Abstract

This thesis addresses certain problems encountered by teams of engineers when modelling complex structures and processes subject to cost and other resource constraints. The cost of a structure or process may be ‘read off’ its specifying model, but the language in which the model is expressed (e.g. CAD) and the language in which resources may be modelled (e.g. spreadsheets) are not naturally compatible. This thesis demonstrates that a number of intermediate steps may be introduced which enable both meaningful translation from one conceptual view to another as well as meaningful collaboration between team members. The work adopts a diagrammatic modelling approach as a natural one in an engineering context when seeking to establish a shared understanding of problems.

Thus, the research question to be answered in this thesis is: ‘To what extent is it possible to improve user-driven software development through interaction with diagrams and without requiring users to learn particular computer languages?’ The goal of the research is to improve collaborative software development through interaction with diagrams, thereby minimising the need for end-users to code directly. To achieve this aim a combination of the paradigms of End-User Programming, Process and Product Modelling and Decision Support, and Semantic Web are exploited and a methodology of User Driven Modelling and Programming (UDM/P) is developed, implemented, and tested as a means of demonstrating the efficacy of diagrammatic modelling.

In greater detail, the research seeks to show that diagrammatic modelling eases problems of maintenance, extensibility, ease of use, and sharing of information. The methodology presented here to achieve this involves a three step translation from a visualised ontology, through a modelling tool, to output to interactive visualisations. An analysis of users groups them into categories of system creator, model builder, and model user. This categorisation corresponds well with the three-step translation process where users develop the ontology, modelling tool, and visualisations for their problem.

This research establishes and exemplifies a novel paradigm of collaborative end-user programming by domain experts. The end-user programmers can use a visual interface where the visualisation of the software exactly matches the structure of the software itself, making translation between user and computer, and vice versa, much more direct and practical. The visualisation is based on an ontology that provides a representation of the software as a tree. The solution is based on translation from a source tree to a result tree, and visualisation of both. The result tree shows a structured representation of the model with a full visualisation of all parts that leads to the computed result.

In conclusion, it is claimed that this direct representation of the structure enables an understanding of the program as an ontology and model that is then visualised, resulting in a more transparent shared understanding by all users. It is further argued that our diagrammatic modelling paradigm consequently eases problems of maintenance, extensibility, ease of use, and sharing of information. This method is applicable to any problem that lends itself to representation as a tree. This is considered a limitation of the method to be addressed in a future project.

Contents

Figures

Tables

Chapter 1 Introduction

1.1 Explanation of the Problem

Diagrammatic modelling plays a major role in current practice in the design of software, and in the design of engineering components. Diagrams also play a role in Business Modelling, and Scientific Modelling and Visualisation. The problem to be investigated is how to use such tools within engineering to create or enable a shared understanding, and what is common in the use of these tools for engineering modelling to other types of modelling.

This research involves enabling computer literate people who are not programmers to create software. The means for this is construction of visual diagrams that represent the task, in a similar way to how family trees and taxonomies can be represented as diagrams. The research involves translating from an ontology to modelling system code and meta-code, and also to a visual interface produced by automated translation, and demonstrated on a website (Hale, 2011). The aim is to enable programming without requiring people to learn computer languages. Of particular interest is making it possible for a wider range of people to collaborate on development of computer models. The main objective of this research is to create a modelling system that can be edited by computer literate non-programmers, and so demonstrate an application of end-user programming that could be used in a generic way. The methodology for this is automation by translation from a human level of abstraction to software. This is explained in chapter 3.

The aspiration behind this thesis is that it is possible to create a systematic diagrammatic representation of logical thought to enable representation of user's wishes to a computer. C.S. Peirce (1906) proclaimed in 'Prolegomena to an Apology for Pragmaticism', "Come on, my Reader, and let us construct a diagram to illustrate the general course of thought; I mean a system of diagrammatization by means of which any course of thought can be represented with exactitude". Perhaps use of new technologies can solve this old problem. That is the purpose of this research, but to limit the scope and so make application of this theory testable, the research is restricted mainly to engineers (because they often think in terms of diagrams) and to the domain of modelling (which often requires diagrams). Others can expand it for different domains and users. So the aim is to apply the research first where it can have the most immediate use and encourage others to expand it for other domains and other users. This research is intended to simplify computing for computer literate non-programmers, this includes many engineers. The main research area is enabling users such as engineers to model the problems they encounter in manufacturing and design. However, the wider aim is to prototype research for enabling a larger range of software users to model their problems. The intention is to create collaborative tools that allow users to develop software in a way they will be familiar with from their use of spreadsheets.

The type of problem modelled by engineers using spreadsheets is usually dependent on a product data structure (bill of materials), component structure, and a structure of processes. These can all be represented as a linked structure of taxonomy trees. Such a structure is relatively simple to represent diagrammatically, and also programmatically as it is recursive; so this is the application for the research. A further advantage is that taxonomy representations are common for fields outside engineering. The example problem examined in this thesis is that of allowing domain experts to create decision support software. This is a way of testing the assumption that these domain expert's thoughts can be represented in a systematic diagrammatic way, and then translated to computer code, which will simplify the tasks of engineers, who experience problems in creating and sharing their software. The main emphasis is on engineers who need to develop and share models and information. The alternatives they have for creation of software are spreadsheets, which do not have collaboration and information modelling abilities sufficiently built in, or complex software that needs considerable expertise to use, and often still has insufficient collaboration or information modelling capabilities. The research is aimed towards the early stage of design where tree-based diagrams are most advantages, rather than the later stages where complex models such as CAD and Finite Element Analysis, are used. Crapo et al. (2002) assert the need for a methodology for creation of systems to enable more collaborative approaches to modelling by domain expert end-users, and that this combined with visualisation would allow engineers to model problems accurately.

1.2 Research Hypothesis

End-User Programming, and Modelling are both related to construction of this hypothesis. End-User Programming involves representing the program to enable end-users to interact with it. The way chosen as most relevant to this thesis is representation by visualisation of the problem. This visualisation effectively shows a model of the program, this model can be represented and interacted with diagrammatically. The target for testing this hypothesis is modelling, such as for engineering problems. So the terms User Driven Programming and User Driven Modelling are used in combination.

Therefore the research question to be answered in this thesis is - ‘To what extent is it possible to improve user-driven software development through interaction with diagrams and without requiring users to learn particular computer languages?’

To investigate this question, it is also necessary to investigate collaborative knowledge management in order to enable creation of a shared understanding of diagrammatic modelling and programming, and of the domain the diagrams/models/programs represent.

The hypothesis is that it is possible to create an End-User Programming environment, usable by non-programmers, which can have a wide variety of uses especially for modelling. This involves a technique termed throughout the thesis as User Driven Modelling (UDM) and/or User Driven Programming (UDP), which are both aspects of the same problem of enabling production of better and more accessible models/programs. The possibilities for a generic User Driven Programming (UDP) environment will be explained. It is possible to create an end-user visual programming environment using Semantic Web technologies, especially for modelling of information, where this approach is well suited. To achieve this it is necessary to link the information visually via equations, perform calculations, and store results for reuse and collaboration. This can make translation from humans to computers easier and more reliable than current software systems and languages. The use of Semantic Web languages as information representation and even as programming languages would assist greatly with interoperability as these modelling languages are standardised for use in a wide range of computer systems. The main focus is creation of modelling systems for decision support.

This hypothesis needs to be tested with a practical application via creation of a modelling/programming system for a real and common problem, supported by diagrams. The problem chosen is design for manufacture as this is a common problem in engineering that needs the use of product and process tree structures. Design for manufacture can also be aided by translation of early stage product design parameters into CAD style diagrams to aid later stage design. Further the problem is collaborative and this requires navigation of a graph/web structure, Semantic Web technologies can be used for this. These structures are also common to problems other than engineering.

1.3 Research Aims

The main aim is to improve collaborative software development through the interaction with diagrams and minimise the need for end-users to learn code. The thesis explains research into modelling problems using software. It advocates an approach to software that enables users to create models, and share them online. This can assist modellers to become end-user programmers and be in control of the model creation process.

This research arises out of projects to create systems to facilitate management of design and cost related knowledge within aerospace organisations, with the aim of using this knowledge to reduce the costs of designing and manufacturing products. This thesis identifies ways that problems arising from the model development process can be addressed, by a new way of providing for the creation of software. With experience from projects, which have used a combination of proprietary software solutions and bespoke software, it is possible to identify the approach of User Driven Programming (UDP) as an effective software development technique. This research brings together approaches of object orientation, the Semantic Web, relational databases, and Model-Driven and Event-Driven programming. Frankel et al. (2004) explain the opportunities for, and importance of this kind of research, and Uschold and Gruninger (2004) discuss such an approach. The approach encourages much greater user involvement in software development. The advantages of increasing user involvement in software development are explained by (Olsson, 2004). The intention of this research is to allow users to create the model/program they develop without the need for writing code. This research brings together End-User Programming, Modelling and the Semantic Web approaches, so the shaded area is examined. Figure 1 shows this :-

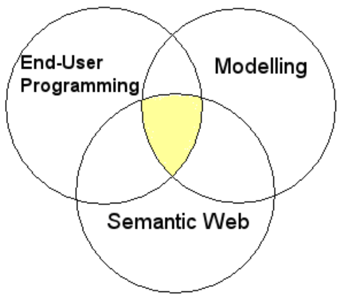


Figure 1. Research Area

Visualisation is important to all three of these research areas. The representation of program structure also needs to be visualised to End-User Programmers so they can create and edit content. Semantic Web research is not an end in itself as without the combination with End-User Programming/Modelling in order that people can create programs; there is insufficient incentive for creation and use of information and knowledge to be represented using Semantic Web technologies. The reverse of this combination is that end-user modelling/programming is not practical without creation and use of structured knowledge/information representation available through an accessible interface, such as representations using the Semantic Web. This fusion of research and technologies is illustrated by the relationships between the research shown by Lieberman (2000), and McGuinness (2003), which contain explanations and links between End-user Programming and Semantic Web research areas. In order to increase the use of Semantic Web technologies it is necessary to create applications that make use of the Semantic Web for practical applications. Enabling modelling with Semantic Web technologies could encourage domain experts to fill ontologies with useful information, so generating more benefit from their use.

Within this thesis the terms 'user', and 'domain expert' are used interchangeably. The user or domain expert may be an end-user, or an end-user programmer/modeller depending on their circumstances and purposes. The user is a domain expert who wants a problem represented and modelled using software. The domain is engineering, but this research could be applied to other domains.

This thesis involves automatically producing modelling software from ontologies. The methodology for this is creation of an ontology of items that hold equations and values, and then converting this automatically to a visual model. This allows non programmers to create and share models. An important area of research is a technique for end-user programming, that of allowing visual modelling of information. This corresponds to the type of work normally undertaken using spreadsheets. This research involves using Semantic Web technologies to enable end-user programming. The technology is applicable to any problem that involves user interaction, calculation, and modeling so it can be applied to a wide range of tasks and subject areas.

1.4 Objectives

This thesis outlines techniques used, in order to enable decision support during product development, whilst minimising dependence on specialist software and detailed programming effort. The User Driven Modelling/Programming (UDM/P) approach and its application to the systems modelling research explained above, is developed using examples ranging from visualisation and calculation of the area of a rectangle, to the modelling and costing of complex processes, and the visualisation of component design.

This research will examine creation of models and modelling systems, and how this can be eased so that a wider range of the population can achieve this. It will also identify ways that creation of models and modelling systems is similar to other types of programming, and how the research can be applied more generally. The main users of the system will be engineers; possibilities for wider usage will be explored.

The purpose of User Driven Modelling/Programming is to enable non programmers to create and adjust models in order to maximise the maintenance, extensibility, ease of use, and sharing of information in models and modelling systems. This is in order to develop a systematic methodology for creation of models that are adaptable and applicable to a range of situations, and so to enable end-users to produce better models of their domain problems. Section 1.41 explains the main objectives, section 1.42, and Figure 2 explain how these objectives are to be achieved. The research will investigate the following areas in order to ease the process of model creation.

1.4.1 Enabling Better and more Adaptable Modelling

Maintenance

Maintenance of models and modelling systems will be improved by :-

- A stepped translation process consisting of Step 1 - Ontology/Taxonomy creation, Step 2 Translation and Modelling, Step 3 - Translation and Visualisation (there is some overlap e.g. modelling is possible at all stages).
- Use of open standards to represent information in a format available to the maximum range of maintainers without being dependent on the computer system or software they use.
- Ensuring that the structure of the modelling/programming system and all its related information is visualised clearly.
- Minimising the amount of code necessary to create a model, and structuring the model so that all connections can be seen.

Extensibility

Extensibility will also be improved by the above means; this enables understanding of a model and so allows for easier re-use. Enoksson (2006) explains the advantages for extensibility of an open standard language, he used RDF for Conceptual Browsing on the Semantic Web. A clear structure and visualisation can be edited with fewer worries about unintended consequences (side effects). This is to be achieved by enabling model builders and users to modify the ontology, and to provide and use translation/modelling capability, and visualisation. This is the three-step translation process developed for User Driven Programming/Modelling. A person could make changes to whichever step is most appropriate depending on the task they are performing and their interests and preferences. McGuinness (2003) observes the importance of extensibility, "*Extensibility*. It will be impossible to anticipate all of the needs an application will have. Thus, it is important to use an environment that can adapt along with the needs of the users and the projects."

Ease of Use

- Maximising accessibility is important to ease of use and vice versa, use of open standards helps achieve this, together with enabling models to run on software and systems that people are widely familiar with. Ease of use should be facilitated by improving model structure.

- Ease of use enables interaction with models that is essential for understanding and trust of the models, which in turn enables sharing of information.

Clear structuring and visualisation of information also assists in making a modelling system easier to use.

Sharing of Information

Maintenance, extensibility and ease of use are the key drivers for sharing of information. Achievement of the objectives in these areas would enable collaboration. Ontologies are used as a way of representing explicit and implicit knowledge. This is visualised as a taxonomy type view of the ontology information. Lack of sharing of information and of the collaboration that this makes possible increases cost. For engineering and many other types of modelling, sharing of information is necessary for enabling of jointly accepted accountable models that demonstrate the cost and value, verification, and validation of processes.

1.4.2 Methodology for Achievement of Objectives

Achievement of the above objectives can make possible creation of manageable, maintainable, and flexible models. To enable these objectives, a diagrammatic representation of models will be used as well as the taxonomy based visualisation. This will make it possible for engineers to use an interface that many of them are familiar with.

Translation

Translation capabilities will be provided to enable better communication between computer systems, and between humans and computer systems. This will allow visualisation of chains of equations, which are common in cost modelling. This visualisation will make it easier for people to add and manage information in large models, and identify cost information. A cost modelling example will be used throughout the thesis, but this work is relevant to modelling in general. To model complex problems a structured approach is needed for representing explicit and implicit knowledge. A translation will be provided in 3 steps :-

- Step 1 - Ontology
- Step 2 - Modelling Tool
- Step 3 - Interactive Visualisation

Step 3 visualises the results and allows interaction with the information to establish the meaning of the results. The translation is based on Semantic Web standards to enable widespread applicability and help ensure this is a generic solution. The visualisation and interactions can be tree/graph based, spreadsheet type, and CAD style as necessary. A further alternative is translation to programming or Meta-programming languages so the information can be re-used by developers who are creating systems with these languages.

