity Analysis Tool and Evaluation

Although the data gathering and entering process is a long laborious one, the output of the tool is useful. The complexity analysis tool successfully analyses some of the complexity characteristics within a system and quantifies them. The tool clearly identifies the characteristics that are dominant, it also quantifies those characteristics for comparison with other systems, and it can also locate key cause or problem areas within the system where complexity may be arising from.

The tool can also help to make the distinction between intrinsic complexities and those that are induced, with additional maturity analysis. The measurement of maturity enables the distinction between induced complexity that is beneficial to the system and that which is not. Incorporating additional complexity within the system but improving the maturity level overall is beneficial in most cases in development and the method of analysis can show these trends.

The complexity analysis tool data was difficult to acquire and convert into a useful format for use within the tool. In most cases the data within the tool had to be gathered and interpreted from of specifications, functional diagrams, lists of equipment. The data collected had to be interpreted consistently, or the output would not allow a good comparison between sub-systems or systems as a whole. Data regarding personnel and suppliers was contained elsewhere, and the time taken to retrieve it made it impossible to incorporate into the tool this time.

The systems that were tested were mature systems that had completed their development lifecycle and were in service. This affected the measurement output and made the maturity aspect of the tool redundant. The technologies within the systems were designed and already mature in most cases, but since the systems were built and operating most of the maturity within the system was at a very high level.

Additionally, since the systems that were tested were completed systems that were in service, so the complexities within those systems could be compared with the experiences within the company. For the most part, the complex parts identified by the tool were consistent with those identified by the staff working on the project, although this was not recorded.

With a larger sample set, and measuring systems at the earlier stages of the lifecycle in particular the conceptual design phases, the analysis of the tools effectiveness could

be better ascertained. However despite this the tool does provide useful information and it is clear how it applies to the systems it is analysing.

The changes of the information and the fidelity of the information that is available throughout the lifecycle of a product will affect the measures that are available and also the accuracy of those complexity predictions. Figure 40 shows the relationship between the complexity of the system and the stage within the lifecycle and the level of intrinsic and induced complexity within the design. The arrows on each lifecycle phase provide an indication of the error that could be expected within each stage for the output of the complexity analysis tool. At those stages of the lifecycle where information is limited, the error is large.

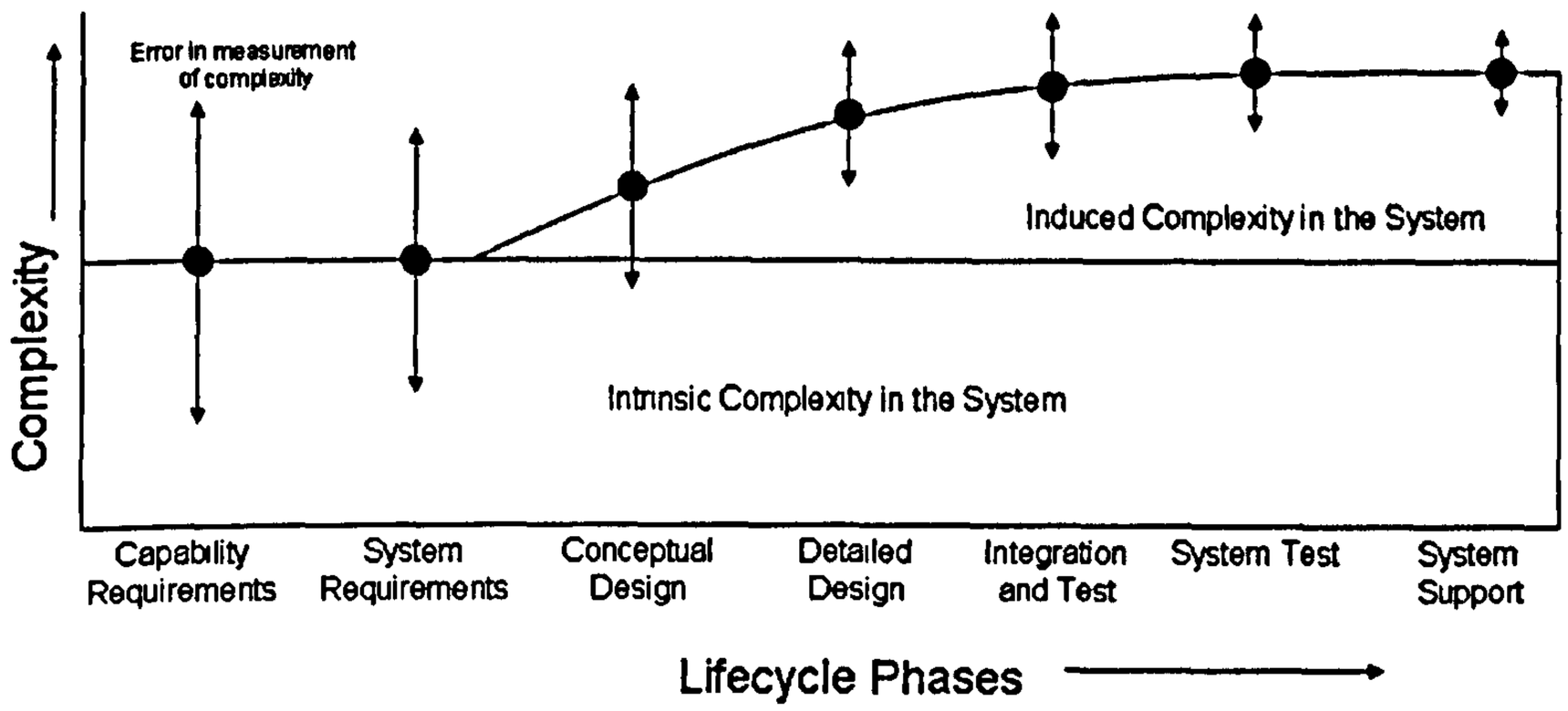


Figure 40 - Maturity and complexity relationships.

The chart consists of two main areas, the first is the intrinsic complexity that exists within the system regardless of the development strategy, and the system has an inherent complexity value for the measurement suite. The induced complexity sits on top of the intrinsic complexity as a result of engineering practices and processes. As development progresses the intrinsic complexity within the system remains the same, but the induced complexity above that begins to be incurred during the conceptual design phases, it is this point that is most crucial in system complexity terms, and it is at this point that the tool is most effective.

The data within the tool is an aspect that has the potential to cause problems in its application. Some data for the tool is readily available at different stages of the lifecycle (shown within Table 144), and subsequently the outputs from the tool can be limited in accuracy.

Point in Development Lifecycle	Comments on Data Availability and the Complexity Analysis Tool Use
Capability Requirements	The data available at this point in the lifecycle that can be used by the complexity analysis tool will be very incomplete; element data would not be known, interfaces not known, system structure unknown. As a result it would be difficult to apply the analysis process at this point within the lifecycle.
System Requirements	Often system requirements are produced with some thought given to the design of the system overall. At this point in the lifecycle information regarding the system structure and the role of elements within that system would be available. The accuracy of the tool output will be limited to the level of information available from the requirements phase.
System \ Sub-System Conceptual Design	The conceptual design phase is a good point at which to apply the complexity measurement approach to a new system under development. The data available becomes much more extensive in terms of interface details, element details, and the maturities of the interface and element technologies. The output the tool provides is also very useful at this stage, as it highlights the potential problems later on in the development phases that complexity could cause.
Sub-System Detail Design	The tool could be applied more easily during the detailed design phases of development. The interfaces and elements along with the system structures would be thoroughly understood at this point, and it would be a case of refining the data within the tool to give more accurate results. The tool during this phase could be used to ensure that any changes made at a conceptual level in the lifecycle to reduce complexity were in fact doing so, and also keeping track of potential additional induced complexity being added to the system unnecessarily.
Sub-System Integration and Test	The tool at this point is only useful as monitoring, and at this late stage within the lifecycle provides little additional help as most of the useful information will have been collected and shown in the conceptual phases.
System Test	
System Operation and Support	Potentially the tool could be used to see how the addition of more functionality in the form of upgrades to a system affects the complexity of that system.

Table 144 - Data availability from the development lifecycle for the complexity analysis tool and its application.

The ideal would be a tool that adapts to the development lifecycle changes, and selects appropriate complexity measures. Incorporation into system development tools would also be appropriate, to negate the problems associated with calculating the measures, making sure they were done consistently, and correctly.

UML based tools have been developed for systems development; SysML, Rhapsody. These tools (SysML, System Architect, and Rhapsody) should be considered for development as they often form part of the system development process. The use of the tool will incorporate conceptual design phases through to the testing and in service phases. The complexity measurement approach could be built into tools like these and calculation made automatic based on the information entered into them.

Another issue to address with regard to the tool output is the nature of the tool input data. The tool input data must be of the same level of abstraction and consistent if systems are to be compared with each other. In this case, the data available for the fuel right system far exceeded that information available for the other systems. This had an influence on the overall results, and perhaps yielded to an over estimate of the

complexity of the fuel rig system in comparison to the other systems. If this tool were to be used regularly within the engineering domain, detailed guides for the data collected must be created so that the level of detail or abstraction is consistent throughout the tool's application.

There are a number of improvements to the tool inputs and outputs that are required, in their current state they are large and unwieldy, and require reduction in order to make them more manageable. For ease of use a better separation of the tool and the data is required so changes to the model can be made quickly and easily.

11.3 The Complexity Analysis Tool Results

Without further analysis of more systems it is not possible to gather any absolutely conclusive evidence that this method of approach will always yield valid results. Using these systems as examples, the evidence does seem to suggest that the NCS system is the most complex as the result of a large software component, which would seem to be accurate based on the scope of the projects.

The tool provides results from both a sub-system and system level. This is expandable, and can be tailored to calculate several levels of decomposition within a system which makes the tool very useful for small and large scale projects. Currently, only 2 levels of decomposition are calculated, a top overall system level, and an underlying sub-system level with their complexities calculated.

The discussion of the results has been split into various sections, and these are as follows:

- Connectivity Results
- Element Complexities
- Interface Complexities
- Commonality
- Maturity

The results will be discussed in this order and this discussion will account for the old and improved tool versions created.

11.3.1 Connectivity Results

The measures within the tool for connectivity within the systems were:

- Connectivity Multi - The number of elements that are connected regardless of the direction of travel of the information, matter or energy. That is if A and B are connected and A passes information to B, a connection exists in the model between A and B, and is counted as a connection for both elements.
- Connectivity Single - The number of connections between elements but taken from a directional view, i.e. there is a connection in which information from A is passed to B, this will be a single connection from A to B and only counted within A.
- Connectivity Single % - The percentage of the total possible connections between elements.

Connectivity Multi did not provide any additional information that was of benefit to understanding the systems complexity characteristics. Subsequently, with any further revisions of the tool there would be no benefit in including it. Connectivity Single, and Single as a percentage cover the concept of intricacy within the systems well enough to provide a reasonable picture, and using the connectivity measures it was possible to deduce the nature of the system structure; hierarchical/non-hierarchical or detail/dynamic.

The connectivity measures successfully showed the level of connectivity within the system and thus an indication of the intricacy within that system. However, the connectivity measures themselves are not enough to determine the level of intricacy in the system on their own. Intricacy is also due to the information being passed and the level of coupling between two components. Although the connectivity is represented, without this other information intricacy cannot truly be estimated or understood.

Additionally, calculations to show the level of hierarchical or non-hierarchical complexity within the system are not totally accurate, but are an estimation of how hierarchical or non-hierarchical a system may or may not be. Despite this, these figures are useful, as they can clearly indicate when a system definitely exhibits non-hierarchical structures.

11.3.2 Element Complexities

The measures within the tool for element complexity within the systems were:

- Element Number - The number of elements within the system or sub-system.
- Element Complexity - The number of elements within the system or sub-system multiplied by the complexity type for those elements.
- Element Complexity Spread - The maturity level of the interfaces within the sub-systems or system components. This level is from 1 to 9 and the selection is made in accordance with Table 110 - MoD TRL definitions.
- Number Of Element Types - The number of different element types within a system or sub-system.
- Element Type Spread - Provide detail regarding the spread of complexity within the elements within a system or sub-system.

The element number information relates well to definitions regarding the size of the system, however if there are not clear common methods for collecting data from systems, these figures are difficult to compare. The level of abstraction associated with this measure means that if systems are to be directly compared they must have their data collected according to the same guidelines. In this case it has been difficult to do this, due to the availability of data on some programmes. Subsequently direct comparisons between systems are tenuous, however despite this it is an obvious measure of system size.

The element complexity within the system provides an overall assessment of the intrinsic complexity of the elements (components) within that system. Although it provides the intrinsic complexity of those components, the result does not necessarily represent the true intrinsic complexity of the system. Some of these elements may actually be contributing to induced complexity, which may be desirable, or it may not. The tool is unable to make this distinction, maturity was introduced in an attempt to make a distinction between new and old technologies within systems, but even this does not provide the complete answer. To improve the output of the tool, induced and intrinsic complexities must be identified, perhaps modelling or representing the system at an architectural and concept level (to indicate the intrinsic complexity of the system) and measuring the complexity of this concept system and then comparing

those complexity values with the complexity values from the actual solution or current design, an appreciation for the induced complexity added by the development process may be possible.

The element complexity spread provides a view of a system which enables focus or emphasis to be placed in areas with high levels of complexity, be it intrinsic or induced. As mentioned earlier, this focus may be misleading as currently it is difficult to separate intrinsic and induced complexities using this measurement approach.

Element types and the spread of those types within the system provide a view of the commonality within the systems and the sub-systems. Variety is a definition of complexity, the higher the diversity within the system the highly the likelihood that system is intrinsically complex. This approach enables not only the system spread to be seen, but that of the sub-systems as well.

The measures here focus on system size, a definition of complexity, element complexity which is linked to concepts of intrinsic complexity, the number of element types and the spread of those elements, indicating a level of commonality or variety within the system along with the spread of complexity to provide a focus on areas with high intrinsic complexity. These characteristics of complexity all relate to the components or characteristics within the CCCS and subsequently items within the framework. It is these relationships that enhance the understanding of the systems under development.

11.3.3 Interface Complexities

The measures within the tool for interface complexity within the systems were:

- Link Number - The total number of links within the sub-system or system being analysed
- Link Complexity - The total number of links within the sub-system or system multiplied by the link complexity factor allocated.
- Link Complexity Spread - Provide detail regarding the composition of the elements within a system or sub-system.
- Number of Link Types - The number of different link types within the system or sub-system.

- Link Type Spread - Provide detail regarding the spread of complexity within the interfaces within a system or sub-system.

As with the elements within the system, the number of interfaces gives an appreciation of the size of the system, and also the level of interaction and coupling (intricacy) within that system or sub-systems.

The complexity and its spread as with the elements enabled an appreciation of intrinsic complexities of the interfaces within the systems and sub-systems. However, like the elements more information is required to clearly distinguish between intrinsic and induced complexities within the system interfaces. The interfaces are heavily subject to the elements within the system, if they are induced, the interfaces between them are also induced by the development process. A clear distinction of intrinsic or induced elements should provide a distinction between interfaces too.

Key information that is missing within the data is the frequency of use of those interfaces, and how they may change the state of the elements which they connect. Interfaces may exist, but may not necessarily contribute to coupling and intricacy, if those interfaces are rarely invoked or if they have little to no effect on the element they connect to or from. If state changes as a result of interfaces could be modelled along with frequencies of use for these interfaces per element, this may provide a better assessment of the coupling between elements.

11.3.4 Organisational

The measures within the tool for organisational complexity within the systems were:

- Skill Number - The number of different skills within the organisational body developing the system.
- Supplier Total - The number of suppliers within the organisational body developing the system.
- Personnel Total - The number of personnel in total working on the system within that company.

Difficulties with data gathering made supplier and personnel information difficult to collect, and subsequently they were not included. However, if a tool of this nature were to be used internally within business collecting this data would be much easier.

The complexity of the skills however is somewhat subjective; there are also political implications when stating one skill is more difficult than another. Subsequently, this information was not considered vastly useful overall.

11.3.5 Commonality

The measures within the tool for commonality complexity within the systems were:

- Commonality Links - The number of link duplication within the system.
- Commonality Link Types - The number of link type duplication within the system.
- Commonality Elements - The number of element duplication within the system.
- Commonality Element Types - The number of element type duplication within the system.
- Commonality Skills - The number of skill duplication within the development organisation for the system.
- Commonality Suppliers - The number of mutual suppliers within the development organisation for the system.
- Commonality Personnel - The number of personnel that are shared between sub-systems.

Commonality or the diversity within the systems and sub-systems is a factor in the complexity of the system. High levels of commonality between components and interface means less variety. The reduction in variety of different interface or element types reduces the skill base required and possibly the level of effort required to develop or manufacture the system. Although collecting information regarding the components and interfaces was simple, collecting information regarding the commonality of suppliers and personnel was very difficult and subsequently this information was not included within the results.

The measure output was not very useful, due to the method by which commonality was calculated the output figures we always very high. The spread information for the different interface and element types shows the level commonality between types

within the system and sub-systems, and subsequently the results are redundant for the commonality of types.

The more useful measurement is the commonality between actual components or interfaces in terms of duplication. This reduces the difficulty in design and manufacture, but it is questionable as to whether or not high commonality or duplication actually reduces the system complexity, emergent properties can be seen within systems of high commonality such as the game of life (Kauffman 1993). Commonality as a measure enables an appreciation of the diversity within the system under test, but not necessarily with a direct relationship to system or sub-system complexity.

11.3.6 Maturity

The measures within the tool for maturity complexity within the systems were:

- Element Maturity Level - The maturity level of the elements within the sub-systems or system components. This level is from 1 to 9 and the selection is made in accordance with Table 110 - MoD TRL definitions.
- Interface Maturity Level - Provide detail regarding the composition of the interfaces within a system or sub-system.

In order to improve the output of the models, maturity of elements and interfaces was considered. Is the interface or element new? Has the interface or element been used before? Is the element or interface standard? Incorporation of some maturity (perhaps factorisation) of the technology was important when differentiating between systems with high levels of COTS/MOTS and those with low levels. It could be argued that systems that rely on mainly COTS/MOTS products lose a lot of the system complexity as a result of this re-use and high maturity. The NCST is a prime example of this; in effect NCST has very little in the way of hardware development. The majority of the new design (and perhaps the complexity within the programme) is contained within the software, which due to the method of measurement here is almost indistinguishable from the hardware system. De-rating hardware complexities as a factor of maturity and on a COTS/MOTS basis would most likely provide a more meaningful result.

NCS goes some way to addressing the issue of hardware and software separation within the results, as the hardware and software have been kept separate within the design documentation since the software is clearly the bulk of the project. Software resides outside of the hardware that it runs on, and therefore factoring the hardware by means of maturity would be easier if the two were separate. This enables the distinction of complexity within hardware and that within software.

This of course begs the question, what about software re-use and maturity? In the development of operating systems many developers re-use certain aspects of their software, and this too should be factored into the tool output.

If measures of maturity are to be used from a concept phase, then this becomes even more important. Often the COTS or MOTS products that will be used within a project are not known, or at least not confirmed within the early stages. The architecture however is, so a list of elements is available with an understanding of their specific roles and attributes, along with the descriptions and numbers of interfaces between then. How is maturity factored in here? Since the hardware and software are not chosen but the architecture is known, measures that analyse the coupling and cohesion of the architecture can be used, but the information required in terms of maturity in order to measure its effect is not known and is omitted.

In all the systems analysed the hardware has been developed along with the software implementation that runs on it. The software is, in most cases, the heart and bulk of the development, and this must be reflected within the measurement scheme. However maturity is only understood after the pure concept starts to employ potential platform realisations, without these realisations these maturity decisions cannot be made.

In this case the measurement strategy needs to be split up to cope with the different stages of design, from the concept development, design and integration to the manufacture and design realisation. Initial measures concentrate on those measures that can be applied to the concept architecture of the system, and gradually as hardware and software realisation decisions are made the shift moves towards maturity and commonality complexity measures can be made, thus giving a balanced approach.

It is interesting that the TRL model also demonstrates that despite components having high levels of maturity, the overall system may still be immature and have a lower TRL than its components. This is paralleled with the nature of complexity within systems, and the emergent properties within systems, a system although comprised of non-complex simple components can exhibit unexpected behaviour, and maturity can be viewed in the same way.

With this close parallel between the nature of complexity and maturity and the influence maturity has on complexity within systems, maturity should be included in the measurement set as a factor affecting the complexity of components and links. The more maturity, or the higher the TRL within that technology, the lower the complexity of its implementation into the design.

In the case of this system, it is difficult to consider maturity along with the rest of the system at this point, as the maturity of all the components of the system will already be sufficient for production. If maturity TRLs were to be assigned to the system components and interfaces which were consistent with TRLs at the beginning of a programme, then the component structure would not be present, as this has evolved during the development lifecycle and has changed since that point.

From these results it is difficult to gauge the usefulness of the maturity measures with these system examples. The systems tested by the tool were mainly COTS, MOTS products within which technologies were deployed which were mature before the project began. This meant the maturities were all very high, usually above level 8 and some aspects were level 9. However, theoretically the maturity measures are very beneficial, as they provide a basis for change within that technology and also an appreciation of the level of understanding, fitting in neatly with definitions of complexity regarding modelling and understanding system elements (see section 2.3).

11.3.7 How can the Tool be used?

The tool would most likely be predominantly used as a method for tracking and managing complexity within systems as they progress through the product lifecycle. The process begins with the gathering of requirements, then creation of the conceptual design, and progresses through to the detailed design, integration and testing.

The conceptual design or concept of the system being developed is devoid of hardware and software specifics, but contains elements, how those elements communicate, what information is passed between them and the function of that element. This conceptual design or concept system design can be analysed by the tool and complexity metrics be produced. It could be argued that the complexity of the conceptual design of the system is the intrinsic complexity of that system, the complexity attributable only to the function and interfaces of the system and not connected to its implementation.

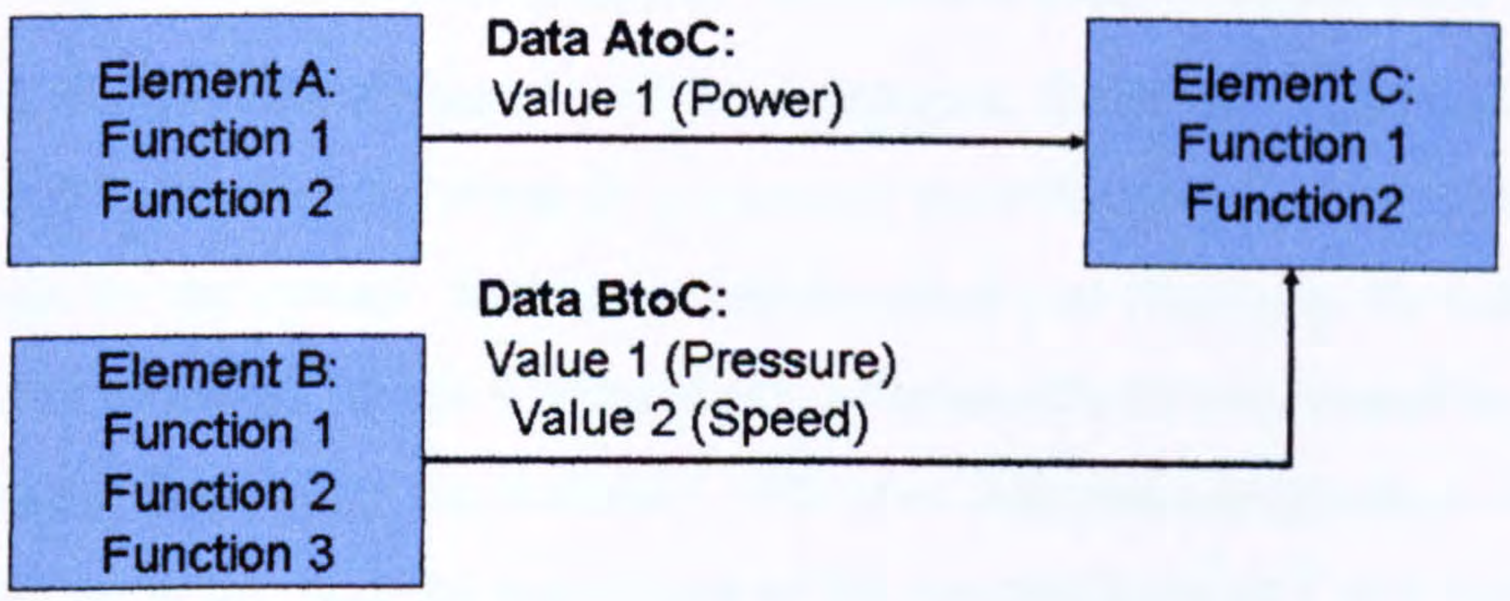


Figure 41 – Conceptual design for a simple system.

Figure 41 shows a conceptual design of a system, the system has 3 elements, those elements have functions that are a result of the requirements for that system, and these elements have information that must pass between them. This conceptual design does not include the hardware or software solution for this system, but just the conceptual design, the conceptual system and the intrinsic complexities can be derived from this representation.

Once the system solution has been implemented (within hardware or software), or part of that system has been created, the complexities of for those components are changed. Figure 42 shows the same system within Figure 41 with a solution developed for element C.

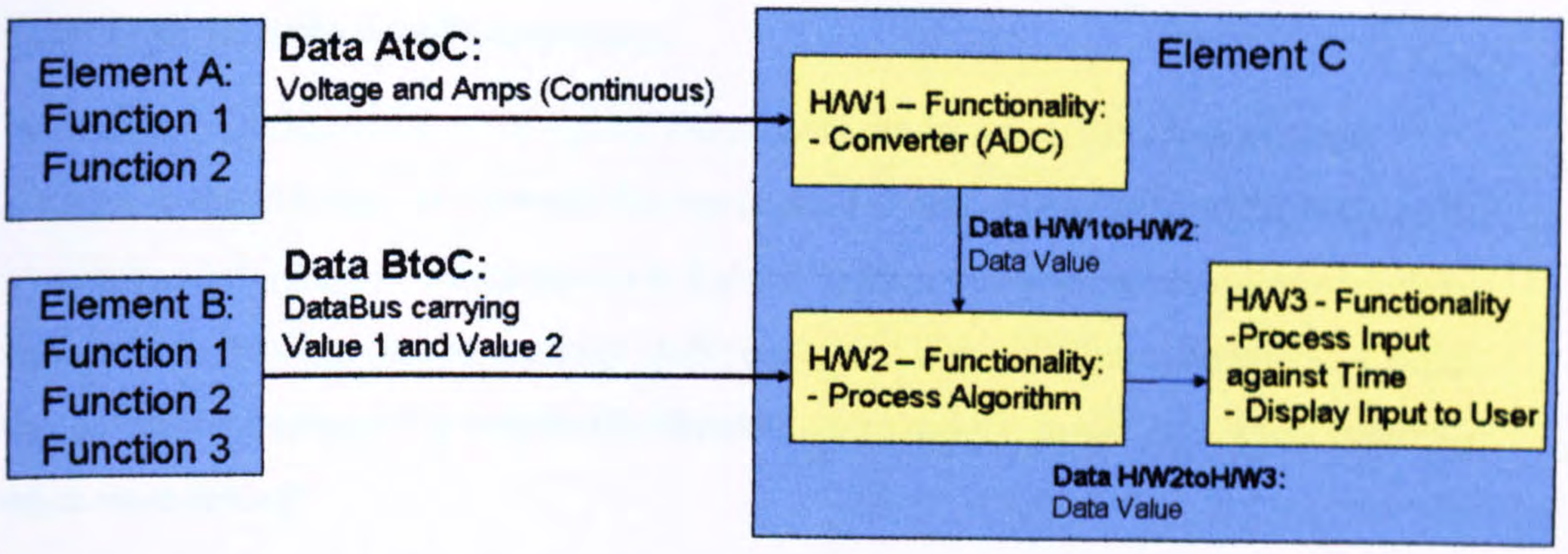


Figure 42 – Conceptual design and actual system design solutions mixed.

The complexities within element C will have changed. Originally there were 2 functions that had to be carried out by element C; these functions are the result of the requirements for the system. In order to achieve these two functions, the actual design of that element C needs 3 components with specific functions and interfaces between them. This design for element C will have different complexities to the concept design, in this case the complexity of the concept element C will be lower than that for the actual design for element C, the difference between these two complexities is the induced complexity resulting from the system design.

The complexity of the concept design (assuming it does not change once design begins) will always be the same or lower than the actual solution implementation as the conceptual design is a pure representation of the systems decomposition, interfaces and functionality, at best the implementation will match this.

Understanding the complexities within the system elements and interfaces, and the differences between the complexity of the conceptual solution and the designed solution can enable management to track induced complexities, intrinsic complexities and also monitor these through the product development lifecycle:

How intrinsically complex is the system?

Determine the complexities within the conceptual design, determine those areas within the conceptual design (interfaces, elements) that have high complexity levels, are these complexity levels acceptable, or does the concept design need further analysis to reduce them?

How high is the induced complexity?

As elements and interfaces are turned from concepts to actual implementations, determine the differences between the conceptual design complexities (intrinsic to the system) and the implementations made (actual system complexity) to determine the induced complexity. Is there a large difference, and if so what has caused it? Could the design be improved to reduce the induced complexity? Is the induced complexity what is expected?

Manage the complexity of the system

Manage the complexity of the system by tracking the intrinsic and induced system element and interface complexities as the system and system solutions are developed.

11.4 Conclusion

It is clear that the measurement tool can show complexity within systems and that the complexities shown can be related to elements of the framework.

Definition Types	Cause Types	Concepts and Classification Types	Measurement Types	Coping Mechanism / Approach Types	Problem Types
1. Irreducible 2. Difficulty in modelling or predictability 3. Difficult to describe, lack of understanding 4. Scale of interfaces and sub-systems 5. The level of interactions and intricacies	1. Design Optimisation 2. Immature Requirements 3. Organisational Change 4. Pre-Production Induced 5. Organisational Culture 6. Interface and Sub-System Intricacy 7. System Scale	1. Linear of Hierarchical Complexity 2. Non-Linear Complexity 3. Adaptive Complexity	1. Requirement Complexity Measures 2. System Decomposition Complexity Measures 3. Interface & Sub-system Complexity Measures 4. Management Complexity Measures	1. Evaluation / Visualisation 2. Numerical Analysis 3. Process Efficiency 4. Improved Architectures 5. Focus On Complex Area 6. Preview Problems	1. Design Optimisation 2. Immature Requirements 3. Organisational Change 4. Pre-Production Induced 5. Organisational Culture

Table 145 - Complexity characteristic type summary.

The tool itself does not relate to every single aspect of each element within the framework (shown in Table 145), however it does provide an insight to a number of these aspects:

Definition Types – Measures within the tool can clearly show the level of intricacy and the scale of the system. With the additional commonality and maturity measurements included within the measurement tool, the understanding of the system and its component parts can be assessed.

Cause Types – The tool measures a series of complexities that can be related to various causes or origins of complexity within systems. The main focus is the intricacy between systems, along with maturity and the scale of the system.

Concepts and Classification Types – The measurement tool is able to offer data that can enable the user to determine the level or likelihood of non-hierarchical or hierarchical structures within the system. Currently the tool is unable to assess a system for adaptive behaviour.

Measurement Types – Obviously the measurement tool is a measurement approach in itself. Due to the nature the data used it does not map to requirements or management complexity measures. This tool does not deal with requirements and the management complexities assessed were difficult to obtain due to the problems collecting the required data. However decomposition of the system can be tested and complexity found within individual sub-systems or the system as a whole, and the links and element complexities form a basis for system intricacy measurement.

Coping Mechanism / Approach Types - In itself the complexity measurement tool is an approach to understanding the complexity within a system, and in doing so can enable the user to focus on the complexity area or areas within a system during development, it can enhance the user ability in improving a system architecture, perhaps reducing intricacy or the dynamic nature of the system, and it itself is a numerical analysis.

Problem Types - The measurement approach does not specifically link to problem types within the framework, but with linkages to complexity origins, and the links between problem types and origins, the tool can go some way to enhancing the ability of an organisation in predicting problem issues that may arise.

The enhancement of the tool from the first pass incorporating maturity, modified commanlity and spread data improved the output and enabled the complexity within the system to be located with much more accuracy.

A potential use of the of the tool is highlighted, where it may be used to monitor complexity levels within systems and determine how much complexity grows as the elements and interfaces are designed and how it changes as the design matures. The

differences between the complexities of the conceptual design of the system and the actual implementation could give an indication to the level of induced complexity which has arisen as a result of the design process within the system; however this needs exploration in further work to determine its usefulness.