Information management and Interaction

This work will be based on information created and held in an Ontology, and accessed using Semantic Web technology. Cost models will be constructed from information chosen by users through an interface that interacts with the user to establish what information is required, how it should be processed, what calculations should be made, and how it should be displayed, e.g. as a diagram or taxonomy. A methodology that involves structuring of information through Ontology and Semantic Web techniques and enabling end-user programming through visualisation and interaction aims to achieve effective production of generic models. The research has been applied mainly to aerospace cost modelling.

The diagram below (Figure 2) illustrates how production of better and more adaptable and applicable models is to be enabled by meeting the objectives of enabling better Maintenance, Extensibility, Ease of Use, and Sharing of Information. These objectives will be enabled by better structuring and better visualisation; this requires work on structuring knowledge using Semantic Web and Ontologies, and enabling better visualisation through end-user programming techniques. This structuring and visualisation makes the models more accessible, and so easier to edit, reuse, adapt and maintain.

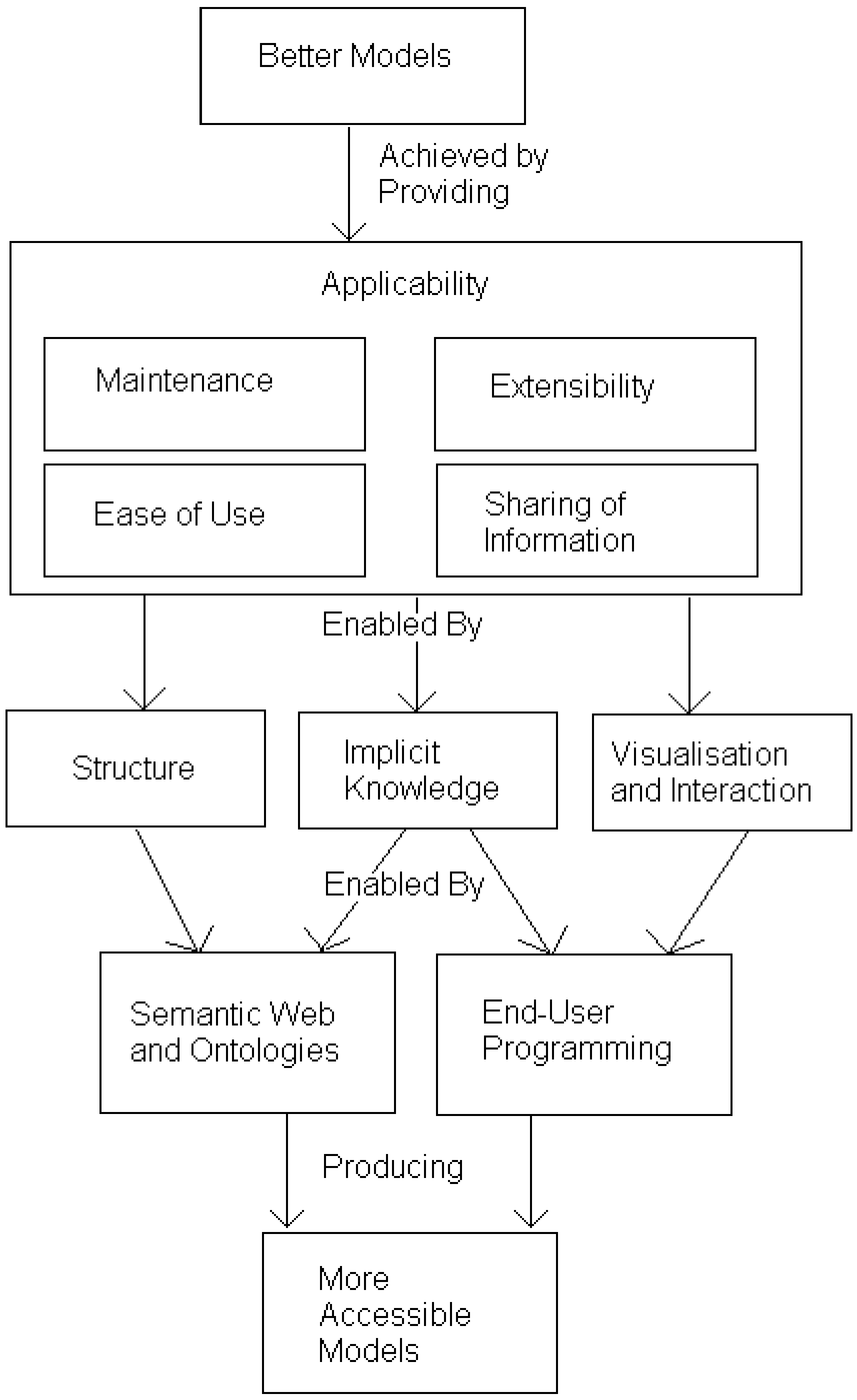


Figure 2. How Objectives and Methodology will aid better modelling

Semantic Web and Ontologies, and End-User Programming are realisation of the Knowledge Management enabled and supported by Visualisation and Interaction. Visualisation, Modelling, Translation and Management via an umbrella process/system and activities is a means for achieving the enabling process for the objective of enabling production of better, accessible, adaptable models.

Chapter 2 'Literature Review' explains the research of others in aspects of knowledge structure and representation, Semantic Web and Ontologies, Visualisation and Interaction and End-user

Programming/Modelling to achieve better and more accessible models. The thesis examines how a system for model creation could be developed and by whom, and assumes the system developer and user categories outlined below. A more detailed explanation of this methodology is provided in chapter 4.

1.5 Target Users

The methodology to achieve the objectives of Maintenance, Extensibility, Ease of Use, and Sharing of Information needs to be adapted to the way people work, with steps matched to people, skills, and roles. Ideally and as a long term aim it would be good to enable all users to interact with all parts of the program. However, organisations have hierarchies, employees have specialisms and for engineering the main argument to support this research is that of importance of this research for management of complex engineering projects rather than as better modelling, and user involvement being an end in itself.

Although a simplification of reality separating out the types of users into three basic types allows enough differentiation to provide a way forward for research and implementation of the methodology. Separating out the three main categories of user assists with matching the users to the tools and technologies they use, and supporting the overall knowledge transfer with a three step translation process. This would provide an infrastructure to enable collaboration. The actual follow up of analysing the collaboration then made possible is not covered in this thesis, which concentrates on the supporting structures and process.

1.5.1 System Creator

The system developer role is represented by people who would use the system advocated in this research in order to develop a domain specific system to create models. For the implementation in this thesis the author takes on the roles of a researcher who develops the capability for the 'System Creator', then represents the 'System Creator' who creates a modelling capability for a specific modelling problem. Then the author takes on the second role of a 'Model Builder' who uses the implementation to solve an aircraft wing related modelling problem. Also, some parts of the implementation were used by Airbus and Rolls-Royce employees both as System Creators, and Model Builders and Users.

This research is mainly aimed towards engineers who develop using spreadsheets, and for spreadsheets there are two main types of users. These types are :-

1.5.2 Model Builders

Model Builders are those who build and adjust equations (formulae) in spreadsheets and tend to protect their spreadsheet against unintended overwriting of cell values.

1.5.2 Model Users

Model Users are those who edit values in a spreadsheet and do not intend to overwrite formulae (though this can happen accidently).

Although there are grey areas between these three categories, it is intended that the thesis uses only these categories in order to simplify the problem of access control and model protection, and engineers would be familiar with this division. A more complex strategy for user rights and management will be left for future work, and in any case such strategies have been applied to work implemented in relation to this thesis by Airbus and Rolls-Royce.

Model Builders - Characteristics

Model builders will create or edit the semantic representation of the model in an ontology editor in order to create models. Model builders do not need knowledge of a programming language, but do need training in how to use the ontology interface to create a model. Also research will be undertaken into making ontology editing easier and more accessible. The system to be created needs to cater for different types of modellers. Expert modellers aim to build models, whereas novice modellers are more inclined to search for a specific numerical answer (Willemain and Powell, 2006). A problem mentioned by Willemain and Powell is that novice modellers often have little or no training in modelling. “Little is known about how they go about their tasks and whether they succeed.” So it is important to make technologies accessible for modellers, and to enable models to be shared and navigated, so that expert modellers can assist novices to create better models.

Model Users - Characteristics

Model users use the models created by model builders to make decisions based on their domain knowledge. This type of user manipulates the tree/graph representation to obtain a result based on the input values they know, or otherwise based on default values. They will use a model to evaluate a problem in order to help in decision making. Ernst et al. (2003) use a similar categorisation, they call model builders ‘modelers’, and models users ‘end-users’. Leaver (2008) uses categories of ‘Application Definer’, ‘RDF Querier’ and ‘Information Editor’ that are equivalent to categories of System developer, Model Builder, and Model User respectively.

1.6 Thesis Structure

In this introduction the aims and objectives were discussed and the theory and methodology introduced.

Chapter 2 - ‘Literature Review’ examines the research that was the main influence on this thesis. The areas investigated are 'Structure and Implicit Knowledge', 'Ontologies for Modelling and Simulation', 'Semantic Web and Ontologies', 'Interaction for Modelling with Ontologies', 'Visualisation and Interaction', 'Model-Driven Programming for Better Model Production', 'Accessibility of Models', 'Translation Steps', 'Better Models'. The main influences on the research are Ontology and Semantic Web research, End-User Programming, Modelling, and Model-Driven Programming. These areas were heavily investigated, as in order to enable the User Driven Programming/Modelling (UDM/P) approach it is necessary to improve accessibility of modelling to enable better model production in order to aid translation from implicit human knowledge into computer models. The means for this is improved user interaction, via visualisation of the structured representation of all relevant information. This is an adaptation of end-user programming towards modelling, and of model-driven programming by provision of improved end-user interaction to enable users to drive the process. This research is adapted in chapter 3 where a structured approach to the adaptation of this research is adapted to enable use and development of software tools to provide the structuring and translation of information in a way appropriate for visualisation to and interaction with users, collaboration, and modelling.

Chapter 3 - ‘User Driven Programming/Modelling (UDM/P) Theory’ examines the theory behind bringing together the research areas explained in chapter 2. Chapter 3 also examines the problem to be solved to enable UDM/P, and explains how Ontology and Semantic Web research can provide the infrastructure for a system to enable Modelling using End-User Programming techniques and translation influenced by Model-Driven Programming. Without Visualisation and Interaction research the UDM/P approach would not be practical, as the approach depends on user interaction and collaboration with the Ontology and Modelling system. Therefore the Ontologies, Modelling, and Visualisation and Interaction research examined in chapter 2 ‘Literature Review’ is necessary for UDM/P combined with research into Human to Computer translation to aid end-user programming. Empowering end-users to model and program, combined with the translation process then enables collaboration via improved Human to Human, and Human to Computer translation. The Position of software tools investigated, within a table, is analysed, to develop a way of combining tools for the User Driven Modelling/Programming (UDM/P) approach. This is examined in Table 2. Language and Tool Mapping and developed further in Table 3. Language and Tool Mapping - Further Development. The layered architecture achieved by a planned use and matching of software tools from Table 3 then aids in the research design and methodology to be outlined in chapter 4, by aiding interoperability to aid translation via open standards. Interoperability makes possible Computer to Computer translation, and translation from human to computer and vice versa through translation between layered open standard software tools. Those software tools nearer the human level of representation act as part of the translation process between humans and the lower layer computer systems/software. Thus structure of representations to enable translation, collaboration and improved interaction to increase end-user involvement in ontology and model development, and so to provide a human to computer interface, is all researched ready for planning of the research design and methodology outlined in Chapter 4.

Chapter 4 - ‘Research Design: Methodology’ examines the design and methodology for a mechanism to apply this combined research explained in chapters 2 and 3. The research design and methodology involves creation of a system to demonstrate implementation prototypes that follow the theory, design and methodology so far explained. This is the means for improving Maintenance, Extensibility, Ease of Use, Sharing of Information, by combining the best aspects of existing research and applying it to improve these factors and so enable User Driven Modelling/Programming (UDM/P). This combination is illustrated by the development of tools created from such research in the areas of Modelling, Spreadsheets, Ontology, and Semantic Web/Web 2.0. This is illustrated with a diagram that shows UDM/P in the middle as a way of combining the advantages of each technology. This is the important area of overlap explored through chapter 4. This overlapping of this research design and methodology leads directly from the research overlap between End-User Programming, Modelling, and Semantic Web. Each of the areas of technology - Modelling, Spreadsheets, Ontology, and Semantic Web/Web 2.0 are examined in detail. This is applied to diagrammatic modelling, and for this diagrammatic modelling (mainly tree based diagrams) to be translated from human computer and back, a translation process needs to be developed. This development is illustrated by diagrams of the translation process, which is aligned to a 3 step process of Step 1 Ontology, Step 2 Modelling, and Step 3 Visualisation. This 3 step process translates the source tree created as a diagram into a result tree and/or CAD style diagram according to what is most suitable to the user. This research design and methodology enables the implementation and prototyping outlined in Chapters 5 and 6.

The prototypes were created in two stages and stage each has a chapter to explain them. Chapter 5 - 'Development: Early Prototypes' examines the first stage of early prototyping to test ideas developed from the literature review and research design and methodology. This examination is in order to understand how to bring the research strands together, and implement them. In this chapter, the work conducted for the ACCS Aerospace Composite Costing System that is relevant to this thesis is examined. For the ACCS project an approach is taken of adapting and extending spreadsheet capabilities towards in depth cost modelling. This approach is not sufficiently generic for this thesis, so within chapter 5 more collaborative spreadsheet type tools are created and implemented as prototypes. This is still not sufficient for the requirement of enabling end-user modelling as it is necessary to provide user interaction direct with the ontology and model, by visualising the tree structure. This issue is examined in section 5.5. Section 5.6 outlines outstanding issues where further work is needed towards an improved implementation. Section 5.7 makes recommendations that need to be followed for the chapter 6 improved implementations. These recommendations are for tools and techniques that improve structuring, user interaction with and visualisation of the model structure.

The final prototype is developed later and is explained in chapter 6 - 'Development: Final Prototype Implementation'. This brings together the various elements of Ontology and Semantic Web infrastructure, translation, modelling and End-User Programming into a single system. Chapter 6 also demonstrates prototype examples of increasing complexity, of representing a rectangle, cost modelling of a manufactured cube, and cost modelling of an aircraft wing spar. These implementations are demonstrated in the context of the 3 step translation process, of Step 1 Ontology, Step 2 Modelling, and Step 3 Visualisation; and the overlapping areas of technology - Modelling, Spreadsheets, Ontology, and Semantic Web/Web 2.0. Survey results are then displayed and an analysis and reflection undertaken to prepare for the Chapter 7 Discussion and Chapter 8 Conclusions.