The tool could still be further improved, with integration into other systems development tools to reduce the laborious data inputting required. As a standalone tool, the level of effort required to produce the results is high, with the integration into other design or development tools this approach could be automated and not required the current level of data entry.

12 Conclusion

The following chapter details the conclusions of this thesis. The conclusions refer back to the discussion within chapter 11 against the aims set out within section 1.3.

12.1 Introduction

The following section details the conclusions of the research, and refers back to the original aims of the research in order to determine the level of success of the work.

12.2 What were the Original Aims and Objectives of this Thesis?

The Aims of this thesis are:

1. **To produce a framework for further research in complexity within engineering.**
2. **To provide valuable input to the complexity theme within the SEIC.**

The Objectives of this thesis are:

1. **To develop an analysis technique that can be used to create a common understanding, appreciation and concept of complexity within systems engineering in an industrial context.**
2. **To determine complexity characteristics within real life systems using metrics and analysis techniques.**
3. **To provide a system wide view of complexity within the interfaces and sub-systems independently as well as a system whole.**
4. **To validate the tool using real systems and conceptual systems.**

The following section details whether or not these objectives were met by the research carried out for this thesis.

12.3 Have the Aims been Met?

To produce a framework for further research in complexity within engineering.

The framework developed within this research provides industry with the ability to structure research into complexity within engineering. Research activities can be

focused on different aspects of the Complexity Framework, and the relationships within the framework to other complexity components will provide a basis to establish how that research will influence and be influenced by other research activities. The Complexity Framework is a good way of structuring, managing and understanding how different research activities will fit, what is required and also which areas are weak and need development.

Overall the objectives have for the most part been met, the lack of validation is a weakness within the research, and also the fact the model has only been applied to completed systems is not the optimum point within the engineering lifecycle that the analysis tool should be applied. Due to practicality the analysis tool had to be applied to completed systems, as the timescales associated with projects were too long (10 or more years in some cases), ideally the tool should be tested by analysing complexity at the different phases of the lifecycle and any predictions or findings made with the tool stored and compared with the system as it developed. However, despite this the tool does measure complexity characteristics within systems effectively, and provides a basis for further work and incorporation into the conceptual phases of development where it may be of better use.

To provide valuable input to the complexity theme within the SEIC.

The research carried out within the research fits neatly into the complexity theme within the SEIC at Loughborough University, and provides the theme with a structure it can use for further research activities.

12.4 Have the Objectives been Met?

The following section will take each of the thesis objectives in turn and assess the level to which the research met the objective:

To develop an analysis technique that can be used to create a common understanding, appreciation and concept of complexity within systems engineering in an industrial context.

An analysis technique was produced using the Complexity Framework, mappings and also the development of the analysis tool. This analysis technique and the framework tool specifically address complexity within engineering systems. These components

all enable an informed and better understanding of complexity in terms of its origins, definitions, coping mechanisms, measures and classifications.

To determine complexity characteristics within real life systems using metrics and analysis techniques.

The complexity analysis tool determines which complexity characteristics are present within systems and also quantifies those characteristics using metrics. The tool takes into account a broad spectrum of complexity measurement techniques and combines those using the Complexity Framework as a basis to provide a multi-dimensional view of complexity with a system. This view not only covers the breadth of complexity characteristics key to industrial problems, but identifies key components within the systems themselves that are areas of high complexity, along with demonstrating why this is the case.

To provide a system wide view of complexity within the interfaces and sub-systems independently as well as a system whole.

The complexity analysis tool does not only analyse the complexity of overall systems, but also identifies the complexity characteristics within the system interfaces and the sub-systems.

To validate the tool using real systems and conceptual systems.

The systems tested were not conceptual systems, the systems tested were in fact proven and accepted systems that are currently in active service with the customer as a consequence, the tool produced was not validated but evaluated. Although the tool does produce what appear to be valid results, an extensive validation exercise is required and the application of the analysis tool to conceptual systems should be considered to further determine the usefulness of the tool in monitoring complexity levels within systems and determining induced complexity (complexity which exists within a system over and above what exists in the system concept).

12.5 *Further Work*

This thesis examines a measurement approach to complexity in order to enhance understanding within industry. The SEIC at Loughborough University will be continuing the complexity theme, this work will become the foundation of the future work at the SEIC and the Complexity Framework outlined here the basis of that work,

in the sense that there are other aspects of the complexity understanding that need exploration in order to give industry the tools to deal with modern systems engineering problems.

There are two key strands of work that should be carried out to further validate the approach and also improve the ability to utilise it. Firstly, the measurement approach can be expanded with more measurements and used on systems of a much larger scale with more levels of decomposition. Establishing this measurement approach on a system from the initial concept phase of development out to manufacture, producing valuable data throughout the lifecycle would be a rigorous test of the tool's effectiveness. Secondly, in its current state the tool is very laborious and difficult to use, however with tools such as System Architect and SysML providing methods of modelling systems in terms of use cases, system decompositions and structures, an approach like the one developed here could be incorporated into those tools and automated.

13 References

ADAMI, C., 2002. What is Complexity?, Wiley Periodicals, Inc.: BioEssays Wiley Periodicals, Inc.

AIR FORCE TECHNOLOGY, Air Force Technology - Nimrod MRA4 - Maritime Reconnaissance Aircraft [Homepage of SPG Media Limited], [Online]. Available: http://www.airforce-technology.com/projects/nimrod/ [03/04/2007, 2007].

AIRBUS, 2006. Airbus announces A380 Delay, *Reinforced Plastics,* **50**(7), pp. 8.

ALIZON, F., SHOOTER, S.B. and SIMPSON, T.W., 2007. Improving an Existing Product Family Based on Commonality/Diversity, Modularity, and Cost. *Design Studies,* **28**(4), pp. 387-409.

AMERICAN HERITAGE DICTIONARY, 2000. The American Heritage Dictionary of the English language.

ANDERSON, M., 2008, Lupus - Study Into 'Unknown Disease'. Available: http://www.wellcome.ac.uk/News/Media-office/Press-releases/2004/WTD002862.htm [17/03/2008, 2008].

ANDREWS, G. and HALFORD, G.S., 2002. A Cognitive Complexity Metric Applied to Cognitive Development. *Cognitive psychology,* **45**(2), pp. 153-219.

ANTUNES, L. and FORTNOW, L., 2003. Sophisitcation Revisited. In: N/A, ed, *Automata, Languages and Programming.* 1 edn. USA: Springer Berlin / Heidelberg, pp. 267-267-277.

ARTHUR, W.B., 1999. Complexity and the Economy. - **284**(- 5411),.

ASHBY, W.R., 1976. An Introduction to Cybernetics. Methuen Univ Pubs.

BAE SYSTEMS, 2007, Products & Services - BAE systems. Available: http://www.baesystems.com/ProductsServices/autoGen_10692011624.html [03/04/2007, 2007].

BARABÁSI, A., 2002. Linked: The New Science of Networks. Perseus Publishing.

BARANGER, M., 2001. Chaos, Complexity, and Entropy.

BARCLAY, I. and DANN, Z., 2000. New-Product-Development Performance Evaluation: A Product-Complexity-Based Methodology. *IEE Proceedings: Science, Measurement and Technology,* **147**(2), pp. 41-55.

BARRON, A., RISSANEN, J. and YU, B., 1998. The minimum description length principle in coding and modelling. *IEEE Trans. Information Theory,* **44**(6), pp. 2743-2743-2760.

BARTHOLOMEW, P., 1999. The Role of MDO within Aerospace Design and Progress Towards an MDO Capability. Defence Evaluation and Research Agency.

BAR-YAM, Y., 2003. When Systems Engineering Fails - Toward Complex Systems Engineering. *New England Complex Systems Institute,* .

BAR-YAM, Y., 2000, 2000-Last Update, The Butterfly Effect [Homepage of NECSI], [Online]. Available: http://necsi.org/guide/concepts/butterflyeffect.html [03/26, 2007].

BATTRAM, A., 1998. Navigating Complexity. London: Industrial Society.

BBC NEWS, 23/01/2004, 2004-Last Update, warning over MoD kit overspend [Homepage of BBC], [Online]. Available: http://news.bbc.co.uk/1/hi/uk_politics/3421309.stm [11/11, 2004].

BBC NEWS, 2008-Last Update, Airbus Confirms Super-Jumbo Delay. Available: http://news.bbc.co.uk/1/hi/business/4598779.stm [16/03/2008, 2008].

BBC NEWS, 2008-Last Update, Q&A: A380 Delays. Available: http://news.bbc.co.uk/1/hi/business/5405524.stm [16/03/2008, 2008].

BEHE, M.J., 2006. Darwin's Black Box : The Biochemical Challenge to Evolution. 10th anniversary edn. London: Free.

BENNETT, C.H., 1990. How to Define Complexity in Physics, and Why In: W. ZUREK, ed, *Complexity, Entropy and the Physics of Information.* 1 edn. California: Addison-Wesley, Redwood City California, pp. 137-137-148.

BENNETT, C.H., 1988. Logical Depth and Physical Complexity. In: ROLF HERKIN, ed, *The Universal {T}uring Machine: A Half-Century Survey.* 1 edn. Oxford: Oxford University Press, pp. 227-227-257.

BENNETT, J., FENYES, P., HAERING, W. and NEAL, M., Issues in Industrial Multidisciplinary Optimization. General Motors Research and Development Center.

BIERI, J., 1955. Cognitive complexity-simplicity and predictive behaviour. *Journal of Abnormal and Social Psychology,* **51**(1), pp. 263-263-268.

BIGGIERO, L., 2001. Sources of Complexity in Human Systems. *Nonlinear Dynamics, Psychology, and Life Sciences,* **5**, pp. 3-19.

BOARDMAN, J.T., ROBSON, G.C.A. and GARRETT, C., Towards a Methodology for Coping with Complexity: An Engineering Approach. *IEE Colloquium on `Controlling Complexity: Systems Theory and Practice' (Digest No.41),* , pp. 6,.

BOCK, C., 2005. Systems Engineering in the Product Lifecycle. *Int. J. Product Development,* **2**, pp. 123-123-137.

BOEING, 2007/01/01, 2007-Last Update, 747 Celebrating the Past, Building the Future [Homepage of Boeing], [Online]. Available: http://www.boeing.com/news/feature/747evolution/747facts.html [02/04, 2007].

BOEING, 2008, Boeing Reschedules Initial 787 Deliveries and First Flight. Available: http://www.boeing.com/commercial/news/2007/q4/071010d_nr.html [16/03/2008, 2008].

BOLTE, J.P., HULSE, D.W., GREGORY, S.V. and SMITH, C., 2007. Modelling Biocomplexity - Actors, Landscapes and Alternative Futures. *Environmental Modelling & Software,* **22**(5), pp. 570-579.

CADENASSO, M.L., PICKETT, S.T.A. and GROVE, J.M., 2006. Dimensions of Ecosystem Complexity: Heterogeneity, Connectivity, and History. *Ecological Complexity,* **3**(1), pp. 1-12.

CARDOSO, J., 2005. How to Measure the Control-flow Complexity of Web Processes and Workflows.

CARRASCOSA, M., EPPINGER, S.D. and WHITNEY, D.E., 1998. Using the Design Structure Matrix to Estimate Product Development Time, 16/09/98 1998, .

CENTER FOR THE STUDY OF COMPLEX SYSTEMS, 2006-Last Update, University of Michigan, Center for the Study of Complex Systems. Available: http://cscs.umich.edu/about/about.html#goals [12/10/2008, 2008].

CHARLES, N.C. and PHILIP, J., 2003. Systems Engineering in an Age of Complexity. , pp. 25-25-34.

CHRIS KRIDLER, 2002. Shuttle Team Shops Online for Spare Parts as Aging System Becomes Obsolete. *Florida Today,* (02/04), pp. N/A-N/A.

COMPLEX SYSTEMS COMPUTATION GROUP, 2007, MDL on The Web. Available: http://www.mdl-research.org/ [4/10/2007, 2007].

COMPLEXITY SCIENCE RESEARCH CENTRE, 2008-Last Update, complexity science research centre. Available: http://technology.open.ac.uk/ccc/csrc/ [14/09/2008, 2008].

COMPLEXITY SOCIETY, 2008, Complexity Society. Available: http://www.complexity-society.com/ [15/08/2008, 2008].

CONWAY, J.H., 2000. On Numbers and Games. AK Peters, Ltd.

CSCS, a-Last Update, the minimum description length principle (MDL). Available: http://www.cscs.umich.edu/~crshalizi/notebooks/mdl.html [4/10/2007, 2007].

CSCS, 2008-Last Update, The Study of Complex Systems. Available: http://www.cscs.umich.edu/old/complexity-eg.html [04/03/2008, 2008].

CSS, 2008, Complex Systems Society : HomePage. Available: http://css.csregistry.org/tiki-index.php [15/08/2008, 2008].

DAVID, J.E., 2008, Introduction to the Edge of Chaos. Available: http://math.hws.edu/xJava/CA/EdgeOfChaos.html [20/02/2008, 2008].

DEEM, M.W., 2005. Complexity in the Immune System. *Computers & Chemical Engineering,* **29**(3), pp. 437-446.

DEMBSKI, W.A., 2004. Irreducible Complexity Revisited. Paper edn. USA: .

DOD ARCHITECTURE FRAMEWORK WORKING GROUP, 2003. DoD Architecture Framework 1.0.

DOOLEY, K.J., 2008, Nominal Definition: Complex Adaptive Systems. Available: http://www.eas.asu.edu/~kdooley/casopdef.html [20/02/2008, 2008].

DU, D. and KO, K., 2000. Theory of computational complexity. New York : Wiley, 2000.

DUNLOP, A., EVANS, W. and RIGGE, L.,1997. Managing complexity in IC Design - Past, present, and future. *BELL LABS TECHNICAL JOURNAL,* **2**, pp. 103-125.

EDC, 2008-Last Update. Available: http://www-edc.eng.cam.ac.uk/ [15/08/2008, 2008].

EDMONDS, B., 2005-Last Update, Simulation and Complexity - How They Can Relate. Available: http://cfpm.org/pub/papers/sac.pdf.

EDMONDS, B., 1999. *Syntactic Measures of Complexity*, University of Manchester.

ELECTRIC ARTS, 2008-Last Update, Classic Live - SimCity.com. Available: http://simcity.ea.com/play/simcity_classic.php [14/09/2008, 2008].

ELIZABETH MCMILLAN, 2001. Organisation Design from a Complexity Perspective – Introducing the FRACTAL WEB.

EPPINGER, S.D., 1991. Model-based Approaches to Managing Concurrent Engineering. *Journal of Engineering Design,* **2**(4), pp. 283-283-290.

EPPINGER, S.D., WHITNEY, D.E., SMITH, R.P. and GEBALA, D.A., 1994. A Model-Based Method for Organizing Tasks in Product Development. *Research in Engineering Design,* **6**(1), pp. 1-1-13.

EUROFIGHTER JAGDFLUGZEUG GMBH, 2007, Eurofighter Typhoon - Nothing Comes Close. Available: http://www.eurofighter.com/Default.asp?Flash=True [03/04/2007, 2007].

EVANS, M.W., 1987. Software Quality Assurance and Management / Michael W. Evans, John J. Marciniak. New York ; Chichester : Wiley, c1987.: .

FEDERATION OF AMERICAN SCIENTISTS, 2008-Last Update, Data Links Group. Available: http://www.fas.org/man/dod-101/sys/ship/weaps/data-links.htm [01/03/2008, 2008].

FEDERATION OF AMERICAN SCIENTISTS, 2008-Last Update, Tactical Digital Information Links (TADIL). Available: http://www.fas.org/irp/program/disseminate/tadil.htm [01/03/2008, 2008].

FERDINANDO, B., GENUARIO, B., SIMONA, B., GIOVANNI, F., SIMONETTA, F., ADRIANA, G., CINZIA, G., PAOLO, G., ANNA, P., STEFANO, P., FERNANDO, R., ORESTINA, D.S., JUERGEN, S., ANTONIO, T. and SEBASTIANO, G., 2004. From Biodiversity and Ecosystem Functioning to the Roots of Ecological Complexity. *Ecological Complexity,* **1**(2), pp. 101-109.

FILIP LARSEN, 2001-Last Update, chaos resources; lorenz attractor. Available: http://images.google.co.uk/imgres?imgurl=http://turnex.dk/chaos/lorenz_attractor01.gif&imgrefurl=http://turnex.dk/chaos/main.html&h=300&w=300&sz=7&hl=en&start=5&um=1&usg=__gsG8kWQKB85efFpDWDwtuhSt6Uw=&tbnid=RKfk9uj_xzOOnM:&tbnh=116&tbnw=116&prev(TRUNCATED) [08/10/2008, 2008].