The way in which the final prototype examined the research question explained in the introduction is discussed in Chapter 7 - 'Discussion'. Chapter 7 examines how the prototype tested the design and implementation that was to meet the research question of 'To what extent is it possible to improve user-driven collaborative software development through interaction with diagrams and without requiring users to learn particular computer languages?' Chapter 7 examines how bringing together the research areas and combining them in a single implementation simplifies the problem analysed and makes it solvable. This examination prepares for the Chapter 8 Conclusion on the extent to which the research assists with the aims of improving Maintenance, Extensibility, Ease of Use, and Sharing of Information, as the step towards making diagrammatic User Driven Modelling/Programming (UDM/P) practical.

Chapter 8 - 'Conclusion and Further Research' examines the extent to which this research met aims of improving Maintenance, Extensibility, Ease of Use, and Sharing of Information. Meeting of these objectives is judged against the requirements for each objective and how these assist with meeting the overall aim of enabling diagrammatic programming. Different options are examined for enabling this diagrammatic programming, based on the research undertaken, and uses to which the research can be developed. Modelling areas and types and the use of the 3 step process are analysed in partnership with examination of the roles of users of the modelling tools to show who could model in what way. The analysis is made of how this assists with Maintenance, Extensibility, Ease of Use, and Sharing of Information. The conclusion is reached that for the research hypothesis of ‘to what extent was it possible to improve user-driven collaborative software development through interaction with diagrams and without requiring users to learn particular computer languages?’, this is possible for tree based modelling (common for engineering cost and process modelling) given the right tools and techniques as evaluated in this thesis. This concluding chapter also includes future work needed in order to fully meet the aim of improving collaborative software development through interaction with diagrams without requiring people to learn computer languages.

Chapter 2 Literature Review

2.1 Introduction

The main areas researched were ‘Structure and Implicit Knowledge’, ‘Interaction for Modelling with Ontologies’, ‘Visualisation and Interaction’, ‘Model-Driven Programming for Better Model Production’, and ‘Accessibility of Models’. These research areas were investigated in order to provide the knowledge needed for researching and development of the User Driven Modelling/Programming system explained in chapter 1.

2.2 Past Research of others and how this affects present modelling problems

The main areas researched were ‘Structure and Implicit Knowledge’ targeted towards ‘Semantic Web and Ontologies’, ‘Interaction for Modelling with Ontologies’, ‘Visualisation and Interaction’, ‘Model-Driven Programming for Better Model Production’, and ‘Accessibility of Models’. Tthe method for choosing these research areas was examination of the history of end-user programming, and relevant references were tracked down for the period since typing of code became possible to the present day, to establish where the gap was that needed to be researched to make diagrammatic end-user programming practical.These research areas were chosen as they were all necessary for the translation process from human based diagrammatic representations to computer models and code. These research areas were investigated in order to provide the knowledge needed for researching and development of the User Driven Modelling/Programming system explained in chapter 1.

Investigation of ‘Model-Driven Programming for Better Model Production, and ‘Accessibility of Models’ was the beginning of the design process for a system based on the research included in this chapter, in order to improve translation between person and software. This translation is to assist with providing better models.

Within each of the above subject areas the problems of enabling people to interact with models of their domain problems were researched. This was in order to provide the knowledge needed for research and development of the system explained in subsequent chapters. The literature review made it possible to investigate how to combine different research and technologies to create a modelling system for end-users.

2.3 Structure and Implicit Knowledge

2.3.1 Ontologies for Modelling and Simulation

The infrastructure of the research for this thesis is an ontology that can be visualised and edited in structured form. This ontology development is step 1 of a translation process that generates a modelling system and visualisation.

The literature review supported research for this thesis that will look to investigate a range of ontology representations as necessary in order to know when a simple representation will be sufficient and when it is necessary to move towards a more formal and machine readable representation, to ensure the modelling approach works. From the literature review it was found that the degree of formality employed in capturing ontology information can be quite variable, ranging from natural language to logical formalisms to facilitate machine understanding. This ontology research links with research into the Semantic Web for interoperable representation of information within each stage of translation.

Author(s)	Title	Comments
Gruber, T. R., 1993b	Toward Principles for the Design of Ontologies Used for Knowledge Sharing	• **Introduction Context and Audience** Formal Ontology in conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers. • **Issues** Definition and explanation of Ontologies, and explanation of how they can be used, and case study. Gruber (1993b, 1) defines an ontology, "An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where an Ontology is a systematic account of existence. For AI systems, what 'exists' is that which can be represented." Gruber goes on to explain design criteria for ontologies. Gruber explains software agents which can be a way of accessing and using ontology knowledge. • **Methods** Gruber defines ontologies and explains their design and use. Gruber examines how agreement could be achieved for ontology terms, and using an engineering case study examines how useful an ontology would be to engineers, and others who often make use of equations and values with standard units. • **Outcomes** This is a useful definition and explanation of ontologies and how to apply them to engineering, the explanation of how to use equations in an ontology is important as equations defined in an ontology can make possible an ontology based modelling system. This system would be generic if agreement could be reached on how to define equations.

Uschold, M., 2003 Uschold, M., 2006	Where are the semantics in the semantic web? Ontologies Ontologies Everywhere - but Who Knows What to Think?	• **Introduction Context and Audience** 9th Intl. Protégé Conference - July 23-26, 2006. This paper is especially relevant to those who manage information and modelling systems in industry. • **Issues** Uschold (2003) and (2006) outline a knowledge representation continuum of :- "**Implicit - Informal (explicit) - Formal (for humans) - Formal (for machines)**" Uschold (2003) and (2006) show a diagram of this continuum and include on it various things ranging from informal such as glossary, thesauri, through such things as schemas and UML to formal logic. Uschold (2003) explains that though the word Semantics actually means 'meaning' there is no clear definition of the term "Semantic Web", instead Uschold defines Semantic Web related technologies and ideas using examples. Uschold (2003) concludes that there is "consensus that the key defining feature is *machine usable Web content*." • **Methods** Uschold (2006) states that "there is nothing inherently good about being further along the semantic continuum. In some cases there will be advantages; in other cases there will not. What is good is what works." Though Uschold (2003) argues that evolution along this continuum will happen over time, as the relevant technology advances; (section 3.6 - 'Ontologies and Semantic Web and their role in Modelling'). His main point in both papers is to have in mind what the Semantics are to be used for and how they are to be used. More machine readability, and a reduction in ambiguity can be worth the extra effort if the project is sufficiently important and long term, because the advantages are "more robust, correctly functioning and easy to maintain" applications, Uschold (2003).

		• **Outcomes** This research explains and demonstrates the ideas behind an ontology based translation between humans and computers to aid modelling/programming.
Horrocks, I., 2002	DAML+OIL: a Reason-able Web Ontology Language.	• **Introduction Context and Audience** Proceedings of the Eighth Conference on Extending Database Technology (EDBT 2002). This is a more theoretical paper than that of Uschold above. • **Issues** Horrocks explains, "An ontology typically consists of a hierarchical description of important concepts in a domain, along with descriptions of the properties of each concept." He also discusses ontology languages and their role in assisting with interoperability. • **Methods** Horrocks argues the advantages of moving towards a more formal ontology. This is in contrast to Uschold's argument of only moving to more formal ontologies as necessary, however, this is merely a difference of emphasis as Horrocks argues for their theoretical importance, while Uschold is examining practical difficulties of finding and managing information for formal ontologies. Horrocks examines ontology languages such as RDF, RDF(S) (RDF Schema), and DAML+OIL, which provide languages for the layers 'RDF + rdfschema', and 'Ontology vocabulary' in Berners-Lees' diagram shown in section 2.3.2 Figure 3 of this thesis. Horrocks et al. (2003) and Miller and Baramidze (2005) explain other ontology languages OWL (Web Ontology Language) and SWRL (Semantic Web Rule Language). • **Outcomes** Horrocks explains Semantic Web technologies and the use of agents and ontologies, and ontology representation

		languages. This illustrates the linked nature of Ontology and Semantic Web research. Horrocks also examines the use of meta-data annotations and how they can make resources accessible to agents. Agents are another important aspect of Ontologies and the Semantic Web. Such research is the basis for provision of an ontology infrastructure for a visual rule and equation editor for modelling/programming. For this thesis RDF, RDF(S), DAML+OIL and OWL have all been used as structured ways to represent information.
Huber, G. P., 2001	Transfer of knowledge in knowledge management systems: unexplored issues and suggested studies	• **Introduction Context and Audience** Written for 'European Journal of Information Systems'. • **Issues** Huber examines the issues in ensuring people transfer their knowledge and suggests the organisation's culture is important in peoples' resistance to this. • **Methods** Analysis and Surveys. • **Outcomes** This resistance can be a further reason why employees often prefer putting their knowledge into spreadsheets under their own control, rather than into a shared knowledge base.

2.3.2 Semantic Web and Ontologies

Author(s)	**Title**	**Comments**
Berners-Lee, T., Fischetti, M., 1999	Weaving the Web	• **Introduction Context and Audience** The book covers early years of Semantic Web creation - Harper San Francisco; Paperback. • **Issues** Berners-Lee and Fischetti sum up the advantage of a Semantic Web program over programs in other languages. They write, "The advantage of putting the rules in RDF is that in doing so, all the reasoning is exposed, whereas a program is a black box: you don't see what happens inside it." The Semantic Web makes use of tree and graph relationships to relate information and people. This relating of information is explained as a 'web', and Berners-Lee and Fischetti, 1999 explain that the term 'web' is used by mathematics to denote a collection of nodes and links in which any node can be linked to any other node. • **Methods** Berners-Lee and Fischetti also argue for collaborative interactivity, which they call 'Intercreativity'. They explain, "the world can be seen as only connections, nothing else. We think of a dictionary as the repository of meaning, but it defines words only in terms of other words. A piece of information is really defined only by what it's related to and how it's related." … "There is really little else to meaning. The structure is everything." Berners-Lee and Fischetti make the point that it is not the power of the language that is important in providing this intercreativity. The simplicity of a language such as RDF makes it easier to provide interconnected solutions to complex problems, without becoming bogged down with the complexity

		of the language itself, and interoperability problems. • **Outcomes** So connectivity and structure are the crucial factors, enabling users to create and follow the information connections that are required for solving a problem and specify this to the computer. Berners-Lee and Fischetti also discuss the use of Semantic Web languages as programming languages and explain the benefits, declaring "The Semantic Web, like the Web already, will make many things previously impossible just obvious. Visual Semantic Web programming is one of those obvious things." An illustration of collaborative interactivity, connectivity and structure, and use of information connections for visual Semantic Web programming is Yahoo Pipes. This is discussed in section 3.3.
McGuinness, D. L., 2003. Berners-Lee, T., 2000.	Chapter - Ontologies Come of Age. Semantic Web on XML.	• **Introduction Context and Audience** McGuinness - In the book Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press. Berners-Lee W3C talk http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide1-0.html • **Issues** McGuinness (2003) provides a useful guide on how ontologies can assist in linking distributed data. This linking and connectivity is also explained by Uschold and Gruninger (2004). McGuinness cites a diagram by Berners-Lee (2000); this diagram is reproduced for this thesis in Figure 3 in this section. The concept illustrated, linked with that of ontologies contains representations of the place of each language in a stacked representation alongside the purpose of the language. • **Methods**

		This layered architecture enables an approach to interoperable and collaborative programming using Semantic Web languages for the middle layers including the ontology. McGuinness considers the role of markup languages in defining content to be machine readable. McGuinness encourages creation of web-based visual representations of information to allow people to examine and agree on information structures. McGuinness outlines 7 ways simple ontologies may be used in practice. These are :- 1 controlled vocabulary. 2 site organization and navigation support. 3 expectation setting. 4 “umbrella” structures from which to extend content. 5 browsing support. 6 search support. 7 sense disambiguation support. For this thesis ontologies were created for all of these purposes except 1 and 4, where ontologies of others were investigated. McGuinness also explains that ontologies can assist with resolving the meaning and consistency of terms. This increased understanding can aid interoperability. For a simple ontology this assists the human reader, and more structured ontologies can also assist automated systems to resolve terms. This subject is explored in sections 3.6 - ‘Ontologies and Semantic Web and their role in Modelling’, and 4.7 - ‘Building a System for

		Ontology and Semantic Web based Modelling'. • **Outcomes** McGuinness (2003) also explains with the aid of a diagram the level of definition in ontologies, from purely human readable to machine readable (Figure 2 in McGuinness' chapter 'Ontologies Come of Age'). This is also explained by Uschold (2003), Uschold (2006) (discussed in 2.31) and Uschold and Gruninger (2004). McGuinness writes about how ease of use via conceptual modelling support and graphical browsing tools is essential if systems are to be usable in the mainstream. So this is an area to be investigated for this thesis. This argument is expanded on in 3.6 - 'Ontologies and Semantic Web and their role in Modelling' and 4.21 - 'Improving and Building On End-User Interaction and Ease of Use'. McGuinness explains the need to examine and agree on information structures, using taxonomies/ontologies.
Berners-Lee et al., 2001	The Semantic Web	• **Introduction Context and Audience** Scientific American article. • **Issues** The Semantic Web was defined by Berners-Lee (2001) as "a web of data that can be processed directly or indirectly by machines". • **Methods** Berners-Lee et al. advocate an interdisciplinary open standard approach; this was later expanded to envisage a "Science of the Web". • **Outcomes** Berners-Lee and colleagues have envisaged the Semantic Web as a global database with the information held

		in a structured form where content is separated from formatting.
5 Berners-Lee et al., 2006	Creating a Science of the Web	• **Introduction Context and Audience** This was published in Science - 11 August 2006. • **Issues** Berners-Lee et al. also explain, "The Web is an engineered space created through formally specified languages and protocols. However, because humans are the creators of Web pages and links between them, their interactions form emergent patterns in the Web at a macroscopic scale." Berners-Lee et al. argue the importance of visualisation for navigation of information :- "Despite excitement about the Semantic Web, most of the world's data are locked in large data stores and are not published as an open Web of inter-referring resources. As a result, the reuse of information has been limited. Substantial research challenges arise in changing this situation: how to effectively query an unbounded Web of linked information repositories, how to align and map between different data models, and how to visualise and navigate the huge connected graph of information that results." This is discussed in relation to engineering modelling in section 3.5 - 'Ontologies for Interoperability and Reuse of Models'. • **Methods** This article advocates an interdisciplinary collaboration and systematic approach. Visualisation, navigation and interactivity can enable this. • **Outcomes**

		The above are key questions for this thesis, where information has to be held and managed to enable modelling. The Semantic Web provides the infrastructure for distributed collaborative modelling, interaction and visualisation.