FRYER, P., 2008 , What are Complex Adaptive Systems?. Available: http://www.trojanmice.com/articles/complexadaptivesystems.htm [20/02/2008, 2008].

G. KEMENY, J., 1953.
Two Measures of Complexity. *The Journal of Philosophy,* **52**(24), pp. 722-722-733.

GENERAL DYNAMICS UK, 2007, General Dynamics UK - FRES & Armoured Fighting Vehicles (AFV). Available: http://www.generaldynamics.uk.com/solutions/fres.html [03/04/2007, 2007].

GONZALEZ, M.C., 2006. Contact Networks of Mobile Agents and Spreading Dynamics.

GOODMAN, N., 1966. The Structure of Appearance. Indiapolis: Bobbs-Merrill.

GORDON, J.L. and SMITH, C., 1998-Last Update, AKRI : Research : Knowledge Management Guidelines. Available: http://www.akri.org/research/km.htm [24/03/2008, 2008].

GOTTINGER, H.W.,1983. Coping with Complexity Perspectives for Economics, Management and Social Sciences Hans W. Gottinger. Dordrecht: Reidel.

GRANOVETTER, M.S., 1974. Getting a Job: A Study of Contacts and Careers.

GREEN, G.C., 2004. The Impact of Cognitive Complexity on Project Leadership Performance. *Information and Software Technology,* **46**(3), pp. 165-172.

GREGORY, A., 2000. Managing Complexity: On The Right Tracks. *Manufacturing Computer Solutions,* **6**(8), pp. 28-9, 31.

GRÜNWALD, P., MYUNG, I.J. and PITT, M., eds, 2005. *A Tutorial Introduction to the Minimum Description Length Principle. In: Advances in Minimum Description Length: Theory and Applications.* 1 edn. USA: MIT Press.

GRUNWALD, P.D. and VITANYI, P.M.B., 2003. Kolmogorov Complexity and Information Theory. *Klwwer Academic Publishers,* .

HAMILTON, S., 1999. Semiconductor Research Corporation: Taking Moore's Law into the Next Century. *Computer,* **32**, pp. 43-48.

HASKINS, C., ed, 2006. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities.* 3 edn. N/A: INCOSE.

HASTINGS, C., 1995. Building the culture of organizational networking: Managing projects in the new organization. *International Journal of Project Management,* **13**(4), pp. 259-263.

HAYLES, J., 28/08/2005, 2005-Last Update, BAE SYSTEMS Nimrod. Available: http://www.aeroflight.co.uk/types/uk/bae_systems/nimrod/nimrod.htm [03/04/2007, 2007].

HEYLIGHEN, F., 1991. Coping with Complexity: Concepts and Principles for a Support System. *Transdisciplinary Research Group,* **1**(1), pp. 39-39-55.

HEYLIGHEN, F., 2008, Complex Adaptive Systems. Available: http://pespmc1.vub.ac.be/CAS.html [20/02/2008, 2008].

HITCHINS, D.K., 1992. Putting Systems to Work. 1 edn. USA: Chichester; New York : Wiley.

IBM, 2004. The Industry Challenges for Systems Engineering - *An Introduction. 1st* edn. Birmingham, United Kingdom: IBM.

ICC, 2008, ICC - Institute for Complexity Sciences. Available: http://www.listaweb.com.pt/icc/ [15/08/2008, 2008].

IEEE, 2006. Establishing Moore's Law. *IEEE Annals of the History of Computing.Vol.28,* **28**, pp. 62.

IEEE, 1992. Standard Glossary of Software Engineering Terminology. *New York: Institute of Electrical and Electronic Engineers.*

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, 1990. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. 1 edn. New York: Institute of Electrical and Electronics Engineers.

ISO, 1999. *Terminology Work - Principles and Methods.* ISO/TC 37/SC 1 N 185 E. Sweden: International Organisation for Standardisation.

JAMES D. ANDREW, 2004. *Delivering Network Enabled Capability: Industrial Procurement and Policy Challenges for the UK.* FOI-R--1278--SE. Stockholm: Swedish Defence Research Agency.

JANS, R., DEGRAEVE, Z. and SCHEPENS, L., 2008. Analysis of An Industrial Component Commonality Problem. *European Journal of Operational Research,* **186**(2), pp. 801-811.

JIM, G., 25/10/2007, 2007-Last Update, Climate is too Complex for Accurate Predictions [Homepage of New Scientist Environment], [Online]. Available: http://environment.newscientist.com/article/dn12833-climate-is-too-complex-for-accurate-predictions.html [05/03/2008, 2008].

JOHNSON, J., 2006. Can Complexity Help us Better Understand Risk? Milton Keynes UK: Department of Design and Innovation, The Open University.

KAUFFMAN, S.A., 1993. The Origins of Order: Self-Organization and Selection in Evolution. New York, N.Y. : OUP, 1993.: .

KEMAL A. DELIC, RALPH DUM, 2006-Last Update, On the Emerging Future of Complexity Sciences. Available: http://www.acm.org/ubiquity/views/v7i10_complexity.html [08/10/2008, 2008].

KESSELER, E. and VANKAN, W.J., 2006. Multidiscilplinary Design Analysis and Multi-Objective Optimisation Applied to Aircraft Wing.

KIM, K. and CHHAJED, D., 2000. Commonality in Product Design: Cost Saving, Valuation Change and Cannibalization. *European Journal of Operational Research,* **125**(3), pp. 602-621.

KIMON, S., 2007-Last Update, An Introduction to Computational Complexity - Part A. Available: http://users.forthnet.gr/ath/kimon/CC/CCC1b.htm [4/10/2007, 2007].

KOCH, P.N., SIMPSON, T.W., ALLEN, J.K. and MISTREE, F., 1997. Statistical Approximations for Multidisciplinary Design Optimization: The Problem of Size.

LANCASTER, A., 24/07/2000, 2007-Last Update, Life at the Edge of Chaos [Homepage of N/A], [Online]. Available: http://socrates.berkeley.edu/~alexl/cribbings-life.html [03/24, 2007].

LANGEFORS, B., 1967. Theoretical Analysis of Information Systems. 2 edn. N/A: Studentlitteratur , 1967.

LEES, B., BRANKI, C. and AIRD, I., 1995. A Framework for Distributed Agent-Based Engineering Support. *Automation in Construction,* **c1**(c1), pp. 631-631-637.

LEMAY, P., 1999-09-14, 1999-Last Update, Entropy and Complexity. Available: http://tecfa.unige.ch/~lemay/thesis/THX-Doctorat/node139.html [4/11/2007, 2007].

LI, M., 1997. An Introduction to Kolmogorov Complexity and its Applications. 2nd edn. New York: Springer.

LI, X., JIN, Y.Y. and CHEN, G.R., 2003. Complexity and Synchronization of the World trade Web. - **328**(- 1-2), pp. - 296.

LLC, 2008-Last Update, Mystery Diseases and Syndromes. Available: http://www.thrivingnow.com/for/Health/mystery-diseases-and-syndromes/ [29/02/2008, 2008].

LÖFGREN, L., 1974. Complexity of Descriptions of Systems: A Foundational Study. *International Journal of General Systems,* **3**, pp. 197-197-214.

MAGEE, C.L. and WECK, O.L., 2004. Complex System Classification, 20 June - 24 June 2004 2004, INCOSE.

MANKINS, J.C., 1995. Technology Readiness Levels.

MARSHALL, R., 1997. Holonic Product Design (HPD): A Workbook. 1.2 edn. Loughborough University: Loughborough University.

MARTIN, PIERRE-ALAIN J. Y., 2004. *A Framework for Quantifying Complexity and Understanding its Sources: Application to two Large-Scale Systems*, Massachusetts Institute of Technology.

MAS LAB, 2008-Last Update, MAS Lab Projects Current. Available: http://mas.cs.umass.edu/research/#current [19/08/2008, 2008].

MAS LABS, 2008a-Last Update, MAS Lab Projects Past. Available: http://mas.cs.umass.edu/research/#past [19/08/2008, 2008].

MAS LABS, 2008b-Last Update, Multi-Agent Systems Lab: Publications from 2008. Available: http://mas.cs.umass.edu/pub/year.html/2008 [19/08/2008, 2008].

MCCABE, T.J., 1976. A Complexity Measure. *IEEE Transactions on Software Engineering,* **SE-2**(4), pp. 308-308-320.

MCDERMID, J.A., 2000. Complexity: Concept, Causes and Control. *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS,* , pp. 2-9.

MIKULECKY, D.C., 2007. Definition of Complexity. http://www3.vcu.edu/complex/ON%20COMPLEXITY.html edn. Medical College of Virginia Commonwealth University.

MODIS, T., 2002. Forecasting the Growth of Complexity and Change. *Technological Forecasting and Social Change,* **69**(4), pp. 377-404.

MOLLICK, E., 2006. Establishing Moore's Law. *IEEE Annals of the History of Computing,* **28;**, pp. 62-75.

MOODY, J.A., 1997. Metrics and Case Studies for Evaluating Engineering Designs Jay Alan Moody...[et al.]. Upper Saddle River, N.J.: Prentice Hall PTR.

MOON, T., , relative entropy and mutual information. Available: http://www.engineering.usu.edu/classes/ece/7680/lecture2/node3.html [4/10/2007, 2007].

MUSÈS, C., 2002. Simplifying Complexity: The Greatest Present Challenge to Management and Government. *Kybernetes: The International Journal of Systems & Cybernetics,* **31**(7), pp. 962-988.

MYERS, N., 1992. The Primary Source: Tropical Forests and our Future. New York: Norton.

NASA, 2008, Multidisciplinary Optimization Branch Technical Archive. Available: http://mdob.larc.nasa.gov/index.html [24/03/2008, 2008].

NATIONAL INSTRUMENTS, 2008, NI LabVIEW - The Software That Powers Virtual Instrumentation - Products and Services - Mational Instruments. Available: http://www.ni.com/labview/ [20/02/2008, 2008].

NAVAL TECHNOLOGY, 2007, Naval Technology - CVF - RN Future Aircraft Carrier [Homepage of SPG Media Limited], [Online]. Available: http://www.naval-technology.com/projects/cvf/ [03/04/2007, 2007].

NAVAL-TECHNOLOGY, 2007, SSN Astute Class - Attack Submarine [Homepage of SPG Media Limited], [Online]. Available: http://www.naval-technology.com/projects/astute/ [03/04/2007, 2007].

NECSI, 2008, New England Complex Systems Institute Welcome Page. Available: http://necsi.org/ [15/08/2008, 2008].

NETINANT, P. and COMPUTER SCIENCE RESEARCH, EDUCATION, AND APPLICATIONS PRESS, Reducing Complexity of System Software Design. *PDPTA -INTERNATIONAL CONFERENCE-; 2004,* , pp. 1280-1284.

NEWCASTLE UNIVERSITY ENGINEERING DESIGN CENTRE, 2008-Last Update, EDC: Complex Process Integration and Coordination. Available: http://www.edc.ncl.ac.uk/research/sub_complexprocess.php [08/10/2008, 2008].

NEWCASTLE UNIVERSITY ENGINEERING DESIGN CENTRE, 2006-Last Update, November 2006: Complexity and Complex Systems. Available: http://www.edc.ncl.ac.uk/highlight/rhnovember2006.php/ [08/10/2008, 2008].

NIGEL GILBERT, 2004. Agent-Based Social Simulation: Dealing with Complexity. University of Surrey.

OXFORD ENGLISH DICTIONARY, 2007, Oxford English Dictionary - Intricacy. Available: http://dictionary.oed.com/cgi/entry/50120159?single=1&query_type=word&querywor d=intricacy&first=1&max_to_show=10 [03/04/2007, 2007].

OXFORD UNIVERSITY, 1992. The Oxford English Dictionary (second edition) on Compact Disc. Oxford : Oxford University Press, 1992-1995: .

PAPADIMITRIOU, C.H., 1994. Computational complexity / Christos H. Papadimitriou. 1 edn. USA: Addison Wesley.

PARRY, M.L., ROSENZWEIG, C., IGLESIAS, A., LIVERMORE, M. and FISCHER, G., 2004. Effects of Climate Change on Global Food Production Under SRES Emissions and Socio-Economic Scenarios. - **14**(- 1),.

PERONA, M. and MIRAGLIOTTA, G., 2004. Complexity Management and Supply Chain Performance Assessment. A Field Study and a Conceptual Framework. *International Journal of Production Economics,* **90**, pp. 103-115.

PERROW, C., 1999. Normal Accidents Living with High-Risk Technologies Charles Perrow. Princeton, NJ: Princeton University Press.

PIKE, J., 2008-Last Update, Future Combat Systems (FCS) / Future Combat System (FCS). Available: http://www.globalsecurity.org/military/systems/ground/fcs.htm [03/04/2007, 2007].

POON, L. and GREBOGI, C., Controlling Complexity. *Physical Review Letters,* **75;**(22), pp. 4023-4026.

R & F DEFENCE PUBLICATIONS, 2007-Last Update, Armed Forces - British Army - The Royal Engineers - Terrier [Homepage of R & F Defence Publications], [Online]. Available: http://www.armedforces.co.uk/army/listings/l0109.html [03/04/2007, 2007].

REINHARD, D., 2005. Graph Theory. Electronic Edition edn. New York: Springer-Verlag Heidelberg.

RICHMOND, S.A., 1996. A Simplification of the Theory of Simplicity. *Synthese,* **107**, pp. 373-373-393.

SANTA FE INSTITUTE, 2008, Santa Fe Institute. Available: http://www.santafe.edu/ [15/08/2008, 2008].

SEIC, L.U., 2008, Systems Engineering Innovation Centre. Available: http://www.seic-loughborough.com/AllPages.html?Page=1 [03/03/2008, 2008].

SENGE, P.M., 1994. The Fifth Discipline Fieldbook Strategies and Tools for Building a Learning Organization. London: Brealey.

SHALIZI, C.R., 2008-Last Update, Complexity Measures. Available: http://www.cscs.umich.edu/~crshalizi/notebooks/complexity-measures.html [12/10/2008, 2008].

SHANNON, C.E., 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal,* **27**, pp. 379-379-423, 623-656.

SHAW, W.H., 2002, Engineering Management in our Modern Age. *2002 IEEE International Engineering Management Conference.Proceedings (Cat.No.02CH37329),* , pp. 9,.

SHENHAR, A.J., DVIR, D., LEVY, O. and MALTZ, A.C., 2001. Project Success: A Multidimensional Strategic Concept. *Long range planning,* **34**(6), pp. 699-725.

SHIH, B.Y. and JANET, E., 2007-Last Update, An Introduction of Network Complexity. Available: http://www.ifm.eng.cam.ac.uk/mcn/pdf_files/part6_7.pdf [01/10, 2007].

SMITH, R.P. and EPPINGER, S.D., 1995. Identifying Controlling Features of Engineering Design Iteration. 1 edn. Massachusetts: Massachusetts Institute of Technology.

SPIEGELMAN, M., 1997. An Introduction to Dynamical Systems and Chaos.

STANHILL G., 2007. A Perspective on Global Warming, Dimming, and Brightening. **- 88**(- 5),.

STEVENS, R., 1998. Systems Engineering Coping with Complexity Richard Stevens et. al. London: Prentice-Hall.

SUSSMAN, J.M., 1999. The new transportation faculty: The evolution to engineering systems. *Transportation Quarterly,* **53**(3), pp. 15-26.

SYSML PARTNERS, 2007, SysML - Open Source Specification Project. Available: http://www.sysml.org/ [9/18/2007, 2007].

SZABO, N., 2003, Introduction to Algorithmic Information Theory. Available: http://szabo.best.vwh.net/kolmogorov.html [4/10/2007, 2007].

TAXÉN, L., 2003. *A Framework for the Coordination of Complex Systems' Development*, Department of Computer and Information Science, Linköpings University.

TAYLOR, C., 2003. UK Defence Procurement Policy.

TELEGRAPH, 2008, Boeing's New 787 'Delayed' Yet Again. Available: http://www.telegraph.co.uk/money/main.jhtml?xml=/money/2008/03/08/cnboeing108.xml [16/03/2008, 2008].

TELELOGIC, 2008-Last Update, Requirements Management for Advanced Systems and Software. Available: http://www.telelogic.com/products/doors/doors/index.cfm [20/02/2008, 2008].

TELELOGIC, 2008-Last Update, Telelogic Rhapsody: Model Driven Development for Maximum Productivity. Available: http://www.telelogic.com/products/rhapsody/index.cfm [01/03/2008, 2008].

TELELOGIC, 2008-Last Update, Telelogic System Architect: *Is a Comprehensive* Business Process Modeling Solution Designed to Develop Enterprise Systems. Available: http://www.telelogic.com/products/systemarchitect/index.cfm [01/03/2008, *2008].*

TESSON, K.J., 2006. Dynamic Networks, An Interdisciplinary Study of Network Organization in Biological and Human Social Systems.

THEVENOT, H.J. and SIMPSON, T.W., 2007. Guidelines to Minimize Variation when Estimating Product Line Commonality through Product Family Dissection. *Design Studies,* **28**(2), pp. 175-194.

UBIQUITY, 2008-Last Update, About Ubiquity. Available: http://www.acm.org/ubiquity/about.html [08/10/2008, 2008].

UC IRVINE, 1996-Last Update, Minimum Spanning Trees. Available: http://www.ics.uci.edu/~eppstein/161/960206.html [17/03/2008, 2008].

UK MINISTRY OF DEFENCE, 2005a. MoDaf Architecture Framework Version 1.1.

UK MINISTRY OF DEFENCE, 2005b. Network Enabled Capability. 1 edn. United Kingdom: UK Ministry of Defence.

UK MINISTRY OF DEFENCE, 1999. Smart Acquisition Handbook. 1 edn. UK: UK Government.

UK MINISTRY OF DEFENCE, a-Last Update, Defence Internet | MicroSite | Future Rapid Effect System (FRES). Available: http://www.mod.uk/DefenceInternet/MicroSite/DPA/OurTeams/FutureRapidEffectSystemfres.htm [03/04/2007, 2007].

UK MINISTRY OF DEFENCE, b-Last Update, Introduction to the Acquisition Operating Framework (AOF). Available: http://www.aof.mod.uk/aofcontent/aofintro.htm [03/03/2008, 2008].

UK MOD, 29/10/01, 2001-Last Update, Ministry of Defence, Defence Issues, Aquisition and Procurement [Homepage of UK MoD], [Online]. Available: http://www.mod.uk/issues/index.htm [11/29, 04].

ULTIMATE BUSINESS RESOURCE, 2007. Understanding the Basics of Project Management.

VAKIL, S.S. and HANSMAN R.J., J., 2002. Approaches to Mitigating Complexity-Driven Issues in Commercial Autoflight Systems. *Reliability Engineering and System Safety,* **75**(2), pp. 133-145.

VALERDI, R., BOEHM, B., REIFER, D. and INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING, 2003. COSYSMO: A Constructive Systems

Engineering Cost Model Coming of Age. *INCOSE -ANNUAL CONFERENCE SYMPOSIUM PROCEEDINGS- CD ROM EDITION,* .

VITANYI, PAUL M. B. and MING, L., 2000. Minimum Description Length Induction, Bayesianism, and Kolmogorov Complexity. *IEEE TRANSACTIONS ON INFORMATION THEORY,* **46**(2), pp. 446-446-464.

VOLTERRA CONSULTING LTD, 2003. An Agent-Based Model of the Extinction Patterns of Capitalism's Largest Firms.

VOLTERRA CONSULTING LTD, 2002. An Agent-Based Model of the Evolution of Market Structure and Competition.

WALDROP, M.M., 1987. Man-Made Minds: The Promise of Artificial Intelligence. New York: Walker.

WARREN WEAVER, 1948. Science and Complexity. *American Scientist,* **36**(08/10/2008), pp. 536.

WARWICK UNIVERSITY, 2008-Last Update, Warwick Complexity. Available: http://www2.warwick.ac.uk/fac/cross_fac/comcom [12/10/2008, 2008].

WASHINGTON, W.M., 2005. The Computational Future for Climate Change Research. *National Center for Atmospheric Research,* **16**, pp. 317-317–324.

WATZLAWICK, P., 1967. Pragmatics of Human Communication: A study of Interactional Patterns, Pathologies, and Paradoxes. W W Norton.

WEILKIENS, T., 2008. Systems Engineering with SysML/UML: Modeling, Analysis, Design (The OMG Press) (The MK/OMG Press). Morgan Kaufmann.

WIENER, N., 1965. Cybernetics or control and communication in the animal and the machine. Cambridge Mass.: The MIT Press.

WIKIPEDIA CONTRIBUTORS, 2008-Last Update, Agent-Based Model [Homepage of Wikipedia, The Free Encyclopedia], [Online]. Available: http://en.wikipedia.org/wiki/agent-based_model?oldid=241095322 [12/10/2008, 2008].

WIKIPEDIA CONTRIBUTORS, 2008-Last Update, Lorenz Attractor [Homepage of Wikipedia, The Free Encyclopedia], [Online]. Available: http://en.wikipedia.org/wiki/lorenz_attractor?oldid=238954405 [12/10/2008, 2008].

WILKIE, F.G. and HYLANDS, B., 1998. Measuring Complexity in C++ Application Software. *Software, Practice & Experience,* **28**(5), pp. 513-546.

WILSON, T.R., 2002. Global Threats and Challenges. *Defense Intelligence Agency,* .

WOLFRAM, S., 1994. Celluar Automata And Complexity: Collected Papers. Westview Press.

14 Appendix A – First Case Study Set Details

Each table shows a different portion of a larger table which for the purpose of this thesis has been re-created in 3 sections. The following table describes those table sections:

Data Type	Table 147	Table 148	Table 149
Programme Case Study			
Brief			
Problem Number			
Problem			
Nature of Complexity			
Driver For Problem			
Lifecycle			
Comments			
Organisational Characteristics			
Organisational Configurational Inducers			
Internal Factors			
External Factors			
Origins of Complexity			
Problem Issues			
Complexity Definitions			
Approach to Problem			
Coping Measures Employed			
Concepts / Classifications			
Complexity Measures			

Table 146 - Description of table contents for the tables detailing the first case study set findings and cross referencing with complexity characteristics.

The tables are labelled as follows:

- Table 147 - Table of the first case study set, detailing the problems, the point in the lifecycle the problem occurred, the drivers of the problems and the nature of the complexity in that problem with comments.
- Table 148 - Table of the first case study set, showing the relevant origins of the complexity for each case
- Table 149 – Shows the various different complexity characteristics within each problem.

14.1 Table 147 – Part 1 of the Complexity Matrix

Programme Case Study	Brief	Problem Number	Problem	Nature of Complexity	Driver For Problem	Lifecycle	Comments
Upgrade Programme	An upgrade of an existing legacy system, this upgrade required the contractor to maintain the functionality of the current system. Work was initiated with a large amount of reverse requirements engineering.	UP01	Organisational Problem - Business units within the same company were involved in a prime, sub-contractor relationship, as well as the relationship with the second company sub-contractor. The technical requirement was not well specified and the roles and responsibilities were not well defined or well respected. People were relaxed about it.	The relationships of the organisation were understood and the problems that were caused were overcome with a relaxation of the contractor boundaries within the prime contracting company. This relaxation and relationship change gave the sub-contracting business unit the ability to deal with the external company directly, if not officially.	Existing organisation and project structure. The timescale, having to do development before the requirements were sorted. MoD has specific contractor handling processes. Company first and time pressures.	Problem Occurred In Stage(s): Development The Problem Affects Stage(s): Development	Work was a problem because they didn't know what they were doing initially; specify the problem a bit more clearly. The lack of a well specified technical requirement.

		UP02	Frequently software was implemented on the hardware that was available at the time, rather than the intended hardware. The maturity level of the requirements used to develop the software was an issue. The low level of maturity within the requirement set, meant that software developed for the first incremental build may need complete alteration for build 2, as requirements that matured between those incremental stages become available for the next software builds. They had an effect on the initial submitted build, and this meant re-design, because these requirements were not available early in the programme, build 1 could not be developed with these requirements in mind to enable a smooth transition in the incremental development cycle. As a result here, the re-work was enormous as a result of this lack of maturity.	Could be a complexity issue, this arises as a result of maturity in the design and difficulty in linking the development programmes for hardware and software together. Also this is as a result of immature requirements and the need to develop software without knowing the nature of the complete product.	Timescales, forced to take on the work despite the lack of understanding of the functionality of the system, politics, commercial requirements.	Problem Occurred In Stage(s): Requirements Definition, Concept The Problem Affects Stage(s): Requirements Definition, Concept, Design, Delivery (Timescales), Acceptance	Re-engineering existing functionality that is not well defined. Developing against poorly defined, or undefined requirements that later get defined, better defined or redefined. 'Re-implementation Plus' of existing system that is a 'black box'

		UP03	The project was essentially an upgrade of the legacy system. As a requirement the customer wanted the functionality of the legacy system to remain with perhaps added functionality along with improved performance and reliability. The legacy system was not easily understood, due to a lack of a complete specification of the system. Modifications during the product life may have been contributing factors to the lack of understanding as to how the product works. In order to enable the customer requirements to be met, the functionality that must be maintained must be reverse engineered into a specification of some kind. This reverse engineering exercise, especially within the BU2, was focused on the software actions of the system.	The lack of maturity of the new customer requirements meant that the working of these requirements was being carried out in parallel with the reverse engineering exercise. The platform specific hardware requirements generated from the reverse engineering and the new requirements generated from the customer gave an incompatibility in requirements between the two. This incompatibility produced a "systems engineering tension" (Brian J. Wilkinson). Caused by a lack of understanding of the requirements for the reverse engineering exercise, the functionality needed to be reverse engineered, not the entire software system.	To understand how the legacy system works in detail, and provide a requirements set based upon that system so the functionality could be maintained across to the new system.	Problem Occurred In Stage(s): Requirements Capture, Concept The Problem Affects Stage(s): Requirements Capture, Concept, Design (knock on effect on the stages after this, due to the timescale impact that the invalid or inappropriate requirements had on the work load for the project)	Re-engineering existing functionality that is not well defined, in terms of the software, the software was completely reverse engineered including aspects of the previous software design that were not directly attributable to the functionality.

		UP04	The design certification approval process was slow, as all design certification and approval was carried out at the prime contractor level. This caused delays in the programme and increased the programme costs.	The problem is as a result of the contractual relationships between the prime and sub-contracting business units within BAE, part of this relationship keeps the approval at the prime contract level, which in a contractual relationship of this kind is acceptable, however in this instance it is not required to be a stringent as the prime and sub are both members of the same overall company. The problem is caused by the processes that are used as a result of this strange relationship.	Time, costs and internal governance.	Problem Occurred In Stage(s): Design, Testing, and Acceptance The Problem Affects Stage(s): All the previous and perhaps beyond because of the slow nature of the approval process, the impact is felt along the whole programme in terms of the time scales.	This approval system is often a good idea, the approval should be taken away from the sub-contractor, but in some cases the nature of the programme itself means that this can cause a tremendous inefficiency in the progress of the work.
		UP05	The mindset of the people within the prime organisation has an influence on the overall programme. Some staff considered the programme a product support programme, and as a result was only interested in these aspects, refusing to agree that in reality this was essentially a new product. The upgrade was considered a basic upgrade support job. This reduces the agility of the organisation to respond to the volatility of the programme nature. This creates a massive amount of organisational inertia to overcome. BU2 considered it as a development contract.	Human factors, mindsets, the nature of the organisation that is carrying out the work, dealt with support programmes so assumed this was one. Cultural differences	Different perspectives, processes, procedures, and processes that are inappropriate to the work. Cultural difference.	Problem Occurred In Stage(s): Initial concept phases The Problem Affects Stage(s): All development and concept phases beyond, until mindset change.	

Lancer	Competition between Bae and GEC before the merger, when the two companies were forced to maintain the boundary split and isolation in order to maintain the competition environment. The systems involved were Lancer (this case) and SIKA. The programmes produced two prototype land reconnaissance vehicles. The two systems would compete for the FRES and FCS programme options for the reconnaissance vehicle.	LA01	Over familiarity between the members of RO defence and C4ISR due to the close proximity meant that the processes were often relaxed and documentation and interface specifications were not always kept up to date.	No, this is just a result of putting two sets of people that already have a level of familiarity in an environment where they were allowed to mix and control was not taken over documentation. Incidentally this was not a major problem on the programme. But it did cause a lack of common assumptions for the Lancer vehicle and lack of accurate recording.	Speed agility and the familiarity that already exists between the two business units. Also the proximity of the business units helped as it allowed easy communication; as a result some development process was relaxed.	Problem Occurred In Stage(s): Concept phases and design phases The Problem Affects Stage(s): All from then on.	Lancer was not a complete programme in its own right. It was a technology demonstrator that existed in conjunction with a second programme called SIKA. Not all the phases of development are applicable to this case study, as not all phases were carried out.
		LA02	Lack of common objectives of business units (BUs) within BAE, means that BUs will always seek to retain work rather than feed it out to other areas of the business that are better equipped.	There are not a large number of interactions it's just a state of ultimate competition internally. A modification of BAE SYSTEMS values may change this approach. Important to realise though that there are advantages to this approach, without it those business units that are completely incapable of making their own profit would not easily be identified.	Each business unit has its own objectives and must achieve those targets.	Problem Occurred In Stage(s): All The Problem Affects Stage(s): All	
		LA03	Engine controller failure, the engine controller kept overloading and failing during the trails for the Lancer vehicle. The cause was unknown, every time engineers approached the vehicle the handbrake was applied and the vehicle switched off, there was no obvious cause of the failure of the solid state control system for the motors.	The cause of the fault was not difficult to find, and this in itself is evidence to show the problem is not complex.	Lack of understanding of the vehicle controls? Or perhaps a poor man machine interface.	Problem Occurred In Stage(s): Development phases when developing the concept of use and man machine interfaces The Problem Affects Stage(s): Testing and design proving.	

		LA04	The coiling of the cable for the sensor pack was having trouble retracting into the hull of the vehicle, due to its size and weight.	A engineering problem within the design that was rectified.	Need to raise and lower the sensor suite.	Problem Occurred In Stage(s): Development and testing phases The Problem Affects Stage(s): Testing and Acceptance	
		LA05	The map display for the vehicle was out by quite a large margin. The mapping system is used to display the vehicle location in relation to the terrain around it, a mixing of GPS data and surface map data. The surface map is skewed due to the curvature of the earth; essentially it goes through a flattening. GPS does not flatten the maps and maintains position in relation to the curvature of the earth.	Was a design decision and fault finding exercise, the problem was easily identified and corrected.	Lack of understanding of the mapping systems used and the lack of sync between the GPS and ordinance maps.	Problem Occurred In Stage(s): Development stages (unknown at this point) The Problem Affects Stage(s): Testing	
		LA06	ESIL, CSIL integration process not followed during the development phase for the system. I have not really identified why this was a problem as such, I don't think anything too negative ever came from it.	This is simply a choice not to follow procedures laid down early in the development planning, probably due to budget constraints and time constraints.	Reducing cost and timescales.	Problem Occurred In Stage(s): Conceptualisation, Development phases including integration etc. The Problem Affects Stage(s): Integration, testing, acceptance.	Not really a problem as such more a comment.