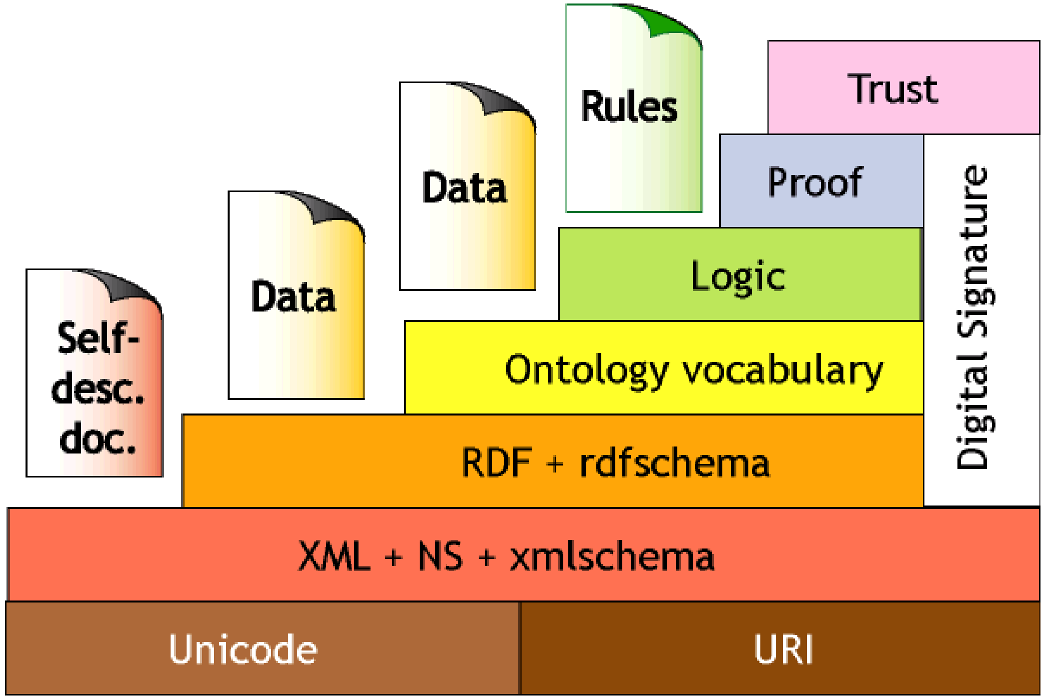


Figure 3. Layered Architecture, sourced from Berners-Lee (2000) and McGuinness (2003)

A diagram of Berners-Lee's, and McGuinness' layered architecture is shown above[1] in Figure 3. Corcho et al. (2003, 55) show a similar stacked diagram of the middle layers of the above architecture focussing on ontology development. Carroll explained in a talk based on this technical report (Carroll and Turner, 2008) that for OWL, RDF(S) and RDF "Each layer does not break the previous layer". This shows that for at least the middle layers of the above diagram translation is possible. This thesis mainly uses light ontologies as they can be represented in several layers whereas for a more structured ontology, information could be lost as it was translated down the layers. A more ambitious translation to and from all the layers, not just the middle ones would be future research, discussed in section 8.21.

For the hypothesis, investigation of Structure and Implicit Knowledge and breaking this down into Ontologies for Modelling and Simulation, and Interaction for Modelling with Ontologies, was necessary to find ways of providing structure. Then this structure was needed for enabling end-users to diagrammatically enter the information needed by the ontologies for collaboration and for the models, via representation/translation of diagrams created by the end-users.

[1] Semantic Web "Layer Cake" (2004) [online]. Available from: http://www.w3.org/2004/Talks/0412-RDF-functions/slide4-0.html [Accessed 25 July 2011], diagram now slightly different.

2.4 Interaction for Modelling with Ontologies

Ontologies can be made editable for maintenance and extension, and modelling tools based on ontologies can provide a structured system for building and editing of models. An ontology can store related information and calculations, any required calculations would then be made and translated to provide a model that can be interpreted by users. The research for this thesis is intended to solve the problems of translation from human to computer and vice versa. This is to be achieved by giving users more involvement in the translation process by letting them interactively model the problem themselves until they are satisfied with the solution. This allows the user to establish “common ground” with the computer, an expression used by Johnson (2004). As well as translating between the user and computer systems it is important to provide translations between different computer systems. An objective of this thesis is to make progress towards providing a modelling and simulation environment over the web as a product of translation from an ontology.

Author(s)	**Title**	**Comments**
Miller and Baramidze , 2005	Simulation and the Semantic Web	• **Introduction Context and Audience** Proceedings of the 2005 Winter Simulation Conference. This paper describes the process of building ontology based simulation, and analyses how to achieve this. • **Issues** Miller and Baramidze (2005) establish that for a “simulation study that includes model building, scenario creation, model execution, output analysis and saving/interpreting results. Ontologies can be useful during all of these phases.” • **Methods** However, they also explain that building generic ontologies for modelling and simulation is hard, as this is not domain specific and there is need for rigorous definitions of mathematical concepts. • **Outcomes** Miller and Baramidze advocate the use of interrelated ontologies rather than one monolithic ontology. For this thesis many related ontologies were used, divided based on commonly used engineering definitions and relationships.

Kim et al., 2002	A Two Stage Modeling and Simulation Process for Web-Based Modeling and Simulation.	• **Introduction Context and Audience** ACM Transactions on Modeling and Computer Simulation. • **Issues** Kim et al. describe their approach to modelling and simulation and how a Web-based solution can be applied to distributed process planning. • **Methods** They explain how techniques of generating executable code from documents specified in standardised XML (eXtensible Markup Language) can be used to create simulations. • **Outcomes** This standards and web-based modelling/simulation approach is an important area of research for this thesis, as it combines modelling with use of XML, other Semantic Web and domain specific standards, and the web to provide accessible models. Section 3.5 - 'Ontologies for Interoperability and Reuse of Models' builds on such research.
Cheung et al., 2005	Ontological Approach for Organisational Knowledge Re-use in Product Developing Environments.	• **Introduction Context and Audience** This paper discusses user friendly ontology tools in an engineering context. It was published for the 11th International Conference on Concurrent Enterprising - ICE 2005. Cheung et al. (2005) provide an ontology editor for knowledge sharing in manufacturing.

		• **Issues** Cheung et al. explain that until recently ontologies have been predominately applied in the medical informatics field. Cheung et al. argue that “With the development of user-friendly ontology editing software and automatic data exchange functions, the application of ontological approaches to exchange information across the WWW is most likely to be an essential aspect of the next generation of global knowledge management tools.” • **Methods** Cheung et al. demonstrate the importance of XML for interoperability and knowledge re-use. • **Outcomes** This is a useful guide to visualising and editing ontologies for use by engineers, and this combined with interoperability via open standards languages helps to make modelling practical. Linking ontologies with modelling tools will also make ontologies useful in engineering and science, and mathematics, whenever calculations are required.

The research into Interaction for Modelling with Ontologies was necessary in order to find ways to ensure it would be possible for computer literate end users to diagrammatically interact with the ontologies/models in order to provide meaningful information for translation to programs/models that could make and feedback calculations.

2.5 Visualisation and Interaction

Dynamic software systems such as outlined by Huhns (2001) and Paternò (2005) have been examined for this thesis. Huhns (2001) and Paternò, (2005) both explain that alternatives to the current approach to software development are required. Huhns argues that current programming techniques are inadequate, and outlines a technique called 'Interaction-Oriented Software Development', concluding that there should be a direct association between users and software, so that they can create programs, in the same way as web pages are currently created. Translation between ontologies, models, and visualisation enables translation between levels of abstraction, and therefore from human to computer and back. This thesis concentrates on visualising the entire program code to end-users as a model. This is how it could be possible to allow engineers and eventually others to program modelling solutions at the level of abstraction with which they are most comfortable. To achieve this, the User Driven Modelling/Programming (UDM/P) approach used in this thesis builds on Model-Driven Programming.

The use of functional language within nodes of a tree, web, or graph gives all the advantages of spreadsheet use in enabling programming by use of expressions, and allowing users to ignore problems of ordering calculations, and of memory use, so simplifying program statements and allowing something closer to natural expression. An additional advantage is displaying of expressions in the appropriate context, this is essential in order to allow collaborative end-user modelling and programming.

Author(s)	**Title**	**Comments**
Crapo et al., 2002	Visualization and Modelling for Intelligent Systems	• **Introduction Context and Audience** Visualization and Modelling for Intelligent Systems. *In:* C. T. Leondes, ed. *Intelligent Systems: Technology and Applications*, Volume I Implementation Techniques. • **Issues** Crapo et al. argue that spreadsheet users are considered as potential modellers. They state that "Every one of the perhaps 30 million users of spreadsheet software can be considered a potential modeller". Crapo et al. explain that in spreadsheets there is little support for the process of conceiving, building, validating, and communicating models, and that significant advances in modelling tools can be achieved using visualisation. Crapo et al. also state that "the modeler's task is to explore the data, understand the relationships relevant to the problem at hand, and capture those relationships in a computational model that will meet a specific need". • **Methods** Crapo et al. also explain that visualisation helps the modeller to maintain a hierarchy of sub models at different stages of development and to navigate effectively between them. This is the reason for breaking down the models into a tree or graph structure for the thesis. This also makes it easier to divide model to computer code and vice versa translation into manageable steps. Crapo et al. explain :-

		“Models are artefacts used to understand our world. As such they are embedded in intelligent systems as representations of knowledge. In the context of mining data to create knowledge, the modeler is often faced with discovering and understanding relationships in data that have no apparent analog in the laws of physical science. Sketches and diagrams as aids in problem solving and as a means of communication are as old as recorded history. The question now is: Can visualisation help us not only to discover the patterns and relationships in these data but also to use newly discovered knowledge to build computational models.” • **Outcomes** Crapo et al. observe that “representations suitable for a computer algorithm are often impenetrable to humans and vice versa, and that experts in modeling and optimization may prefer different representations than do experts in the problems domain.” The research of Crapo et al. has helped in understanding how structured visualisation can be related to structured modelling in order to enable modelling for people, e.g. those who are currently using spreadsheets.

Jackiw and Finzer, 1993 Guibert et al., 2004	The Geometer's Sketchpad:Programming by Geometry Example-based Programming: a pertinent visual approach for learning to program	• **Introduction Context and Audience** The Geometer's Sketchpad:Programming by Geometry. *In:* A. Cypher, ed. *Watch What I Do: Programming by Demonstration*. MIT Press, Chapter 1. Guibert, N., Girard, P., Guittet, L., 2004. Example-based Programming: a pertinent visual approach for learning to program. *In*: *Proceedings of the working conference on Advanced visual interfaces*. pp 358-361 • **Issues** Jackiw and Finzer (1993) and Guibert et al. (2004) demonstrate how a view of the problem that is more visual and nearer to the person's way of thinking can assist with the modeller's tasks. Jackiw and Finzer and Guibert et al. illustrate programming by demonstration, where the user visualises a problem by drawing a diagram of it. Guibert et al. (2004) explain and expand on Smith's (1977) work with an example demonstrating how numbers fail to reveal the concept behind them. The example is a numerical representation of a triangle. This representation is 'fregean' because it does not show the concept of a triangle. Next to this is a diagram of the triangle that does show the concept, this is referred to as 'analogical' representation because it includes the context of the information. • **Methods** Jackiw and Finzer describe programming by drawing diagrams that are converted to graphical representations. They also describe an example where a diagram is translated to a graphical representation; the authors explain this as 'spatial programming'. Jackiw and Finzer explain that this type of programming removes the distinction between programmers and users, and helps people to "understand how a geometric construction can be defined by a system of dependencies". • **Outcomes** Jackiw and Finzer and Guibert et al.'s research could enable end-users to customise software to model problems before and while they are trying to solve them, instead of having to request the software provider to add features, at great cost in terms of time, money, and added complexity of software. Investigation of such translation from different visualisations and representations is an important aim of this thesis, and is discussed in section 3.3 - 'Visualisation and Interaction for Modelling'. The translation process from the visualisation is explained in section 3.3.1 - 'Users and De-abstraction for Translation', and illustrated with an example Figure 49 in section 6.5.3.1.

Huhns, M., 2001	Interaction-Oriented Software Development	• **Introduction Context and Audience** International Journal of Software Engineering and Knowledge Engineering. Huhns explains that current techniques are inadequate for widening programming participation. • **Issues** Huhns outlines the technique of Interaction-Oriented Software Development in order to provide a direct association between users and software, necessary for end-user programming. This is a kind of visual diagrammatic programming. • **Methods** This direct user software association enables end-users to create programs via visual interactive diagrams, and also in the same way as web pages are created at present. This is illustrated by the use of Yahoo Pipes, explained in section 3.3. • **Outcomes** Huhns' research influenced this thesis by arguing and demonstrating interactive visual end-user programming. This can enable translation from models to software.

Fischer, G., 2007	Meta-Design: A Conceptual Framework for End-User Software Engineering	• **Introduction Context and Audience** End-User Software Engineering Dagstuhl Seminar. • **Issues** Fischer observes that it is the mismatches between end-users needs and software support that enables new understandings. Fischer argues that software development can never be completely delegated to software professionals, because domain experts are the only people that fully understand the domain specific tasks that must be performed. Fischer also argues for an approach to enabling end-user programming that makes it interesting to end-users. • **Methods** Fischer explains the concept of meta-design as aimed at creating infrastructures for collaborative design assuming future uses and problems cannot be completely anticipated during development of a system. • **Outcomes** This influenced the design and implementation for this thesis. Because it is so difficult to anticipate future uses and problems with software, an approach of meta-design to build systems for building systems is used. This approach (illustrated in section 3.3.2) assists with maintenance, extensibility and re-use.

Paternò, F., 2005	Model-based tools for pervasive usability	• **Introduction Context and Audience** Interacting with Computers 17. This paper investigates model-based tools for design of interactive systems. • **Issues** Paternò outlines research that identifies abstraction levels for a software system. These levels are ‘task and object model’, ‘abstract user interface’, ‘concrete user interface’, and ‘final user interface’. This is important in enabling end-user programming, such as for engineers to model problems and create programs at a high level of abstraction. Stages take development through to a user interface that consists of interaction objects. • **Methods** This approach can be used for automating the design of the user interface and the production of the underlying software. Paternò states that “One fundamental challenge for the coming years is to develop environments that allow people without a particular background in programming to develop their own applications”. • **Outcomes** Paternò goes on to explain that “Natural development implies that people should be able to work through familiar and immediately understandable representations that allow them to easily express relevant concepts”. Visualisation and Interaction for provision of a system for enabling end-user development such as this is discussed in section 3.3 - ‘Visualisation and Interaction for Modelling’.

For the hypothesis, investigation of Visualisation and Interaction was important in order to determine what kind of interface was needed in order to enable the interaction necessary by end-users in order to enter the information needed by the ontologies for the models.

2.6 Model-Driven Programming for Better Model Production

Model-Driven Programming involves creation of a high level interface for editing models to represent users' ideas, which can be translated to program code and alternative visualisations. Visualisation and interaction research is important for enabling model creation and editing by end-user programmers; Elenius (2005) illustrates tools for achieving this. The approach of Model-Driven Programming can assist with empowerment of users to customise software at a high level of abstraction, and so minimize the need for delegation of problems to IT professionals.

Model-Driven Programming involves two transformation techniques; these are Model Transformation and Program Transformation. Model Transformation can be used to translate a model with a representation of the problem that users would be familiar with, into a model with a representation that can be more directly translated into program code. Model Transformation can be applied to problems involving design models e.g. UML (Unified Modeling Language) diagrams, architectural descriptions, and requirements specifications, and to mathematical, engineering, and scientific modelling tools. Program Transformation is the act of changing one program into another. The languages in which the program being transformed and the resulting program are written are called the 'source' and 'target' languages, respectively.

Author(s)	**Title**	**Comments**
Erwig et al., 2006	Automatic Generation and Maintenance of Correct Spreadsheets?	• **Introduction Context and Audience** Proceedings of the 27th international conference on Software Engineering. • **Issues** Erwig et al. explain worries about the error rate in spreadsheets, "Given the billions of spreadsheets in use, this leaves the worlds of business and finance horribly vulnerable to programming mistakes". • **Methods** Erwig et al. demonstrate a system for automatically generating and maintaining correct spreadsheets. • **Outcomes** Improved spreadsheets can be part of a User Driven Modelling/Programming approach; this is expanded on in section 4.1 - 'Creation of Modelling System'. This argument is developed further in sections 4.2, 4.3, and 5.1.1, and solutions prototyped in 6.4.2. Erwig et al. have produced a spreadsheet add-on that can be used to improve spreadsheet accuracy. Erwig et al.'s research has influenced this thesis towards examining an aspect of end-user programming/modelling, investigating both improvements to and alternatives to spreadsheet modelling for this kind of computer literate end-user programmer/modeller.

Johnson, P., 2004	Interactions, collaborations and breakdowns	• **Introduction Context and Audience** ACM International Conference Proceeding Series; Proceedings of the 3rd annual conference on Task models and diagrams. This paper is about modelling human tasks. • **Issues** Johnson makes the point that UML (Unified Modeling Language) tools need to be extended to better enable modelling of collaborative tasks. Johnson explains that successful interaction requires mapping between levels of abstractions and that translation between the levels of abstraction required by users and computers is difficult. He explains that this problem often means systems are created that make the user cope with the problems of this mis-translation. • **Methods** Johnsons' research allows the user to establish "common ground" with the computer, by the use of translation layers to convert from a user level abstraction to a computer level abstraction. • **Outcomes** Johnsons' research influences the approach used in this thesis of translations between a domain expert level of abstraction and software. Also in this thesis translations are made between different views to give flexibility for representing information appropriately to different users.

Begel, A., 2007	End-user Programming for Scientists: Modeling Complex Systems	• **Introduction Context and Audience** End-User Software Engineering Dagstuhl Seminar. • **Issues** Begel emphasizes that if programming is left only to programmers rather than allowing domain experts to be involved, the program becomes a ‘black box’ and the domain expert cannot trust or verify the results. • **Methods** Begel argues that end-users may lack a mindset to form mental models of how to make the computer do what they want. Begel also explains that text based computer languages are often too obscure for end-user programmers. • **Outcomes** These problems of software languages being a black box and the obscurity of some text based programming language semantics were mentioned repeatedly by engineers during this thesis. So these became important issues for the thesis, and it was judged necessary to visualise software language semantics and translate these into appropriate representations. It is necessary to research and develop a visualisation and modelling environment that helps translate engineers’ ideas into computer models.

For the hypothesis, investigation of Model-Driven Programming for Better Model Production was necessary for analysing what was needed for the translation that provides the link between modelling and programming, and acts as the translator for stepped de-abstraction of diagram into code.

Design of a translation system to enable domain experts to create models/programs is illustrated in sections 3.3.1 and 3.3.2.

Research was also undertaken into how end-user programming has been developed over past decades. 2 main conclusions resulted from examining the history of end-user computing :-

- Research that created prototype systems for specialist users, school children, and other researchers and programmers, but had limited acceptance and use in the mass market can be reused with more up to date technology to assist development.
- More pragmatic research that involved creation of tools for the mass market, but which avoided more long term research issues can now be extended.

2.7 Accessibility of Models

2.7.1 Translation Steps

The translation steps are intended to ensure that users who are computer literate, but have little time to program, or knowledge of programming languages, can create software to represent a problem they want solved. Users would not have to write computer code. Instead diagrams, natural language (Mihalcea et al., 2006), and formulae would be used to define the source model. Mihalcea et al. challenge the assumption that formal programming languages are the only way to communicate with computers, and offer an alternative of mapping human language to program structures; they call this approach ‘Programming Semantics’. The aim is to convert natural language into program code.

This thesis concentrates mainly on diagrammatic and formulae based representations, and translation tools to convert this to code. As far as possible the tools and techniques used should be open standard for ease of use, re-use, and/or transformation for other hardware or software systems. The main advantage of open standard representation of information provided with the Semantic Web is that information can be transferred from one application to another. Additionally it provides a layered architecture that allows for a stepped translation from users to computer and back for conveying results of a modelling run. This approach to modelling is explained in more detail in chapters 3 and 4, and the implementation is explained in chapters 5 and 6.

2.7.2 Better Models

This section investigates ways collaborative model/program building can be supported and improved.

Author(s)	Title	Comments

Lieberman, H., 2007	End-User Software Engineering Position Paper	• **Introduction Context and Audience** End-User Software Engineering Dagstuhl Seminar. • **Issues** Lieberman asks "Why is it so much harder to program a computer than simply to use a computer application? I can't think of any good reason why this is so; we just happen to have a tradition of arcane programming languages and mystically complex software development techniques. We can do much better." • **Methods** Lieberman argues that researchers should use program transformation, and visualisation to make the end-user programming process as automatic as possible. In order that people can become 'End-User Software Engineers' without even realizing it. Lieberman also argues that HCI experts have concentrated on ease of use and should examine ease of programming. • **Outcomes** To achieve Lieberman's aims of enabling end-user programming it is necessary to undertake interdisciplinary research to combine different research approaches. Interdisciplinary research to enable automated program transformation is advocated in this thesis, to enable better modelling/programming, and wider access to this technology. The program transformation approach argued for by Lieberman is used in this thesis to translate between a domain expert End-User Programmer abstraction from/to models represented by Semantic Web languages, ontologies and code.

Bishop, J., 2006	Multi-platform user interface construction: a challenge for software engineering-in-the-small	• **Introduction Context and Audience** International Conference on Software Engineering, Proceedings of the 28th international conference on Software engineering. • **Issues** Bishop explains current problems: The current practice is for GUIs to be specified by creating objects, calling methods to place them in the correct places in a window, and then linking them to code that will process any actions required. If hand-coded, such a process is tedious and error-prone; if a builder or designer program is used, hundreds of lines of code are generated and incorporated into one's program, often labeled 'do not touch'. Either approach violates the software engineering principles of efficiency and maintainability. Bishop explains the importance of platform independence and re-use of GUIs. • **Methods** Bishop investigates, evaluates and advocates the use of platform independent programming languages, including the use of meta and declarative languages. Bishop ascertains that XML (eXtensible Markup Language) has become the base for many Meta-languages. Applying this to GUIs Bishop states, "Our premise is that it should be a software engineering tenet that a GUI can be specified as a separate component in a system, and then linked in a coherent way to the rest of the computational logic of the program." • **Outcomes** Use of platform independent and meta and declarative languages is an important part of the research for this thesis, in order to ease the programming problems explained above by Bishop. Current practice leads to the problems of Maintenance, Extensibility, Ease of Use, and Sharing of Information explained in this research, especially for large and collaborative complex systems, so an alternative approach is required. For calculation based modelling the model/program structure can be reflected in the GUI and they both have the same logical structure. This means it is possible to enable the model to generate a user interface, or the user interface to generate a model. For this thesis model builders and users should be enabled to construct information/model based systems and manage them. Section 5.5.2 - 'Better Representation and Structure' examines how XML can be used to create platform independent interfaces, reduce coding required and improve re-use.

Resnick, M., 1996	Distributed Constructionism	• **Introduction Context and Audience** Proceedings of the International Conference on the Learning Sciences Association for the Advancement of Computing in Education. • **Issues** Resnick develops and expounds the theory of constructionism, “the idea that people construct new knowledge with particular effectiveness when they are engaged in constructing personally-meaningful products”. Resnick goes on to say, “This vision puts *construction* (not information) at the center of the analysis. It views computer networks not as a channel for information distribution, but primarily as a new medium for construction, providing new ways for students to learn through construction activities by embedding the activities within a community.” • **Methods** This can also be applied to model creation for engineering, science and business. Resnick explains a theory that influences this thesis, known as ‘Distributed Constructionism’. This involves gaining an understanding of a problem by interacting with a knowledge building community, the problem to be modelled, and tools to model the problem, and build a solution. An example that Resnick cites is that of economic market simulation where participants became part of the simulation they constructed in order to understand economic models. • **Outcomes** Distributed constructionism theory can be applied with end-user programming and ontology modelling and building, which are explained in this thesis, and prototyped.

2.8 Conclusions - Focus for Developing a Theory and Apparatus

For the hypothesis, investigation of Accessibility of Models was necessary for enabling end-users to become end-user programmers by providing usable tools that translate diagrammatic representations into models/programs. This investigation is broken down into Translation Steps and ways to provide Better Models in order to focus on the need and way of translating, so as to focus on the main aim that is enabling end-users to provide better models than currently possible.

From the literature review (which covered End-User Programming, Semantic Web, and Modelling) a major conclusion was that this research should concentrate on where these research areas overlap. Investigating all these fields fully is not possible, and the nature of interdisciplinary research is that it is most necessary to investigate these overlaps where two or more fields need to be investigated simultaneously for a particular purpose. So this thesis aims to bring these fields together, investigate the overlap and implement solutions for this, rather than to investigate the full extent of all these fields.

These research fields are :-

- Structure and Implicit Knowledge
- Interaction for Modelling with Ontologies
- Visualisation and Interaction
- Model-Driven Programming for Better Model Production
- Accessibility of Models

A methodology for amalgamating this research for creation of a tool to fill the gap in the area of research where all these overlap is shown in 4.1 Creation of Modelling System - Figure 13.

The literature review indicated that the hypothesis needed revising to - it is possible to create an End-User Programming environment, usable by computer literate non-programmers, which can have a wide variety of uses especially for tree/graph/network based modelling.

Chapter 3 examines the theory behind bringing together the research fields covered in the literature investigated and explained in chapter 2. The overlapping fields investigated for the theory in Chapter 3 relate to and develop/extend the overlapping fields examined in the Chapter 2 literature review. Chapter 4 describes the methodology used for building on the research.

Chapter 3 User Driven Programming/Modelling Theory

3.1 Building a Theoretical Apparatus

The first stage of this research is to develop a theoretical apparatus based on and developing from selection of the approaches from Chapter 2. Following this, the next stage is based on developing of tools and methodologies for the practical application of research. Developing from the list of fields investigated and illustrated in the conclusion to Chapter 2, (Section 2.8), enables development of a theory and apparatus particularly for this thesis, and especially in order to build the diagrammatic modelling environment required. Applying the Chapter 2 list to this specific problem enables generation of a more specific list of areas to be researched, these are listed below, and each are covered by an individual section.

3.2 - Information Structuring and Navigation

3.3 - Visualisation and Interaction for Modelling

3.4 - Ontology Based Modelling Solutions

3.5 - Ontologies for Interoperability and Reuse of Models

3.6 - Ontologies and Semantic Web and their role in Modelling

3.7 - Meta-Programming as a Model Creation Technique

This is the infrastructure for examining construction of a system for modelling based collaboration.

This research examines a sociotechnical problem of peoples' interaction with computers for modelling problems (especially with tree/web-based structures). A sociotechnical approach involves putting people at the centre of technology as implied by the order of wording social/people issues are the greatest priority and the technology is to meet peoples' needs. As Berners-Lee and Fischetti (1999) explained (discussed in 2.32 - 'Semantic Web and Ontologies'), connectivity and structure are the crucial factors, enabling users to create and follow the information connections that are required for solving a problem, and to specify this problem to the computer. These matters of interaction, connectivity, and structure are the main factors in this research and in enabling end-user programming. These factors can be applied to connecting research and connecting information sources, furthering of this work is important, and can be achieved by enabling connectivity between the open source ontology, modelling, and visualisation tools investigated, with those tools and applications commonly used in industry and organisations. These applications already hold large amounts of information, sometimes they are legacy applications that have been filled with information, and have been used to solve modelling problems for many years. The following sections illustrate how technologies can be developed and applied to link, navigate, model, and visualise information held in different formats and technologies and so connect the people using this heterogeneous information. More detailed implementations are demonstrated in chapter 6.

For this thesis, prototypes are created at an early stage; this is in order to iterate through a process of reflection and refinement of the methodology. The software design cycle is speeded by adapting a Rapid Application Development approach of creating examples for the process of developing the research ideas, and the methodology and testing concurrently. As the relevant software technologies were/are developing at a fast pace, it is necessary to interact with these technologies to establish how the objectives of the thesis can be realised. This approach to the thesis is developed further in chapter 4 and illustrated in Figure 15 Section 4.6. As prototypes are created, research ideas are developed, and this development is co-ordinated with ongoing examination of related research of others, the theory and methodology is tested, refined, rectified, and proved in stages. The iterative approach leads to recursive revisiting of issues, moving from theory through methodology and implementation and improving each. The iterative research approach is adopted early and used in order to begin to develop the methodology that will be examined and further developed in chapter 4.

3.2 Information Structuring and Navigation

The theory behind this thesis is that it can be made possible for users to drive a model/program by specifying what they want and allowing the program to respond dynamically. For this thesis translations were performed. An example translation was created for and with Cost Indicating and Estimator (COINER)[2] (Figure 52), this translation was provided for Airbus. Such techniques were also tested by Marsh et al. (2001) and (2002).

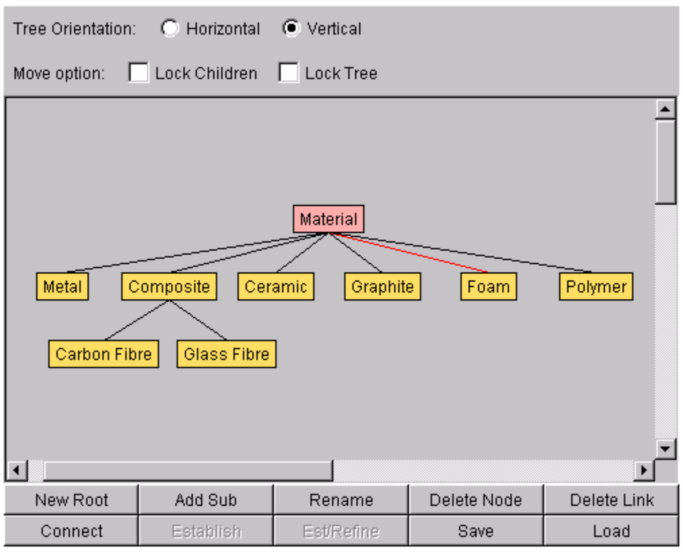
Another diagrammatic approach is that of Lee et al. (2000). They present a distributed visual reasoning system for intelligent information retrieval on the Web. For that research, the user can design a query by linking active icons, and then inputting the required parameters. Users can then see the structure of the query and obtain results from the information database. An approach is used in this thesis of linking nodes that represent items, values, and equation parameters, in order to visualise the whole model/program.