Marksman	An anti-aircraft tank turret, with two guns and high performance swivel turret mounting, containing a new radar concept and predictor system. The product was highly successful in testing, but was not easily sold due to constraints on budget and a lack of a capability requirement for it.	MK01	Drawing packs were not updated as the programme neared completion. There were a set of manufacturing drawings for the turret, along with a very large set of drawing changes. This meant that the manufacture of the system was difficult from the original set as there would be references to several changes, some of which may be contradictory to each other.	This result was caused by a lack of funds and interest in the product. If the product had shown more promise perhaps this work would have been carried out. Leaving the drawing packs means that the manufacture and support for the system is very difficult, this could be referred to as induced complexity.	Reduce costs of the programme, lack of a market for the product that was created. Aim to finish complete the programme while reducing the amount of money spent.	Problem Occurred In Stage(s): Detailed Design, Acceptance, Manufacturing The Problem Affects Stage(s): Manufacture, Product Support	
		MK02	There was no pre-production work carried out on the Marksman system. Essentially this was done to reduce the budget required to complete the delivery to the Fins. This lack of pre-production essentially meant that each system was a custom build. 6 Systems were produced in the end and there was a lack of commonality between all of them making the maintenance activity very awkward.	Increased the cost of the support for these systems. The problem has increased the complexity of the support of the systems, since all are unique. The complexity is reduced commonality.	Reduce cost overall, the MoD didn't want the system; the market wasn't there for the product. There was no clear requirement for the product. Produce a set of prototypes that could then be sold. Produce demonstrators to provide a market.	Problem Occurred In Stage(s): Detailed Design, Acceptance, Manufacturing The Problem Affects Stage(s): Manufacture, Product Support	

Astute	Background – Initially the Astute programme was contracted to GEC, after the merger between GEC and Bee, the contract was for a complete platform, the first of contract that actually procured the whole system from a single company. The contract initially was set out as a procurement exercise and was handled by the Procurement Contacts Office (PCO). The programme estimated cost was in the order of £3bn and has been running for a total of 9 years to date (02/05/05). The submarine is essentially a replacement for the Swiftshore class submarine, and is intended to be a stealthy smaller boat. The original requirement was set against the cold war threat, which has obviously is no longer valid as a requirement	AT01	The astute requirement set was originally 10,000. Most of these requirements were deemed to be erroneous, incomplete, inconsistent, or design constraining. The initial programme was set up as a procurement programme managed by the Procurement Contracts Office (PCO) rather than treated as a separate programme. Managing these requirements had a massive cost in both money and time. The erroneous requirements set needed to be changed or better defined in order to continue with the work.	The issue here is that there was a steering towards a procurement exercise rather than a design exercise, the Procurement Contracts Office was given control of the programme and project progression was actually measured on the number of sub-contracts placed. The changing of the requirement set from the 10,000 erroneous requirements to a more manageable and appropriate 1000 requirements based around a design programme rather than procurement programme, gave more freedom to the engineering teams to develop the system properly. The 10,000 requirements were still contractual obligations, but since the new set was also agreed it gave the company a bit more leverage in the original set and not constrain the programme so much.	Reduce the cost of the requirements management exercise, improve the understanding of the requirements, produce a more appropriate requirements set that is based on designing a system rather than procuring a system, produce a complex requirements set, produce a requirements set that can be mapped to scenarios for acceptance.	Problem Occurred In Stage(s): Pre-programme stages, bid stages, requirements gathering, conceptual design stages The Problem Affects Stage(s): Concept Design, Acceptance, System Testing, and generally reduces the workload associated with requirements management throughout the product lifecycle.	Interesting point here, do the same lifecycle components apply to a procurement programme? Or is there a different style of programme for this instance? I will check this.

	at this point in time.	AT02	In particular in relation to the requirements set was the "the product must be fit for purpose" requirement. There was no definition of what this actually meant, and therefore no method of providing compliance evidence against it.	The lack of a concise definition based on what fit for purpose actually means. The requirement is very vague and cannot be tested against. The existence of requirements like this makes the acceptance task very difficult.	The driver is to have a requirement within the requirement set that ensures the output meets the needs of the customer in the envisaged use.	Problem Occurred In Stage(s): Requirements capture The Problem Affects Stage(s): All beyond the requirements capture phase, but Acceptance in particular.	
		AT03	Similar to the previous there was a requirement that stated the platform must "perform better than previous platforms". There was no acceptance criteria established for this, essentially if the platform exhibited one better characteristic that previous platforms it would be "better". This linked back to the previous problem in terms of the solution.	The fact that the requirement acceptance criteria were not defined is just as a result of a lack of enlightenment of the systems engineering processes. The complexity is caused when trying to define how this requirement is met.	The customer requires that the contractor delivers a programme that is better than the previous system and wants to have some method to test this for acceptance.	Problem Occurred In Stage(s): Requirements capture The Problem Affects Stage(s): All beyond the requirements capture phase, but Acceptance in particular.	

		AT04	The mindset of the programme was geared towards a procurement exercise, as mentioned previously most of the measurement of progress was directly proportional to the number of sub-contracts placed for the procurement of equipment. Unfortunately, this was not just a procurement exercise but really a full blown design exercise.	The mindset was as a result of the nature of the programme, the PCO was managing the programme, so they will manage it as a procurement exercise. This is obvious, and initially was the intended strategy.	Procurement exercises are seen to be less difficult, and a large amount of technology already existed for the submarine, therefore this was done to make the programme simpler and more cost effective.	Problem Occurred In Stage(s): Requirements, Concept, and Design (lack of it due to procurement type strategy) The Problem Affects Stage(s): All, but has been somewhat rectified now.	
		AT05	VSEL originally competed with GEC for the contract for Astute. Once the contract was awarded the VSEL company was bought by GEC on the understanding it would be used to build the submarines. As manufacture work began, more and more personnel moved to the VSEL sites, and control shifted from BAE to what was VSEL. VSEL began to work as they always had worked without employing the BAE processes within their manufacture procedures.	Partly due to the fact that VSEL bid for the programme in the first place, and have a lot of experience running this kind of project. As the programme progressed through the concept stages and work began there was a shift of control from the portion of the business where the concept and design work was being carried out to the VSEL site where manufacture had started.	VSELs experience and culture.	Problem Occurred In Stage(s): Manufacturing (although this happens quite early in a procurement programme) The Problem Affects Stage(s): Manufacture	

		AT06	Service routing is extremely difficult within a submarine, there are large number of services that need to be routed and transferred within the boat. High pressure air, hydraulic fluids, fuels, low pressure air, CO2, electrical signals (sensitive, non-sensitive, low power, high power), power (high levels, low levels), etc. In an unlimited 3D space this is quite easy to achieve. But in a limited 3D space such as a submarine, and taking into account the large number of services and the constraints of the routing, the optimisation process is extremely difficult.	A complexity issue, this fits the bill nicely. There are a large number of different services, which you may call nodes. A concept that has not been clearly identified if that it is not just the information exchange between these nodes that is the problem, in this case it is not this simple. Some of the interactions are not information exchanges, but the very existence of the need for a relationship between two nodes impacts the ability of relationships between others!	Submarines are massively constrained by space, therefore the space must be used effectively, this is the main driver when routing the services, the other drivers are the standards and requirements when considering the separation of service routes, i.e. sensitive electronic services not next to high power service lines.	Problem Occurred In Stage(s): Development, Manufacture (can apply to stages where custom builds are necessary later in the development stages) The Problem Affects Stage(s): The same stages, but the routing could affect the supportability of the product, therefore operation and in service support are affected also.	I think that this is one of the first major complexity issues within the BAE SYSTEMS product space.
Test Bed	Technical demonstrator programme for the MOD, and was a 50% split between industry and the DPA. It consisted of BAE SYSTEMS and a number of other companies. Due to the politics of the time, mainly that Warton was massively overdue with its contract, it was important that this contract was done with the up most to maintain the timescales and complete the programme on time. The contract took 1 year to get agreed, and a further 5 to complete making the project running time 6 years in total.	TB01	The organisation changed dramatically from the initial stages of the programme to the final stages of the programme. The merger between BAe and GEC changed the functionality of the organisation. Projects that have to operate through these dynamic organisational environments.	Organisational change is a certainty as programmes run over longer time periods. This is not as a direct result of complexity but companies trying to remain competitive in evolving markets by purchasing other companies to integrate their technologies and services.	The changing nature of business, and the need to adapt to the market.	Problem Occurred In Stage(s): All The Problem Affects Stage(s): All	This is a generic problem that exists for all, in particular ICP.
		TB02	The DPA head can only remain the head for a maximum of 2 years, in a programme running over 5 years, there will be a number of different DPA heads.	The change of the DPA head may impact the programme, but is simply a change of management in an area that is not directly integrated into the programme team. The effect was small since it was mitigated by the education of the DPA head by the project manager.	DPA policy.	Problem Occurred In Stage(s): All The Problem Affects Stage(s): All	No set programme stages, it depends when the DPA head changes.

		TB03	The nature of the product means that FSS (Frequency Selective Surface) needed to be used in the design. An important issue in the design of these surfaces is the connections of these materials to the other more standard materials within the vehicle, how is this done? And how difficult are the manufacturing exercises required to develop these materials.	It is a case of setting the manufacturing requirements and process, and developing the appropriate joining of the two surfaces.	Technical requirements to make the design suitable.	Problem Occurred In Stage(s): Development stages, due to the nature of the product requirement. The Problem Affects Stage(s): Manufacturing and testing.	
		TB04	Multi-disciplinary design optimisation was a major problem in the design stages of the product. There are a number of key elements and parameters within the design that need to be considered. These elements are not always measurable in a numerical sense; this makes the optimisation of the product difficult. Without enormous computing power there is no way that the optimum design could be produced, as a result concepting must take precedence.	This is a complexity issue; MDDO is a complexity issue that affects a large number of programmes. The routing problem within the Astute programme is also a MDDO problem. In this case the number of parameters involved in generating the optimised design is enormous and unmanageable. It is reduced	The need to optimise the design for so many variables.	Problem Occurred In Stage(s): All bar the testing element, at this point the MDDO should have been completed. The Problem Affects Stage(s): All including the testing element.	A complexity issue for certain, and a generic complexity issue that goes for all programmes, the nature of the multi-discipline design exercise can change in terms of variables (type and number) some aspects are indefinable (shape).
Information Capability Programme (NEC)	ICP is the integration of all the BAE SYSTEM component businesses; it's essentially the development of capability that the customer has not yet realised. NEC or Network Enabled Capability as it was originally coined is the realisation of information	IP01	The ICP programme operates over the entire organisational spectrum of BAE SYSTEMS. Every change in organisational structure will affect the programme and as a result the ICP organisation must remain agile in order to cope.	Unpredictability and uncertainty in the entities with which ICP interacts. ICP has no control over the changing operational requirements for capability or changes in military policy, doctrine, etc. and their impact on the programmes with whom they interact.	The changing organisational structure of BAE SYSTEMS. Increasing need for the integrated capability within the battle space.	Problem Occurred In Stage(s): All The Problem Affects Stage(s): All	A generic problem, the changing and evolving organisational landscape.

	transfer between different elements of the battle space. The image often given by BAE SYSTEMS shows UAVs, aircraft, soldiers, ground vehicles, ships, submarines all communicating with each other and exchanging information across the battle space. This is the realisation of what is ICP.	IP02	Lack of understanding of the customer, the customer does not know what capabilities they wish to procure. The customer does not understand the nature of the ICP programme. It is up to the ICP programme to develop the capabilities that the customer does not know they require.	Could be a complexity issue, but more a risk issue. There are massive risks involved in predicting the nature of the perceived requirements for the ICP product line. If this goes wrong there are potential cost consequences.	The need to educate the customer in the capability of the ICP capability.	Problem Occurred In Stage(s): All The Problem Affects Stage(s): All	I suppose the big effect of this, is the budget that is devoted to the ICP product base. This is a difficult one and I am not sure how to fill this one in yet.
		IP03	The technology maturity level for a lot of the key technologies that the ICP programme will require are of limited readiness levels.	Could be a complexity issue, but more a risk issue. There are massive risks involved in predicting the nature of the perceived technology needs for the ICP product line, as the requirements are not mature enough to provide confidence.	The need to keep up to date with the technology that is available, and ensure that technologies that are required by the ICP element are mature when they are required.	Problem Occurred In Stage(s): Development phases The Problem Affects Stage(s): Development phases	To be able to ensure the capability is ready technology must be invested in. The only stages affected are all the development stages, without the maturity of the technology these cannot proceed, there are also impacts to other aspects of the ICP capability, if they are dependent on previous technology implementations.
		IP04	No understanding of system operation. ICP understands the technologies and their readiness levels, but does not understand how the system will impact the operational environment, until it is actually in the field. Suddenly units are provided with systems that relay information all over the battle space, there is no way of predicting exactly how this will affect the operation of the system.	Definitely a complexity issue, this relates to the emergence of behaviours within the battle space that are unpredictable. Until this system is actually in operation there is no way of telling how this information exchange will change the decision making of the individual commanders or units on the battle space. There is a chance that these emergent properties could enhance the battle field, a goal of ICP, but there is a chance that this emergence could be negative.	ICP is integrated onto BAE SYSTEMS products as a feature.	Problem Occurred In Stage(s): Development, all the stages from concept to design The Problem Affects Stage(s): Operation, they will not exhibit this emergence until the system is in operation.	

Table 147 - Table of the first case study set, detailing the problems, the point in the lifecycle the problem occurred, the drivers of the problems and the nature of the complexity in that problem with comments.

14.2 Table 148 – Part 2 of the Complexity Matrix

Programme Case Study	Brief	Problem Number	Origins of Problem				
			Organisational Characteristics	Organisational Configuration Inducers	Internal Factors	External Factors	Origins of Complexity
Upgrade Programme	An upgrade of an existing legacy system, this upgrade required the contractor to maintain the functionality of the current system. Work was initiated with a large amount of reverse requirements engineering.	UP01	Structure, role definitions, interactions and boundaries.	Prime contracting, Sub-contracting relationships and organisation, Contractual constraints	N/A	N/A	A lack of use of process, due to the familiarity. It was not seen as an issue.
		UP02	Planning for the reverse engineering exercise, project strategy, succession, design history.	At this time I can't see a relationship between the lack of maturity in the requirement, and the organisational configuration.	N/A	N/A	Lack of the original design, no understanding of the legacy equipment, lack of configuration control over upgrading.
		UP03	NA	The separation of the requirements exercise into two areas is an inducer of the complexity, if the same personnel were creating the new requirements as well as generating functional requirements from the previous system there may not have been an inconsistency.	System functionality	N/A	Lack of the original design. The reverse engineering to the original system.
		UP04	The governance of the organisation, the policies regarding the approval and certification and sub-contract management processes.	Process implementation in terms of approval and certification process setup from Prime to Sub contract. Prime contracting, Sub-contracting relationships and organisation.	N/A	N/A	Internal requirement to provide it, the internal delivery mechanism (BU2 to BU1)

Programme Case Study	Brief	Problem Number	Origins of Problem				
			Organisational Characteristics	Organisational Configuration Inducers	Internal Factors	External Factors	Origins of Complexity
		UP05	Culture, strategy, processes, function.	Not so much a configuration problem, but a mindset problem.	N/A	N/A	Didn't understand the nature of the problem in the early stages, it was allocated to the wrong organisation. Technology was old and the next development would be an update of the original system.
Lancer	Competition between BAe and GEC before the merger, when the two companies were forced to maintain the boundary split and isolation in order to maintain the competition environment. The systems involved were Lancer (this case) and SIKA. The programmes produced two prototype land reconnaissance vehicles. The two systems would compete for the FRES and FCS programme options for the reconnaissance vehicle.	LA01	The mindset of the organisation.	The proximity of the business units, the lack of a clear defined contractual link between the two, the level of communication that could be achieved between the two.	N/A	N/A	Mindset of personnel.
		LA02	Separate goals and targets.	Individual organisations have individual targets and they are not always compatible or even that best suit the business.	N/A	N/A	Lack of common direction.
		LA03	N/A	N/A	The development of the system control feature did not incorporate a safety system to prevent this failure.	Lack of understanding of the use of the system by the developers?	Lack of understanding of the use of the system by the developers?
		LA04	N/A	N/A	N/A	N/A	I am not sure this is even a complexity issue.
		LA05	N/A	N/A	N/A	N/A	N/A
		LA06	N/A	N/A	N/A	N/A	Budget, requirement to integrate quickly.

Programme Case Study	Brief	Problem Number	Origins of Problem				
			Organisational Characteristics	Organisational Configuration Inducers	Internal Factors	External Factors	Origins of Complexity
Marksman	An anti-aircraft tank turret, with two guns and high performance swivel turret mounting, containing a new radar concept and predictor system. The product was highly successful in testing, but was not easily sold due to constraints on budget, and a lack of a capability requirement for it.	MK01	N/A	N/A	N/A	N/A	The lack of the pre-production phase. The requirement to keep costs to a minimum as the product would not go into full production.
		MK02	N/A	N/A	N/A	N/A	The lack of the pre-production phase. The requirement to keep costs to a minimum as the product would not go into full production.
Astute	Background – Initially the Astute programme was contracted to GEC, after the merger between GEC and BAe, the contract was for a complete platform, the first of contract that actually procured the whole system from a single company. The contract initially was set out as a procurement exercise and was handled by the Procurement Contacts Office (PCO). The programme estimated cost was in the order of £3bn and has been running for a total of 9 years to date (02/05/05). The submarine is essentially a replacement for the Swiftshore class submarine, and is intended to be a stealthy smaller boat. The original requirement was set against the cold war	AT01	Mindset, structure, upper management	The Procurement Contracts Office being the prime contractor for the project means that the wrong mindset was instilled from the start.	N/A	N/A	Erroneous requirements set. Initial prime contracting structure of the organisation. Requirements not able to provide testing and acceptance criteria for the product. Attempt to classify all the requirements exactly rather than leaving them capability based.
		AT02	N/A	N/A	N/A	N/A	Lack of appropriate requirements, vague requirements.
		AT03	N/A	N/A	N/A	N//A	Lack of appropriate requirements, vague requirements.