Tree/Graph View

Figure 4 illustrates experimentation with a Web-based interface. This is a modification of the Graph Layout demonstration that came with Java jdk1.1.4 (software development tool). This type of user interface allows a user to create a program using structures and visual queries and is similar to the COINER visual interface.

[2] Koonce, D., Judd, R., Keyser, T., Maisel, D. T., 2003. A hierarchical cost estimation tool. *Computers in Industry*,Vol 50 pp 293–302, [online]. Available from: http://engineeringmastersonline.ohio.edu/docs/CII_Better_Systems.pdf [Accessed 11 July 2011].

.


Tree Orientation: Horizontal [✓ Vertical]

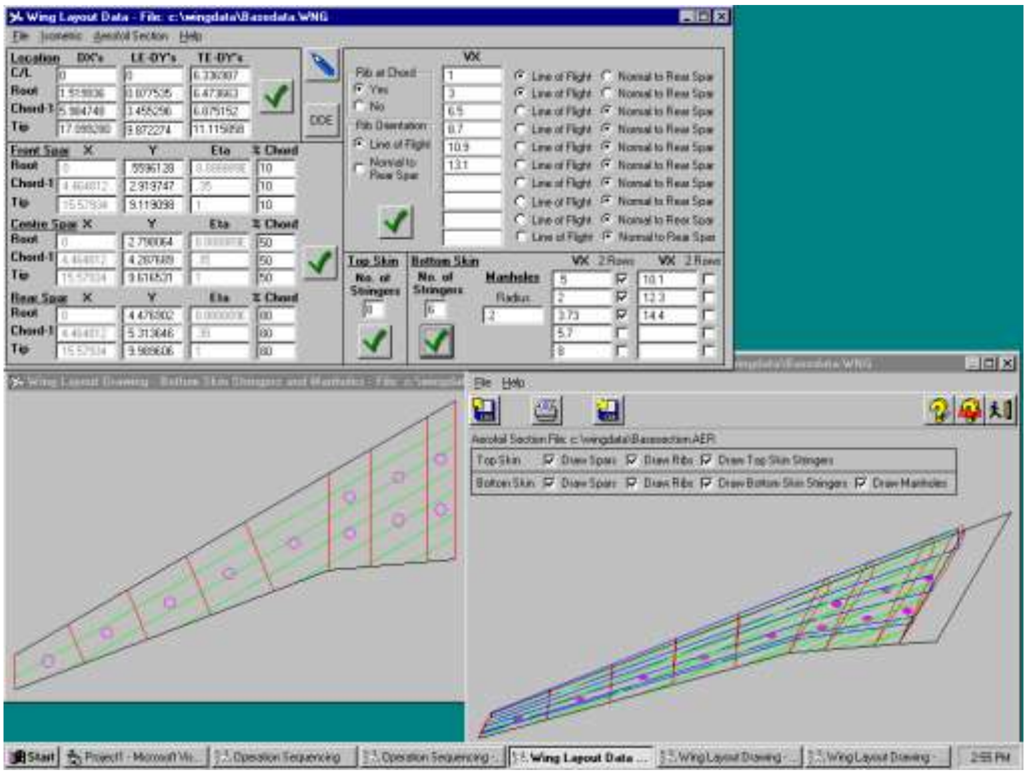
Move option: Lock Children Lock Tree

Figure 4. Java Graph Layout example, possible user interface for user driven program

Such an interface can be used to create or edit taxonomies/ontologies or to send information to a web-based decision support program. As well as inheritance relationships, such diagrams could model a manufacturing process flow. Enoksson (2006) models things a similar way for his concept maps that break an overall ontology down into concept sub-ontologies/taxonomies. It is important to alter the view as appropriate for users and situations, e.g. horizontal, vertical, or maybe circular layout, or a 3 dimensional look, as well as allowing variation in positioning of nodes, to assist with navigation. Modelling of processes is examined in section 3.5.1, and illustrated with examples in the appendix.

Diagrammatic View

This prototype demonstrates how wing design could be aided by modelling visually and interactively with feedback provided immediately. As the inputs change, calculations and the diagrammatic visualisation change in response. This was provided as a Visual Basic executable for testing, whereas the aim is to incorporate this functionality into a web-based open standard tool to be produced automatically from an ontology. Figure 5 and Figure 6 show the prototype.


Wing Layout Data - File: c:\wingdata\Basedata.WNG
File Isometric Aerofoil Section Help
DOE
Wing Layout Drawing - Bottom Skin Stringers and Manholes - File: c:\wingdata
File Help
Start
Project1 - Microsoft Vis...
Operation Sequencing
Operation Sequencing -
Wing Layout Data ...
Wing Layout Drawing -
Wing Layout Drawing -
2:55 PM

Location	DX's	LE-DY's	TE-DY's
C/L	0	0	6.336307
Root	1.519936	0.077535	6.473663
Chord-1	5.984748	3.455296	6.075152
Tip	17.099280	9.872274	11.115858

Front Spar	X	Y	Eta	% Chord
Root	0	.5596128	8.888889E	10
Chord-1	4.464812	2.919747	.35	10
Tip	15.57934	9.119098	1	10

Centre Spar	X	Y	Eta	% Chord
Root	0	2.790064	0.000000E	50
Chord-1	4.464812	4.287689	.35	50
Tip	15.57934	9.616571	1	50

Rear Spar	X	Y	Eta	% Chord
Root	0	4.476802	0.000000E	80
Chord-1	4.464812	5.313646	.35	80
Tip	15.57934	9.989606	1	80

Rib at Chord: (•) Yes ( ) No
Rib Orientation: (•) Line of Flight ( ) Normal to Rear Spar

VX		
1	(•) Line of Flight	( ) Normal to Rear Spar
3	(•) Line of Flight	( ) Normal to Rear Spar
6.5	( ) Line of Flight	(•) Normal to Rear Spar
8.7	( ) Line of Flight	(•) Normal to Rear Spar
10.9	( ) Line of Flight	(•) Normal to Rear Spar
13.1	( ) Line of Flight	(•) Normal to Rear Spar
	( ) Line of Flight	(•) Normal to Rear Spar
	( ) Line of Flight	(•) Normal to Rear Spar
	( ) Line of Flight	(•) Normal to Rear Spar
	( ) Line of Flight	(•) Normal to Rear Spar

Top Skin — No. of Stringers: 8

Bottom Skin — No. of Stringers: 6

Manholes — Radius: .2

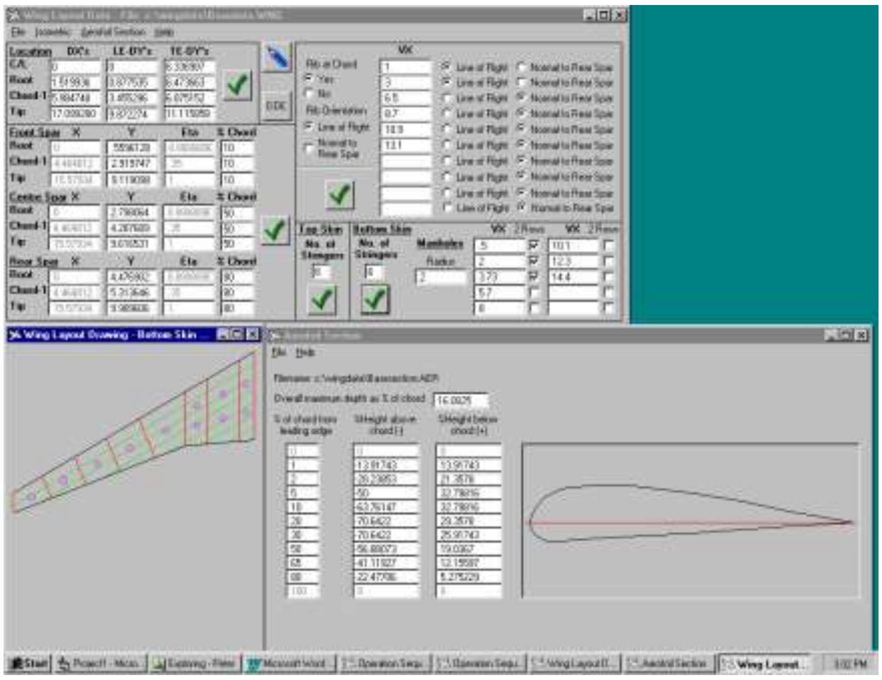
VX	2 Rows	VX	2 Rows
5	☑	10.1	☐
2	☑	12.3	☐
3.73	☑	14.4	☐
5.7	☐		☐
8	☐		☐

Aerofoil Section File: c:\wingdata\Basesection.AER

Top Skin ☑ Draw Spars ☑ Draw Ribs ☑ Draw Top Skin Stringers

Bottom Skin ☑ Draw Spars ☑ Draw Ribs ☑ Draw Bottom Skin Stringers ☑ Draw Manholes

Figure 5. Early End-user Modelling Example



Filename: c:\wingdata\Basesection.AER

Overall maximum depth as % of chord: 16.0025

% of chord from leading edge	%Height above chord (-)	%Height below chord (+)
0	0	0
1	-13.91743	13.91743
2	-28.23853	21.3578
5	-50	32.79816
10	-63.76147	32.79816
20	-70.6422	29.3578
30	-70.6422	25.91743
50	-56.88073	19.0367
65	-41.11927	12.15587
80	-22.47706	5.275229
100	0	0

Figure 6. Early End-user Modelling Example 2

Later (Figure 49) an automated construction system is provided for representing such interactive diagrams using open standards and represented on the web. The theory behind this section is that of showing examples using whatever method most puts across the information in an understandable way. This illustrates the concept that the visualisation and interaction represents and allows users to manipulate the information and get feedback on what has changed. This approach was used by Cypher (1993).

3.3 Visualisation and Interaction for Modelling

Visualisation and interaction with the model is more important than the result calculation. A visualised model can be edited when there are changes to the problem modelled. Visualisation shows the working of the model and calculation, the calculation can be constructed from the model, whereas the model cannot be constructed from a result figure. Eng and Salustri (2006) outline the role of computers in aiding decision making, and explain that the human mind is the best tool for making decisions. They explain that visualisation systems must help people use the information access capabilities of computers. Section 3.6 examines how such information access capabilities can assist people to model problems. So the research for this thesis aims to use the layered Semantic architecture described in section 2.3.1 Figure 3, and translate between the layers to enable human/computer translation. This approach is intended to improve interaction rather than enable computing decision making through artificial intelligence; the emphasis is on decision support for design and manufacture. The detail of this approach and the methodology for automating translation for users is explained further in sections 3.7 and 4.1. Such techniques as genetic algorithms are outside the scope of this thesis. Instead the emphasis is on clear visualisation, interaction and translation, of structured information.

A visual Graphical User Interface (GUI) can provide a high-level abstracted view of a model that could then be translated through a layered architecture, into a machine understandable format. This layered translation is demonstrated in chapter 6. Hudak et al. (2007) explain the history of Haskell (a language based on functions/expressions), its support for XML and Web scripting languages, and Haskell GUI research. A GUI is necessary for end-users to program/model using such a language in order to visualise the expressions. So some form of diagrammatic modelling environment is needed as an interface for whatever language is used. Functions have the advantage of being similar to what engineers use in spreadsheets, and can be easily visualised in tree form.

Naeve (2005) argues that “combining the human semantics of UML with the machine semantics of RDF enables more efficient and user-friendly forms of human-computer interaction.” Repenning (2007) explains the need for enhancements to UML to aid end-user programming. Repenning also argues that “Visual programming languages using drag and drop mechanisms as the programming approach makes it virtually impossible to create syntactic errors.” So, “With the syntactic challenge being - more or less - out of the way we can focus on the semantic level of End-User Programming.” Ways such research could be prototyped are examined in section 4.2.1 - ‘Improving and Building On End-User Interaction and Ease of Use’. This use of models that are visualised to users and enable interaction could make a high level model driven approach to production of better models possible.

It is then necessary to establish ways to combine the application of research for diagrammatic programming that emerged from the fields of End-User Programming, Semantic Web, and Modelling. This needs to be applied using technologies now available including new visualisation, modelling, and interaction tools, some available over the web. User Driven Programming is the application of the interface between End-User Programming, Semantic Web, and Modelling. Ways that research is pursued in this thesis in order to make User Driven Programming possible are :-

1. Semantic Web and Web 2.0 approach to enabling User Generated Content.
2. User Centric Extensions to UML (Unified Modelling Language) e.g. (Vernazza, 2007) this approach also ventures into 1 and 3, as it needs an interactive interface e.g. via the web that is clearly visualised, showing the structure.
3. Visual Programming extensions to spreadsheet type formulae based modelling, this is explained in section 3.3.1, and enabled using approach 1.

Approach 1 of Semantic Web and Web 2.0 interaction combined with approach 2 of diagrammatic modelling is illustrated by Yahoo Pipes[3], which provides for drag and drop editing of visual components that can combine, sort, and filter RSS data sources in order to automate web application development. This enables modelling of information, and using such diagrammatic programming combined with enabling of calculations would combine these three ways of producing applications. Yahoo Pipes enables creation of modelling applications with Semantic Web information, so this could assist in providing more incentive for provision of Semantic Web information and applications to use it. Yahoo Pipes is like a simplified version of UML (Unified Modeling Language). This kind of technology could open the possibility of such technology being used to read and eventually produce ontologies. Using UML for production of ontologies is as advocated by Baclawski et al. (2001), discussed in section 3.7.1, Kogut et al. (2002), and Enoksson (2006). Also such visual creation illustrates visual diagrammatic programming advocated by Huhns (2001) and Paternò (2005) (examined in section 2.5). Section 2.3.2 - ‘Semantic Web and Ontologies’ examines Berners-Lee and Fischetti’s (1999) argument that connections are all important for collaborative modelling of information and that these connections can be modelled diagrammatically. An example of an XMI (XML Metadata Interchange) representation of a node in a UML diagram is shown in Figure 71 in the Appendix, and an example using Yahoo Pipes is shown (Figure 72).

Use of options 1 (Semantic Web and Web 2.0), 2 (diagrammatic modelling), and 3 (Visualisation of modelling) illustrate that there is considerable overlap between these three approaches and they must be integrated within interdisciplinary research to enable User Driven Modelling/Programming. One approach to this is a Semantic User Interface; this is explained by Vernazza (2007). This interface can enable Drag and Drop programming that combines the benefits of all three research approaches. An important factor is to connect the user interface with the underlying code, so the two share the same structure and users can properly see how their actions can change the actual code. This enables collaborative user-driven programming.

[3] Yahoo Pipes (2011) [online]. Available from: http://pipes.yahoo.com/pipes/ [Accessed 26 May 2011].