Programme Case Study	Brief	Problem Number	Origins of Problem				
			Organisational Characteristics	Organisational Configuration Inducers	Internal Factors	External Factors	Origins of Complexity
	threat, which has obviously is no longer valid as a requirement at this point in time.	AT04	Culture, strategy, success measurement, mindset.	The Procurement Contracts Office being the prime contractor for the project means that the wrong mindset was instilled from the start.	N/A	N/A	Organisational culture and the structure of the prime contract team being incompatible with the nature of the programme. Procurement strategy rather than design strategy.
		AT05	There are two organisations involved here, the problem was their culture, approach, working practices and of course the geographical location.	The physical distance, and working practices of each organisation. The distinction of the staff, one all BAE, the other all VSEL.	N/A	N/A	Organisational culture, VSEL past experience and culture, division of process and procedure, geographical separation of the manufacture and procurement office site.
		AT06	N/A	All contractors within the programme require services and make demands on the services that are available or request additional services, these then must be catered for. These service demands change and evolve as the product matures in development.	Standards, Specifications, Service Requirements, Number of Services, Level of Interactions, Space Constraints.	Changing service requirements and constraints as the design matures.	Level of interactions, changing service requirements, large number of constraints in the forms of standards, links, etc. See the internal factors and complexity concepts.
Test Bed	Technical demonstrator programme for the MOD, and was a 50% split between industry and the DPA. It consisted of BAE SYSTEMS and a number of other companies. Due to the politics of the time, mainly that Warton was massively overdue with its contract, it was important that this contract was done with the up most to maintain the timescales and complete the programme on time. The contract took 1 year to get agreed,	TB01	Changing nature of the organisation, in terms of personnel, goals, targets.	Organisational configuration inducers are the key reason for this problem issue. The configuration constantly changes and adapts to market pressures and company strategy. These changes force unwanted changes on the development team and their resources, and in some cases process.	N/A	N/A	Changing organisational configurations.
		TB02	N/A	N/A	N/A	N/A	Not convinced that this is even a complexity issue?

Programme Case Study	Brief	Problem Number	Origins of Problem				
			Organisational Characteristics	Organisational Configuration Inducers	Internal Factors	External Factors	Origins of Complexity
	and a further 5 to complete making the project running time 6 years in total.	TB03	N/A	N/A	Interfaces between FSS materials and non FSS materials.	N/A	Need to maintain the core requirements.
		TB04	N/A	N/A	The massive number of variables and the diverse nature of these variables and the requirement to ensure that these variables are optimised for the design.	N/A	Large number of variables, of different types, and requirements requiring an optimisation of these variables.
Information Capability Programme (NEC)	ICP is the integration of all the BAE SYSTEM component businesses, it's essentially the development of capability that the customer has not yet realised. NEC or Network Enabled Capability as it was originally coined, is the realisation of information transfer between different elements of the battle space. The image often given by BAE SYSTEMS shows UAVs, aircraft, soldiers, ground vehicles, ships, submarines all communicating with each other and exchanging information across the battle space. This is the realisation of what is ICP.	IP01	Interfaces and boundaries, organisational change, co-evolving strategies, personal and professional relationships, being able to provide a flexible response.	The whole nature of the problem is the configurational change of the organisation that the programme has to operate within.	N/A	The development of the NEC capability requirement. MoD Policy and BAE SYSTEMS response, competitors and partners response.	The corporate response to perceived need. Continual corporate re-organisation.
		IP02	N/A	N/A	N/A	N/A	I am not sure about this.
		IP03	Be agile to change or purchase companies in order to acquire the needed technologies.	N/A	N/A	N/A	I am not sure about this.
		IP04	N/A	N/A	N/A	Human interaction and interpretation of the information that the system provides. The information, unless a clear doctrine (which BAE SYSTEMS does not provide), will have large effects on system operation.	Interfaces between elements where they did not previously exist. The changing nature of the system as a result of the introduction of ICP information exchanges.

Table 148 - Table of the first case study set, showing the relevant origins of the complexity for each case

14.3 Table 149 - Part 3 of the Complexity Matrix

Programme Case Study	Brief	Problem Number	Problem Issues	Complexity Definitions	Approach to Problem	Coping Measures Employed	Concepts / Classifications	Complexity Measures
Upgrade Programme	An upgrade of an existing legacy system, this upgrade required the contractor to maintain the functionality of the current system. Work was initiated with a large amount of reverse requirements engineering.	UP01	Organisational problems, Contracts, Work Blocking, Slow Working, Over Familiarity.	This does not really fit with the complexity definitions.	The problem was addressed by forming a joint project team and joint management team as a subset, so that organisationally they knew where they were.	The measures were the changing of the organisational structure from a contract based approach to a work package.	Soft and intangible, lack of definition, invisible. Dynamic complexity	N/A
		UP02	Requirement maturity issue. Timescales, the requirement to get work underway despite the lack of concrete requirements information.	Related to future prediction based on current information. There is not really a complexity definition that fits this type of problem, no complexity definitions I have found so far have relationships with requirement maturity.	There was no set coping mechanism, the work resulted in large amounts of re-work, but due to management control has now begun to settle into a more manageable state. The coping mechanism employed was the re-work. Evolutionary integration mechanisms could be used here.	Re-work was the coping measure.	Uncertainty of the nature of the legacy system, complex black box, lack of transparency. Dynamic complexity	N/A
		UP03	A requirements capture issue, resulting from the need for a definition of design, the requirements replicate a lot of information that is not required by the engineering process.	Related to requirements derivation, unsure of the link to the complexity definitions. Perhaps a term requirements complexity.	In a sense the coping mechanism here was partly to blame for the increase in the amount of re-work required for the system. The reverse engineering exercise meant that every aspect of the software design was replicated, even though some aspects of the design were platform specific. Reverse engineer everything and after that re-integrate the new requirements.	Working with the immature requirements as they came in and adapting as these requirements changed during the merge of new system requirements and the original system reverse engineering of the functional requirements.	Complex black box, lack of transparency. Dynamic complexity could be used to describe this effect as the relationships between the requirements were non linear, they were connected laterally as the requirements package was created. Two parallel tasks essentially.	N/A

		UP04	Engineering process and governance problem (hindrance). Organisational responsibility problem, time scale problem, work progression problem.	This could potentially be a hierarchy complexity issue, one of the basic complexity concepts, but there is no definition that I have found so far that really matches with this style of problem.	Originally the design and certification was carried out within BU1, however with the change in the structure of the organisational relationships within the company the design certification and approval was carried out within the BU2 for their aspects of the design for the product. This speeded up the development procedures.	Shift the acceptance and design certification responsibility to the business unit developing the system components to accelerate the development.	Over complicated inefficient process, that required simplification. The concept that could apply here is perhaps detail complexity, a hierarchical structure, the problem exists in a relationship within that structure.	N/A
		UP05	Cultural	None that I can think of apply to this problem.	The formation of a joint project team to change the mindset of the primary organisation. A project trouble-shooter was appointed. The mindset of the organisation had to change in order to allow progression to be more efficient. The level of design required may have shifted the mindset of those working on the programme. There is no real coping mechanism for this problem.	Change the mindset of those personnel working on the programme as development continues.	Soft and intangible, indefinable, conflict of perceptions, lack of understanding of the reality of the problem.	N/A
Lancer	Competition between BAe and GEC before the merger, when the two companies were forced to maintain the boundary split and isolation in order to maintain the competition environment.	LA01	Culture and over familiarity between business units. The lack of a formal contractual link between business units causing specifications to be neglected and increase workloads later on in development.	None that I can think of apply to this problem.	To my knowledge this was never addressed specifically, but the programme did complete and the product was successful.	Continue working with the over familiarity between the two business units. Cope with the effects after they occur.	?	N/A

	The systems involved were Lancer (this case) and SIKA. The programmes produced two prototype land reconnaissance vehicles. The two systems would compete for the FRES and FCS programme options for the Reconnaissance vehicle.	LA02	A generic problem across all companies, the problems are lack of common direction, usually a reduction in the overall profitability of work, no optimisation for business units for their line of work.	Perhaps a complexity definition suits this as it is a set of organisational entities that are exhibiting independent behaviour while interacting and the effect on the whole may not be obvious?	There was no coping mechanism introduced to deal with this issue.	I am not sure how this is coped with? Or even if the business units should cope with it as such? Perhaps this is the whole nature of the problem, that due to its nature it is not coped with anywhere within the development lifecycle.	Independent elements with independent goals expected to function together as one, but the behaviour not quite understood.	N/A
		LA03	Engine failure during testing resulted from misuse.	Fault finding, cause and effect difficult to ascertain perhaps.	Fault finding was the coping mechanism that was employed here, it was not an obvious problem, later it came to light that the handbrake was being left on while the vehicle moved as there were no indications within the vehicle it was applied. Thus the vehicle was moving with the brake applied an overloading the control system.	N/A	Cause and effect difficult to ascertain.	N/A
		LA04	Difficult to ensure safe raising and lowering of the sensor suite.	I can't think of any that apply?	An external contractor was employed to re-visit the problem and a device was developed that ensured the coiling of the cabling was done as the sensor pack was retracted.	Employ external contractor to correct the problem.	I can't think of any?	N/A

		LA05	The lack of understanding of the GPS system mapping, and it's incompatibility with conventional mapping producing a vehicle location that is not accurate.	Not sure I can think of an application of a complex definition.	There are standards that are employed to compensate for this effect WGS84 is one of these standards. The problem was solved by applying this standard to all the mapping systems of the vehicle and obtaining 2d surface maps that were also WGS84 compliant. This solved the problem and the discrepancy was resolved.	Rework the problem and repair it after.	Reduced understanding of the elements is seen as ignorance, which is not an excuse for complexity.	N/A
		LA06	Not following the set design processes in an effort to accelerate the development of the technology demonstrator.	I'm not convinced there are any suitable definitions for this.	The CSIL only employed to do the integration process to save time and money to accelerate the programme.	Cope with using the CSIL only as the integration platform for the whole system. The vehicle was then integrated and during testing was repaired while actually on trials.	N/A	N/A
Marksman	An anti-aircraft tank turret, with two guns and high performance swivel turret mounting, containing a new radar concept and predictor system. The product was highly successful in testing, but was not easily sold due to constraints on budget, and a lack of a capability requirement for it.	MK01	Reduction in the understanding of the product build. The drawing packs were not kept up to date and subsequently the changes to the drawings were not integrated into the complete drawing pack set. Slows down the building process, adds a lot of unnecessary re-work.	There is not really a clear complexity definition that fits with this problem issue.	No coping mechanisms were employed for this issue to tackle it, this was not viewed as a problem as such, because the sales for the system were small. It was decided that only the 6 systems would be built for the Fins, and the company would cope with the difficulties for these six and then produce no more of these systems.	Reduce the level of commitment to the support programme for the product.	Reduced understanding of the product build phases. Induced complexity during the manufacture phase and also at later stages in the product life.	

		MK02	No commonality, no manufacturing processes defined for the product, difficult product support and maintainability.	There is not really a clear complexity definition that fits with this problem issue.	There were no coping mechanisms really introduced to counter this problem. The company simply reduced the costs associated with the manufacture stages by removing the pre-production phase from the programme, after this they coped by supporting uncommon systems and only providing a set amount of support.	Reduce the level of commitment to the support programme for the product.	Lack of commonality.	

Astute	Background – Initially the Astute programme was contracted to GEC, after the merger between GEC and BAe, the contract was for a complete platform, the first of contract that actually procured the whole system from a single company. The contract initially was set out as a procurement exercise and was handled by the Procurement Contacts Office (PCO). The programme estimated cost was in the order of £3bn and has been running for a total of 9 years to date (02/05/05). The submarine is essentially a replacement for the Swiftshore class submarine, and is intended to be a stealthy smaller boat. The original requirement was set against the cold war threat, which has obviously is no longer valid as a requirement at this point in time.	AT01	Erroneous requirements, incomplete requirements, inappropriate requirements, wrong mindset of project leaders (Procurement not Design).	There are no direct definitions that seem to correspond with this type of problem, it is a lack of common direction and understanding of the nature of the capability required in the initial stages. I cannot see a map to a complexity definition at this stage.	Initially for the first part of the programme (3 years) there was no coping mechanism employed. Progress was measured against the number of procurement contracts placed with sub-contractors (since this was a procurement exercise) . The problem was later addressed when a team was brought in to evaluate the requirement set. A new requirements document (1000 requirements) was produced geared towards a capability procurement rather than specific product procurement, and this requirement set was mapped to the original 10000. So the coping mechanism here is to employ external help in the form of consultants.	The coping measure is to seek external help and reduce the set of requirements to a more manageable amount and also change the nature of the requirements so that they more accurately reflect the design and build nature of the programme, steering away from the procurement idea.	Over complicated requirement set, process inappropriate for the task, changing and evolving requirements, capability based requirements rather than fixed.	Measure the number of requirements.

		AT02	The complete inability to satisfy the requirement. There is no definition of what is fit for purpose and as a result the requirement cannot be satisfied.		There had to be a definition of these requirements in order for the system to be accepted. The purpose of the product may change as time progresses, and as threats change and evolve, so the fit for purpose definition was important if not vital to the success of proving compliance to that requirement. The programme had been running 3 years before a definition of fit for purpose had been established. The requirements evolved out of the original requirements set to the 1000 requirement set and from this a "concept of operations" (CONOPS) was produced for the product, forming the basis for the acceptance criteria of the fit for purpose requirement.	The measure was to define fit for purpose with respect to the new requirement set and use this to form the acceptance criteria.	Not sure what goes here for this?	N/A

		AT03	A requirement that is not defined and cannot be tested against in order to accept the product.		Better than existing platforms is a very vague requirement, to be better than the existing platforms, all that is required technically is an improvement in just one area, perhaps just 1 knot faster for example, but this is obviously not what the customer has in mind and therefore the CONOPS must encompass the areas where the platform must improve on previous platforms. The CONOPS provided the acceptance criteria for this requirement also. Also not mentioned before, is that the CONOPS also provided the basis for the testing of the requirements, by allowing the definition of scenarios and exercises for the platform.	Related to the previous problem, the better than the previous platform requirement is also not defined, and the concept of operation provided the basis from which to define and test to this requirement.	Not sure what goes here for this?	N/A
		AT04	Measurement of success was against the number of procurement contracts that were placed, the fact that this was really a design exercise for a completely new boat was dismissed, and a more enlightened approach to the engineering would have rectified this.		The mindset was forced to change as the requirements sets changed. There was a realisation that this was no longer a procurement exercise and actually a full blown design activity later in the programme. In a sense the introduction of the contractors to deal with the requirements issue, inadvertently provided the mechanism to cope with this problem as well.	The ideal coping measure would be to remove control from the PCO and transfer it into a full blown design programme, this was not done in itself.	Not sure what goes here for this?	N/A

		AT05	VSEL starting to use their own processes rather than the ones devised by the prime contractor.		Not entirely sure what was done here, I think they moved a large number of BAE staff to the VSEL site in an attempt to keep control of the programme, this could be the coping mechanism that they employed here.	Move BAE staff to the VESL site in order to maintain control and reduce the influence of VSEL culture on the development process.	Individual elements of the system functioning independently and the overall effect not known or fully appreciated.	N/A
		AT06	Space within the boat, the routing of large number of services that require routing, the standards that must be maintained while routing.	This could be defined as a system where the component parts are understood in terms of their properties and the number of them, and their interactions. But the solution is not obvious due to the number and level of interaction caused.	Unsure exactly how the coping mechanism for this operates, but there must be a detailed one. If the services change within the submarine the effect on the routing could be massive. The level of interaction between the service number, the constraints of the routing, and the 3d space constraint makes this problem difficult to cope with. This is not just limited to submarines, but on any system where there are a large number of services of different types that exhibit interactions that are not always obvious, and are controlled by constraints such as standards.	I am not sure what the coping method is here, perhaps an expensive MDDO process?	Large number of interactions, understanding of the component parts, large number of elements to consider, unable to be understood by any one person.	Interactions, number of services, number of different types of interactions, space constraint.