Results can be visualised using stylesheets and interactive software, and translated into trees, graphs and other kinds of representations e.g. SVG (Scalable Vector Graphics) CAD style diagrams. Section 3.3.2 describes this translation process. Gross (2007) argues the need for end-user programming by designers using diagrams and scratchpads. Tufte (1990) explains how diagrams can be more effective than words for representing geometry. This links with the theme through the thesis of translating from an abstract to a concrete representation; Green et al. (2007) explain this distinction between abstract and concrete models. This distinction is more gradual than the distinction between classes and objects for object-oriented programming. Naeve (2005) examines this strong separation between types (classes), and instances (objects) and considers this to be a weakness, which he rectifies for ULM (Unified Language Modeling) developed from UML. The use and importance of translation from ontologies and Semantic Web data to interactive SVG is explained by a JISC report (Anderson, 2007, 52) that quotes Berners-Lee. Anderson's report explains the importance of SVG, and states that "At the WWW2006 conference in Edinburgh, when asked by TechWatch[4] about the likely characteristics of 'Web 3.0', Berners-Lee stated that he believes that the next steps are likely to involve the integration of high-powered graphics (Scalable Vector Graphics, or SVG) and that underlying these graphics will be semantic data, obtained from the RDF Web, that 'huge data space'."

Tools (further discussed in section 3.4), and techniques for building on the research of 2.4 - 'Interaction for Modelling with Ontologies' can enable ontology based modelling and visualisation. An extensive literature review and the summary of this identified that this area was under-researched. Further research helps build research and implementation in this field. This can then be applied to solve engineering problems as explained in section 3.5.1 with techniques investigated further in section 3.6, and demonstrated in chapter 6. Protégé[5] has OWL plug-ins available that provide extra capabilities for representing and visualising information, and also reasoning tools for maintaining and analysing the logical constructs (Storey et al., 2004) and (Elenius, 2005). The University of Victoria Computer-Human Interaction and Software Engineering lab (CHISEL)[6] has developed Jambalaya (Ernst et al., 2003) for visualisation of knowledge and relationships. Researchers at the University of Queensland Australia have developed a hyperbolic browser to display RDF files; this is explained in Eklund et al. (2002). Ernst et al. (2003) explain that the "larger ontologies that are being developed quickly exhaust human capacity for conceptualizing them in their entirety", so visualisation tools must assist users to view the information they need. McGuinness (2003) recommends using different presentational styles according to users and their needs:

[4] Techwatch (2011) *TechWatch (Technology and Standards Watch)* [online]. Available from: http://www.jisc.ac.uk/techwatch [Accessed 26 May 2011].

[5] Protégé (2011) *Welcome to Protégé* [online]. Available from: http://protege.stanford.edu/ [Accessed 26 May 2011].

[6] University of Victoria (2011) *Model Driven Visualization (MDV)* [online]. Available from: http://www.thechiselgroup.org/?q=mdv [Accessed 26 May 2011].

Presentation Style :- Possibly closely related to user type is presentation style. Some users need to see extensive detail, some need pruned information, and some need abstractions. Presentation of information may be textual, graphical, or other. While no one environment needs to support all presentation styles, it is important that the environment is at least extensible enough to have new presentation methods added when needed.

Fluit et al. (2003) establish the importance of enabling different views according to peoples' questions and purposes. Ernst et al. (2003) also discuss mapping between users' mental model of a system, and the system model. They suggest multiple views of the problem to help a user understand the problem. This subject is explored by Crapo et al. (2002), and is the basis of the visualisation techniques used in this thesis, to enable the user to create and understand models that are subsequently translated into software representations.

3.3.1 Users and De-abstraction for Translation

Scanlan et al. (2002) and Hale et al. (2003) explain about levels of abstraction in cost modelling, the intention here is to apply this to more general modelling problems relevant to this thesis. The translation from a high-level abstraction to code involves visualisation of chains of equations, which are common in cost modelling. This visualisation enables people to add and manage information in large models, and identify cost information. A cost modelling example is used, but this work is relevant to modelling in general. To model complex problems a structured approach is needed, for representing explicit and implicit knowledge. A translation will be provided in 3 steps, and the roles and skills of people who would make use of this translation are shown :-

Table 1. Users and De-abstraction for Translation

Step	**Person Role**	**Skills**	**Tool Type**
Step 1	System Creator	Programmer	Ontology and System Translator
Step 2	Model Builder	End-User Programmer	Modelling Tool and System Translator
Step 3	Model User	End-User	Interactive Visualisation

Step 3 visualises results and enables interaction with the information to establish the meaning of the results. The translation uses Semantic Web standards to enable widespread applicability and ensure this is a generic solution. The visualisation and interactions can be tree/graph based, spreadsheet type, and CAD style as necessary. Another option is translation to programming or Meta-programming languages so the information can be re-used by developers who are creating systems with these languages.

McGuinness (2003) considers the importance of supporting different types of user, "*Diverse user support.* Some environments are made for power users, some for naïve users, and some have settings that allow users to customize environments as appropriate to the type of user. It is important to determine if the environment can support all of the types of users anticipated.”

Though this research aims to make visual diagrammatic programming possible the emphasis is not just on visual programming. End-user programmers might prefer a visual interface and could use this but this is likely to have a speed and performance penalty. Therefore the translation must ensure that any code produced is accessible in text form as well as being visualised. This translation approach ensures that code can be produced in multiple languages, so this makes text based code editing more accessible. Use of open standards assists in this translation process.

The research is applied mainly to aerospace process and cost modelling. Cost models have been constructed from information chosen by users through an interface that interacts with people to establish what information is required, how it should be processed, what calculations should be made, and how it should be displayed, e.g. as a diagram or taxonomy. Structuring of information through Ontology and Semantic Web techniques, and enabling End-User Programming through visualisation and interaction can achieve effective production of generic models, and is demonstrated in later chapters.

3.3.2 Translation Process

The intention is to examine tools and technologies that can translate from a domain representation and/or abstract representation of a problem into program code, and examine a systematic way to make this possible. The translation process is shown in Figure 7; this process makes use of open standards languages, hence the interest in structured representation and open standards.

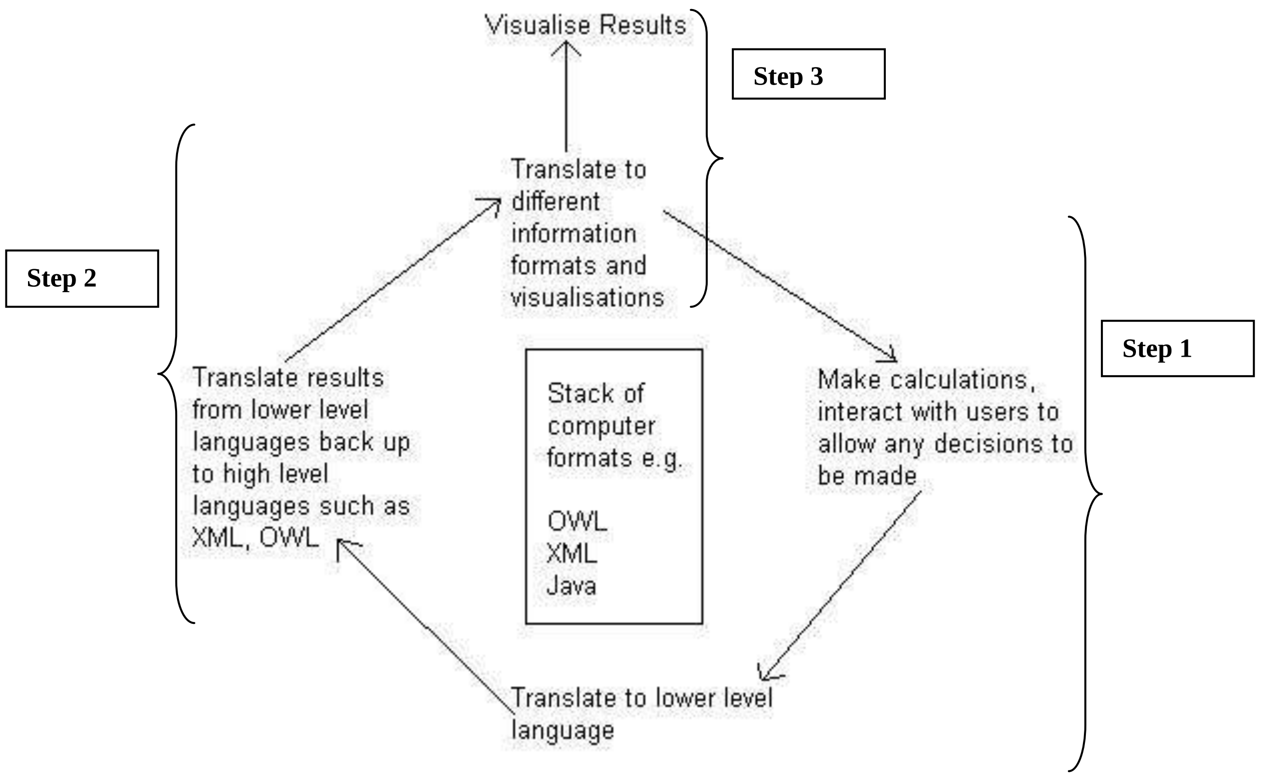

Visualise Results
Step 3
Translate to different information formats and visualisations
Step 2
Step 1
Translate results from lower level languages back up to high level languages such as XML, OWL
Stack of computer formats e.g.
OWL
XML
Java
Make calculations, interact with users to allow any decisions to be made
Translate to lower level language

Figure 7. Translation Process

Figure 8 illustrates how visualisation is to be incorporated into the translation process :-

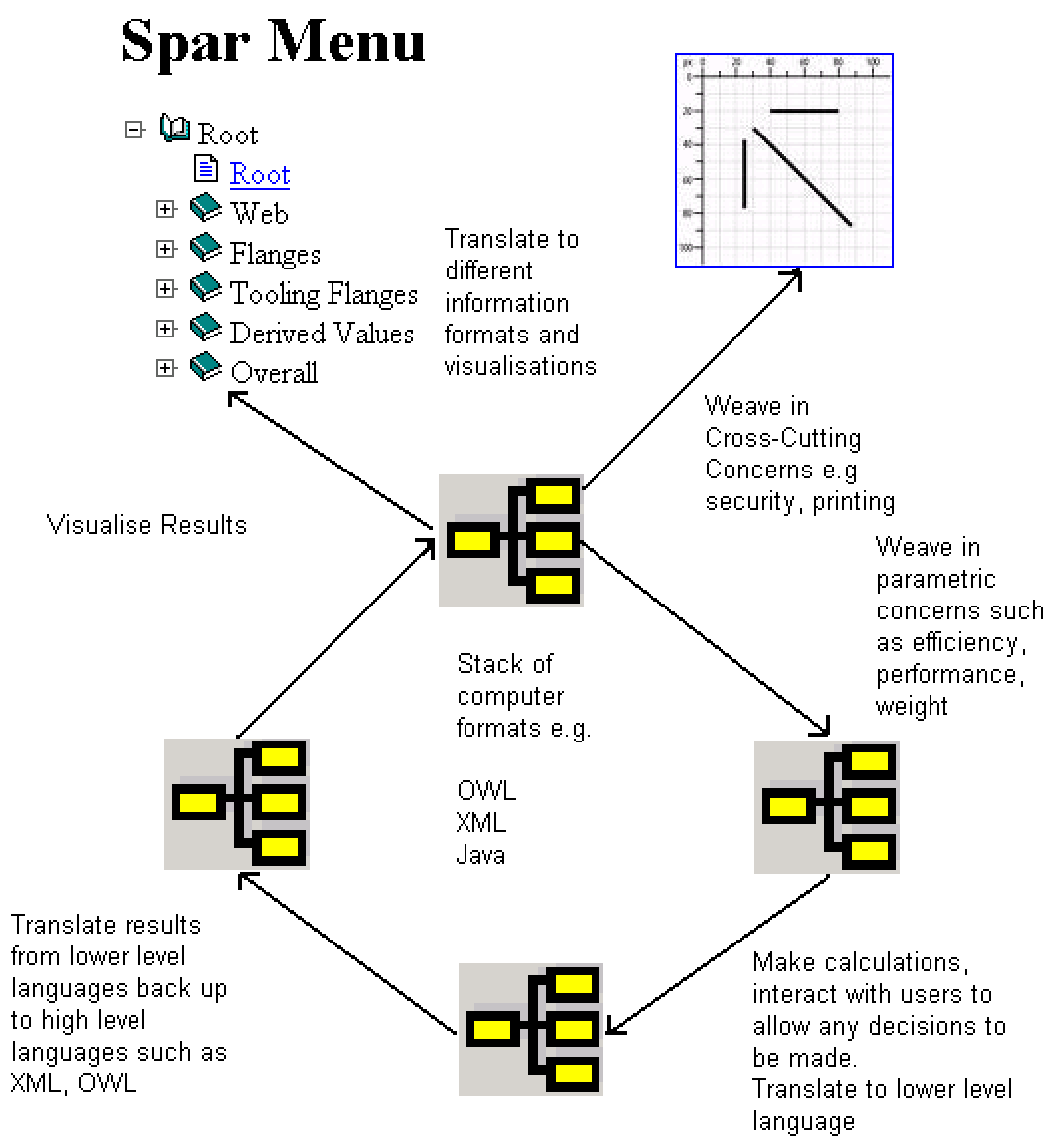

Spar Menu
Root
Root
Web
Flanges
Tooling Flanges
Derived Values
Overall
Translate to different information formats and visualisations
Weave in Cross-Cutting Concerns e.g security, printing
Visualise Results
Weave in parametric concerns such as efficiency, performance, weight
Stack of computer formats e.g.
OWL
XML
Java
Translate results from lower level languages back up to high level languages such as XML, OWL
Make calculations, interact with users to allow any decisions to be made.
Translate to lower level language

Figure 8. Translation for Interaction and Visualisation

To find alternative ways of representing models that do not require the user to write code it has to be easier to interact with and change the models, and to share and develop information with colleagues. The information used in the models resides in an ontology, and models can be automatically produced from this ontology via a recursive translation tool. Ontologies can also enable communication between computer systems, and between computer systems and users; Garcia-Castro and Gomez-Perez (2006) evaluate mechanisms for this, and section 3.5 examines ontologies for interoperability of models/modelling. In this thesis a technique is used of interpreting information in order to create decision support programs automatically, in response to user choices. This technique is then extended for use in the automatic creation of programs in various computer languages and systems. This is achieved by automated translation of the ontology and Vanguard System[7] information into other

[7] Vanguard System (2011) [online]. Available from: http://www.vanguardsw.com/solutions/application/modeling-and-simulation/
[Accessed 21 April 2011].

languages. Vanguard System acts as a reasoning agent on the translated ontology information to derive new information, as recommended by Uschold and Gruninger (2004). The basis of this process is that elaborators are nodes in the tree/graph, which are automatically created and dynamically write objects. Bechhofer and Carroll (2004) examine "how well the triple oriented RDF abstract syntax can encode the more conventional tree structured syntax for description logics." The translation process allows the wingbox (main wing structure) definition to be translated to the decision support system for costing, and then to other software such as web pages/applications for further processing and visualisation. An open standard semantic editor (Protégé) is used to structure the ontology information into related taxonomies. This ontology holds the definitions of nodes representing information, and calculations to be performed. Taxonomies/sub ontologies are created in Protégé for 'Parts', 'Materials', 'Consumables', 'Processes', 'Rates', and 'Tooling' for a prototype costing system. 'Parts' is the core taxonomy. New categories can be produced as required. Domain experts would edit the taxonomies; these experts could specify the relationships of classes and the equations to be used via a visual user interface in Protégé. These relationships are evaluated and translated to produce computer code. Uschold and Gruninger (2004) describe an approach of 'Ontology as Specification'. This involves building the ontology for the required domain, producing software consistent with the ontology, and translating between these. The translation process consists of :-

- Step 1 - Ontology
- Step 2 - Modelling Tool
- Step 3 - Interactive Visualisation

Figure 9 illustrates how code is produced from the semantic relationships.