Test Bed	Technical demonstrator programme for the MOD, and was a 50% split between industry and the DPA. It consisted of BAE SYSTEMS and a number of other companies. Due to the politics of the time, mainly that Warton was massively overdue with its contract, it was important that this contract was done with the up most to maintain the timescales and complete the programme on time. The contract took 1 year to get agreed, and a further 5 to complete making the project running time 6 years in total.	TB01	Changing company landscapes, evolving supplier base, and contractual agreements, changing personnel, shifts of knowledge.	Not sure what complexity definitions suite this well.	When the project started it suffered from the usual commercial landscape changes. During the merger between GEC and BAe, there was little effect to the team of individuals working on the programme, it maintained the same management and the same key technical staff. The programme benefited from the fact one person remained on the programme throughout the whole lifecycle.	A company is forced to cope with these types of changes, they must adapt to the changing nature of the business and the changing market. In this project some aspects of these changes were mitigated against (although I am not convinced this was an intended coping mechanism) by maintaining tacit knowledge of the product by keeping key personnel.	Unstable environments, constant requirement to adapt and evolve.	N/A
		TB02	The lack of understanding of the DPA head of the programme, and the risk that the requirements could change massively.	Not sure which complexity definitions could apply here if any.	There was an education like process that brought the current head of the DPA up to speed regarding the project outcomes and deliverables, while also including an education of the nature of the product that was being developed and the technologies that were involved.	Educate the DPA head early after their appointment in an effort to keep them in touch with the work that is being carried out in order to minimise the impact of the change.	N/A	N/A
		TB03	The difficulty of interfacing these materials while maintaining the overall requirement.	Not sure about this, potentially the reduced understanding of the interactions of the materials makes the development of the FSS interfaces difficult.	The manufacturing of the materials was sub-contracted out to a company that specialised in the generation of FSS materials, and special connections were developed to make the connections between this material and the standard hull seamless to reduce the radar signatures.	With an industrial partner develop a interface design between the materials that maintains the materials desired qualities.	Interactions and lack of understanding of those interactions, and the need to juggle large numbers of variables, also the requirement for accuracy, but again, I am not convinced that these are really complexity concepts.	N/A

		TB04	The massive scope of the MDDO exercise, how to choose the optimal design.	Large number of interactions and the lack of understanding of the overall problem.	Risk management and QFDs were used to conceptualise and assess different options. Rather than trying to achieve the perfect solution and optimisation for the parameters (of which there were a very large amount) concepts were generated by skilled engineers and these concepts explored using risk generation and QFD diagrams (including weighing generations and scoring for different concepts against the chosen system attributes of importance).	Use of QFDs to develop a set of design concepts. These concepts are then factored within a QFD and the most appropriate design chosen based on weightings. The MDDO exercise is reduced by the development of the concepts, rather than a complete MDDO exercise the problem scope is constrained using engineering knowledge and experience.	Interactions, large number of elements, some elements indefinable, elements of different types that are not comparable, reduced understanding of the operation of the whole, or the effect on the whole of changing a single variable.	Maybe some mathematical measures for the mathematical complexity elements in this MDDO exercise?
Information Capability Programme (NEC)	ICP is the integration of all the BAE SYSTEM component businesses, it's essentially the development of capability that the customer has not yet realised. NEC or Network Enabled Capability as it was originally coined, is the realisation of information transfer between different elements of the battle space. The image often given by BAE SYSTEMS shows UAVs, aircraft, soldiers, ground vehicles, ships, submarines all communicating with each other and exchanging information across the battle space. This is the	IP01	Organisational changes have a big effect on this programme. The programme operates as a linkage between the business units developing the network enabled capability between them, any changes within the organisation will affect how it operates.	Constant evolving nature of the environment could be connected to a complexity definition.	ICP deals with the programme it runs rather than dealing with the organisational changes. It has no identifiable (stated) coping mechanism for dealing with changing relationships.	Try to remain agile enough to be able to absorb the changes in the organisational structure.	Unpredictable and uncertain functionality requirements. Complex Adaptive and Dynamic.	N/A
		IP02	Predicting the customer requirements before they actually exist and making these predictions accurate.	Not sure which complexity definitions could apply here if any.	Customer needs are anticipated before they are realised by the customer themselves in an effort to predict the outcome of future work. ICP does not offer its own product but offers its service through other programmes within BAE.	Bringing the customer round to the ICP mindset?	N/A	N/A

	realisation of what is ICP.	IP03	Ensure that technologies are ready on time for the ICP implementations.	Not sure which complexity definitions could apply here if any.	The business unit recommends which technologies are required by which perceived requirement. These technologies are then either purchased or invested in so that they mature at the time the perceived requirement comes to light.	Invest in technology development, take over companies that specialise in the technologies that will or are required to ensure they reach the required maturity level.	Unpredictable and uncertain technology readiness levels.	There are TRL measures I am sure, and a whole strand of BAE SYSTEMS maturity modelling.
		IP04	The problem issue is the lack of ability of prediction of the overall system behaviour despite a detailed knowledge of the component parts and the schedule of the introduction of these component parts. The other aspect of the design problem, is that there is an introduction of information links between elements of a system and these links have not existed before. There is no way of predicting the nature of the system change. The system will be unpredictable when the information link is established.	Relates to the definitions that support the idea that despite the knowledge of the component parts the overall system behaviour does not mean that the overall behaviour of the system while in operation can be understood. The design manager in this case, seems to not subscribe to the concept of the emergence of behaviour despite a detailed understanding of the components. This subscribes to most of the complexity definitions.	The BMEC is a large simulation system that is capable of modelling the ICP product and the affect of the information exchange that the system offers. This is used to understand the behaviour of the product when it is in operation, but this is of a much smaller scale and is not representative of the true battle space environment.	The system introduces connections in an incremental manner. Aspects of the ICP capability are introduced over time, this incremental method has its merits as the problems that occur as the increment is added can be repaired before the addition of the next increment. Although the method will tackle the complexity issue, it is perhaps not the most efficient method at doing so.	Complex Adaptive Systems, Evolution of systems, incremental design.	Unsure of which measures can be used here.

Table 149 – Shows the various different complexity characteristics within each problem.

15 Appendix B – Problem Description Tables

The following data, shows the relationships between the problem issues analysed in the first set of case studies with the solutions applied to them.

15.1 Astute

Problem Solutions		Problem Causes: Programme Issues			Organisational					Human	Requirements		System Design			Testing				Development Process					Manufacture Process		Support	
		Timescales	Budget	Size and Scope	Current Structure	Structure Change	Governance	Sub-Contracts	Customer	Mindset	Definition	Maturity	System Internal Interfaces	System External Interfaces	Legacy Equipment	Planning	Trials	Acceptance	Certification	Concurrent	Serial	Deliverables	Rework	Parellel	Drawing	Pre-Production	Maintain	Reliability
Programme Issues	Timescales										AT01	AT01																
	Budget																											
	Size and Scope										AT01	AT01																
Organisational	Current Structure									AT04,AT05																		
	Structure Change				AT06																							
	Governance					AT05		AT04,AT05		AT04,AT05										AT05								
	Sub-Contracts					AT05	AT02,AT03			AT04,AT05			AT06	AT06	AT06													
	Customer						AT02,AT03			AT04																		
Human	Mindset					AT05	AT01																					
Requirements	Definition									AT01						AT02,AT03	AT02,AT03	AT01,AT02,A	AT01									
	Maturity										AT01					AT02,AT03	AT02,AT03	AT01,AT02,A	AT01									
System Design	System Internal interfaces									AT01									AT06									
	System External interfaces									AT01									AT06									
	Legacy Equipment																		AT06									
Testing	Planning																											
	Trials																											
	Acceptance										AT02,AT03																	
	Certification																											
Development Process	Concurrent					AT05	AT05																					
	Serial																											
	Deliverables					AT01	AT01			AT01	AT01																	
	Rework																											
	Parellel																											
Manufacture Process	Drawing																											
	Pre-Production																											
Support	Maintain																											
	Reliability																											

Table 150 - The table of the relationships of the Astute problems and the product lifecycle.

15.2 Test Bed

Problem Solutions		Problem Causes: Programme Issues			Organisational					Human	Requirements		System Design			Testing				Development Process					Manufacture Process		Support	
		Timescales	Budget	Size and Scope	Current Structure	Structure Change	Governance	Sub-Contracts	Customer	Mindset	Definition	Maturity	System Internal Interfaces	System External Interfaces	Legacy Equipment	Planning	Trials	Acceptance	Certification	Concurrent	Serial	Deliverables	Rework	Parelel	Drawing	Pre-Production	Maintain	Reliability
Programme Issues	Timescales					TB01			TB02	TB01	TB02							TB04										
	Budget					TB01			TB02									TB04										
	Size and Scope																											
Organisational	Current Structure																											
	Structure Change				TB01,TB02					TB01																		
	Governance																											
	Sub-Contracts																											
	Customer									TB02								TB02										
Human	Mindset				TB02																							
Requirements	Definition												TB04	TB04				TB04										
	Maturity																											
System Design	System Internal Interfaces	TB04						TB03			TB04							TB03,TB04								TB03		
	System External Interfaces	TB04						TB03			TB04							TB04										
	Legacy Equipment																											
Testing	Planning																											
	Trials																											
	Acceptance																											
	Certification																											
Development Process	Concurrent	TB04																										
	Serial																											
	Deliverables																											
	Rework																											
	Parelel																											
Manufacture Process	Drawing																											
	Pre-Production																											
Support	Maintain																											
	Reliability																											

Table 151 - The table of the relationships of the Test Bed problems and the product lifecycle.

15.3 Integrated Capability Programme

Problem Solutions		Problem Causes: Programme Issues			Organisational					Human	Requirements		System Design			Testing				Development Process					Manufacture Process		Support	
		Timescales	Budget	Size and Scope	Current Structure	Structure Change	Governance	Sub-Contracts	Customer	Mindset	Definition	Maturity	System Internal Interfaces	System External Interfaces	Legacy Equipment	Planning	Trials	Acceptance	Certification	Concurrent	Serial	Deliverables	Rework	Parallel	Drawing	Pre-Production	Maintain	Reliability
Programme Issues	Timescales			IP01		IP01																						
	Budget					IP01																						
	Size and Scope																											
Organisational	Current Structure			IP03																								
	Structure Change			IP03																IP01		IP01						IP04
	Governance																											
	Sub-Contracts																											
	Customer									IP02	IP02	IP02																
Human	Mindset										IP02																	
Requirements	Definition																											
	Maturity																											
System Design	System Internal Interfaces																			IP03				IP03				IP04
	System External Interfaces																			IP03				IP03				IP04
	Legacy Equipment																											IP04
Testing	Planning																											
	Trials																											
	Acceptance																											
	Certification																											
Development Process	Concurrent																											
	Serial																											
	Deliverables																											
	Rework																											
	Parallel																											
Manufacture Process	Drawing																											
	Pre-Production																											
Support	Maintain																			IP04								
	Reliability																			IP04								

Table 152 - The table of the relationships of the Integrated Capability Programme problems and the product lifecycle.

15.4 Upgrade Programme

Problem Solutions		Problem Causes: Programme Issues			Organisational					Human	Requirements		System Design			Testing				Development Process					Manufacture Process		Support	
		Timescales	Budget	Size and Scope	Current Structure	Structure Change	Governance	Sub-Contracts	Customer	Mindset	Definition	Maturity	System Internal Interfaces	System External Interfaces	Legacy Equipment	Planning	Trials	Acceptance	Certification	Concurrent	Serial	Deliverables	Rework	Parallel	Drawing	Pre-Production	Maintain	Reliability
Programme Issues	Timescales																											
	Budget																											
	Size and Scope																											
Organisational	Current Structure																											
	Structure Change																											
	Governance																											
	Sub-Contracts																											
	Customer																											
Human	Mindset																											
Requirements	Definition																											
	Maturity																											
System Design	System Internal Interfaces																											
	System External Interfaces																											
	Legacy Equipment																											
Testing	Planning																											
	Trials																											
	Acceptance																											
	Certification																											
Development Process	Concurrent																											
	Serial																											
	Deliverables																											
	Rework																											
	Parallel																											
Manufacture Process	Drawing																											
	Pre-Production																											
Support	Maintain																											
	Reliability																											

Table 153 - The table of the relationships of the Upgrade Programme problems and the product lifecycle.

15.5 Marksman

Problem Causes / Problem Solutions

Problem Solutions		Programme Issues			Organisational					Human	Requirements		System Design			Testing				Development Process					Manufacture Process		Support	
		Timescales	Budget	Size and Scope	Current Structure	Structure Change	Governance	Sub-Contracts	Customer	Mindset	Definition	Maturity	System Internal Interfaces	System External Interfaces	Legacy Equipment	Planning	Trials	Acceptance	Certification	Concurrent	Serial	Deliverables	Rework	Parallel	Drawing	Pre-Production	Maintain	Reliability
Programme Issues	Timescales		MK01	MK01																					MK01	MK02		
	Budget																								MK01, MK02			
	Size and Scope																								MK01, MK02			
Organisational	Current Structure																											
	Structure Change																											
	Governance																											
	Sub-Contracts																											
	Customer																											
Human	Mindset																											
Requirements	Definition																											
	Maturity																											
System Design	System Internal Interfaces																											
	System External Interfaces																											
	Legacy Equipment																											
Testing	Planning																											
	Trials																											
	Acceptance																											
	Certification																											
Development Process	Concurrent																											
	Serial																											
	Deliverables																											
	Rework																											
	Parallel																											
Manufacture Process	Drawing																											
	Pre-Production																											
Support	Maintain																											
	Reliability																											

Table 154 - The table of the relationships of the Marksman problems and the product lifecycle.

15.6 Lancer

Problem Solutions		Problem Causes: Programme issues – Timescales	Budget	Size and Scope	Organisational – Current Structure	Structure Change	Governance	Sub-Contracts	Customer	Human – Mindset	Requirements – Definition	Maturity	System Design – System Internal Interfaces	System External Interfaces	Legacy Equipment	Testing – Planning	Trials	Acceptance	Certification	Development Process – Concurrent	Serial	Deliverables	Rework	Parallel	Manufacture Process – Drawing	Pre-Production	Support – Maintain	Reliability
Programme issues	Timescales				LA01		LA02	LA01		LA02														LA06					
	Budget						LA02			LA02														LA06					
	Size and Scope																											
Organisational	Current Structure							LA01		LA01																		
	Structure Change																		LA01				LA01					
	Governance																											
	Sub-Contracts																											
	Customer																											
Human	Mindset						LA06																					
Requirements	Definition																											
	Maturity																											
System Design	System Internal Interfaces	LA06	LA06					LA04		LA03				LA05		LA03	LA03, LA04	LA03, LA04		LA06			LA06					LA04, LA05
	System External Interfaces	LA06	LA06																	LA06			LA06					
	Legacy Equipment																											
Testing	Planning																		LA03									
	Trials																											
	Acceptance																											
	Certification																											
Development Process	Concurrent																											
	Serial																											
	Deliverables																											
	Rework													LA05														
	Parallel																											
Manufacture Process	Drawing																											
	Pre-Production																											
Support	Maintain																											
	Reliability																											

Table 155 - The table of the relationships of the Lancer problems and the product lifecycle.

16 Appendix C – The Complexity Measurement Tool

This appendix consists of the various complexity measurement tools used to collect the data above. Within this appendix find those tools used for the Fuel Rig, NCS, and NST. All the tools are found within this parent directory:

\\Submission DVD\Analysis Tools

This directory is then split into three subsections and each of these subsections contains the measurement tools with the data included:

\\Submission DVD\Analysis Tools\Fuel Rig Results\

Analysis Tool Fuel 260806.xls

\\Submission DVD\Analysis Tools\NCS Results\

Analysis Tool NCS 160807.xls

\\Submission DVD\Analysis Tools\NST Results\

Analysis Tool NST 160807 Option 1WS.xls

Analysis Tool NST 160807 Option 2WS.xls

Analysis Tool NST 160807 Option 3WS.xls

Analysis Tool NST 160807 Option 4WS.xls

Analysis Tool NST 160807 Option 5WS.xls

Analysis Tool NST 160807 Option 6WS.xls

17 Appendix D – The Complexity Component and Characteristics Store

This appendix is the Complexity Component and Characteristic Store (CCCS), it is contained within the accompanying DVD in the following location.

\\Submission DVD\ Complexity Component and Characteristic Store (CCCS)\

Complexity Component and Characteristic Store.xls