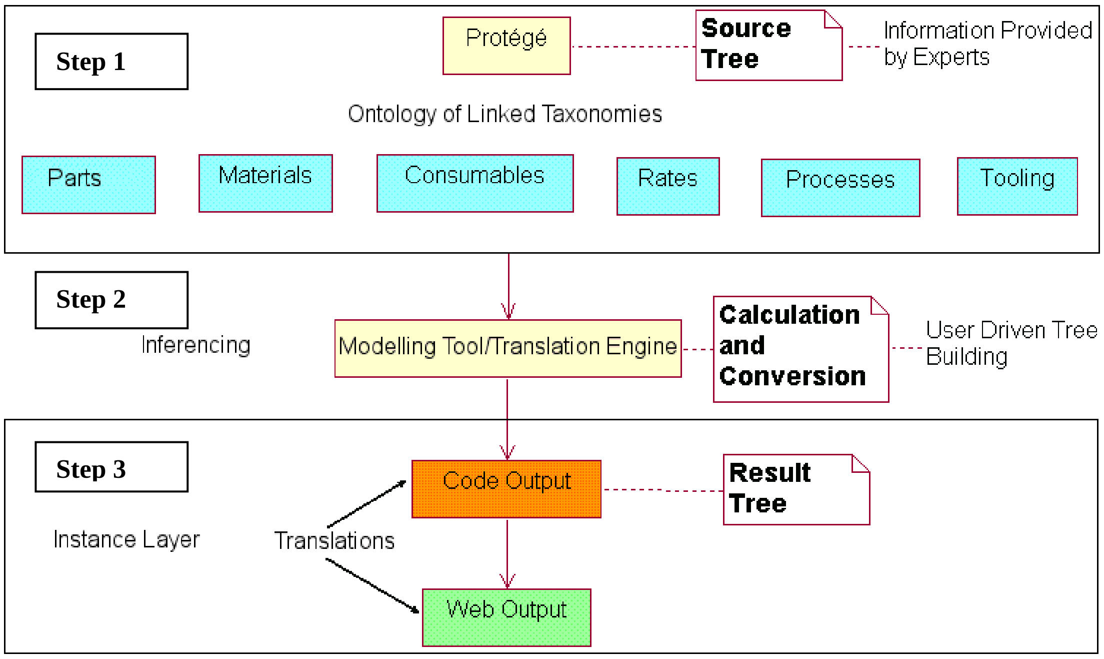


Figure 9. Translation Process Design

Uschold and Gruninger (2004, 61) describe the benefits of such an approach as "documentation, maintenance, reliability, and knowledge re-use". This translation process is also used for purposes that

Uschold and Gruninger describe as ‘Neutral Authoring’, and ‘Ontology-Based Search’ demonstrated in section 6.4.4.1. Uschold and Gruninger (2004, 61) describe neutral authoring as providing an “artifact authored in single language, based on ontology - converted to multiple target formats”. Code written for this thesis within Vanguard System provides multiple translations to different languages and formats from the Protégé ontology. Uschold and Gruninger describe the benefits of Neutral Authoring as “knowledge reuse, maintainability, long term knowledge retention.”

The model created (resulting from the translation) can be used as it is, or be a template for the generation of further models. An example interface, a section from a model produced automatically is demonstrated in chapter 6. This information is saved using a generic structure based on keys that define all relationships, into a relational database. This enables storage of hierarchical data in a relational database and also allows for separation of information into tables according to category, and the use of SQL (Structured Query Language) to automatically query and structure the information as required. This approach assists with the problem of mismatch between object and relational structures (further examination of this issue is in section 4.7). An advantage of using a stack of open standard representations is that this allows for choice in which search language(s) to use, for example XQuery (examined in section 5.5.3) can be used for XML, and SPARQL (6.4.4) for RDF and OWL. The main disadvantage is that either the structure must be simple enough to be represented at the lowest level of Semantics, or loss of Semantics through the downward translation must be allowed for and managed. This disadvantage is the reason for concentrating on use of light ontologies, as explained in section 2.3.2. Vanguard’s tree based decision support tool (Vanguard System) reads this information, by means of translation code added for this thesis, and represents it as colour-coded nodes. The translation code written for this thesis automatically queries the taxonomies/sub ontologies that make up the overall ontology, and links the information as required for the model. The translation code builds in all the links required for the equations and thus links information from different taxonomies/sub ontologies; the information is colour coded according to which taxonomy it is from. This same kind of translation code can be reused for any tree-based modelling problem, it builds the equations and follows the links to build each equation tree, and attach this to the rest of the tree.

The ontology represents the relationships between nodes in a machine independent way, so this makes it possible to translate the information into Meta languages via recursive querying. For Step 2 translation, SQL (Structured Query Language) is then used to access the underlying database representation of the ontology. These SQL calls cycle recursively through all the relationships and write out result code for each node, and each relationship automatically. The translation code reads node names and node types (e.g. class, attribute) so it can make an appropriate SQL call for each node, and make a copy in another language or system, then look for children or attributes. This allows any tree to be represented in computer languages. Then recursive routines write the programming language to be output.

Now this has been achieved, the problem that remains is that of needing to individually write a recursive routine for each programming language to be output. However, the recursive routines for each language are almost identical, as the syntax varies, not the semantics, so automated code writing produces models/programs. This assists with the problem of needing to individually write a recursive routine for each programming language to be output. This methodology opens up the possibilities for providing generic modelling capabilities that program with semantics and create meta-code syntax. Then it is possible to take the burden of learning multiple programming languages from the model builders and perform translations for them. Once this translation is achieved for a particular language the solution can be used indefinitely by different model builders for different problems. A future possibility could be providing the translation itself as a single meta-code solution that could itself be translated to multiple language systems; this is discussed in section 8.2.4 - 'Meta-Programming and Rule Based Programming'. As well as outputting programs, the translation system can produce Meta languages and/or Semantic web languages, and these research areas are likely to provide a way of translating the output meta-program into generic software. This makes possible future research into generating Meta programs based on the semantics defined in the visual diagrams, and so makes the solution more generic and independent of syntax.

This translation code reproduces a taxonomy/ontology and makes it available for modelling/programming systems. This taxonomy/ontology is a copied subset of the main ontology produced as an instance of the main ontology according to model builder choices and for the modelling/programming purposes of that model builder. So automated code writing produces models/programs. This is illustrated in Figure 10, which shows copying/translating from ontology to modelling system.

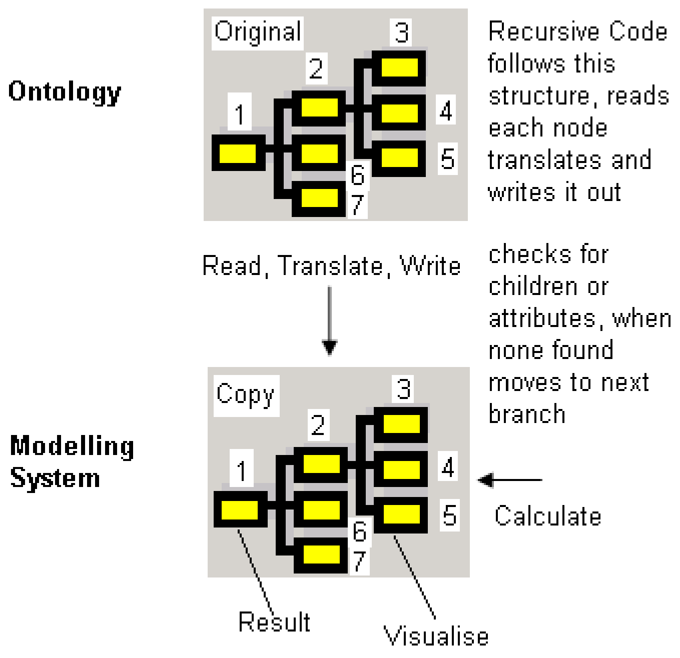

Ontology
Original
1
2
3
4
5
6
7
Recursive Code follows this structure, reads each node translates and writes it out
Read, Translate, Write
checks for children or attributes, when none found moves to next branch
Modelling System
Copy
1
2
3
4
5
6
7
Calculate
Result
Visualise

Figure 10. Recursive Translation - Automated Copying from ontology to modelling system

Also the translation can link different ontologies/taxonomies together when they are required in order to solve a problem. So the approach is to gather information from ontologies/taxonomies as required for solving a problem as specified by the model builder.

It is possible to create an interactive CAD type representation of a component for Step 3 of the translation if information is given in Step 1 and/or 2 to define how attributes of an engineering component or physical object should be drawn. This translation is similar to the kind used for CAD systems, such as CSG (Constructive Solid Geometry) and/or boundary representation BREP. The engineering component could then be represented as a tree, in STEP (**St**andard for the **E**xchange of **P**roduct Model Data), (shown in appendix), and as an interactive CAD type diagram, as appropriate. An example is given in 6.5.3.1 Figure 49 with sufficient detail to demonstrate this. So in a full system STEPml (XML-based STEP) could be used in this translation. SVG (Scalable Vector Graphics) is used for the visualisation of this Web demonstration[8]. Zhao and Liu (2008) examine mapping of STEP representations to ontology languages OWL and SWRL and how this benefits interoperability. Application of techniques to improve interoperability for this research is discussed in section 3.5. Zhao and Liu also show a diagram (similar to Berners-Lee's - Figure 3, page 32) of the position of OWL and SWRL in a stack of standards from XML in the Syntax layer up to OWL/SWRL in the Logic/Rule layer of 'Semantics'.

Figure 11 examines many of the ways translation could be used for User Driven Modelling/Programming. An example of the repeatable process of this design is that the 'System Translator Program' created in Step 1 produces a new 'System/Translator Program' in Step 2 which creates a Visualisation. This technique is used to translate an ontology to a CAD style diagram using a second stage of translation. The second 'System Translator Program' could also have created a 'Model/Program', 'Meta Program' or translate to an External Application. So, this is not an exhaustive diagram, as many types of translation not shown on this diagram would be possible. Another option is that Step 1 could be repeated to translate between ontologies.

[8] Hale, P. (2009) *Interactive SVG Examples* [online]. Available from: http://www.cems.uwe.ac.uk/~phale/InteractiveSVGExamples.htm [Accessed 25 July 2009].

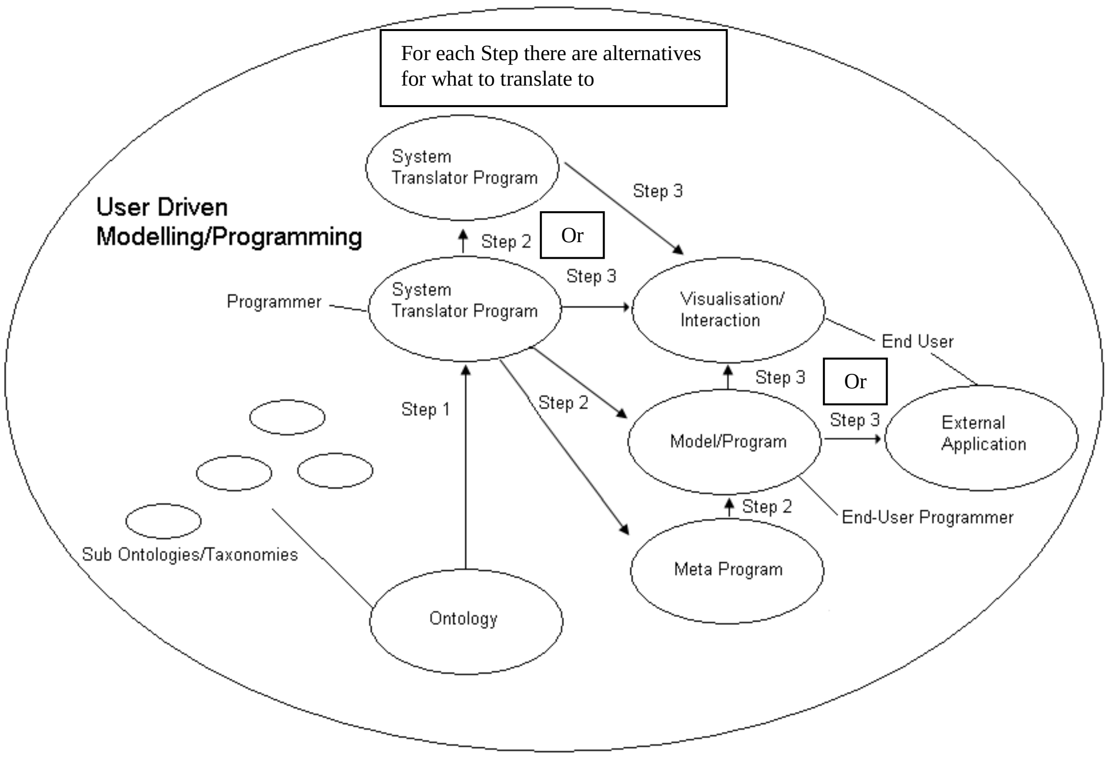

For each Step there are alternatives for what to translate to
System Translator Program
Step 3
User Driven Modelling/Programming
Step 2
Or
Step 3
Programmer
System Translator Program
Visualisation/ Interaction
End User
Step 3
Or
Step 1
Step 2
Model/Program
Step 3
External Application
Step 2
End-User Programmer
Sub Ontologies/Taxonomies
Meta Program
Ontology

Figure 11. Translation Process for User Driven Modelling/Programming (UDM/P)

3.4 Ontology Based Modelling Solutions

Cheung et al. (2007) assert the necessity for collaboration tools to support early stage product development within networked enterprises. They explain how greater use of collaboration tools can be achieved. Cheung et al. make the point that any collaborative system to support the early stages of product development can also be used for the later stages, providing there is an open standard basis for information representation. Linking of models with each other and with ontologies is essential in order to deal with the increased complexity of products and collaborative engineering that Cheung et al. explain. This linking will be enabled by techniques advocated in section 3.5 - ‘Interoperability and Reuse of Models’, and an ontology based infrastructure as described in section 3.6.2 - ‘Central Ontology Management’. Fensel et al. (2001) explain that organizations are often geographically dispersed and act as virtual teams, so virtual collaboration is essential. Corcho et al. (2003) examine the issue of ontology based collaboration and argue that:

“The future work in this field should be driven towards the creation of a common integrated workbench for ontology developers to facilitate ontology development, exchange, evaluation, evolution and management, to provide methodological support for these tasks, and translations to and from different ontology languages. This workbench should not be created from scratch, but instead integrating the technology components that are currently available”.

This integration can provide the infrastructure for ontology based modelling. Corcho et al. compare, contrast and describe ontology development tools, making use of a helpful tabular format. Naeve

(2005) gives an example of the need for “semantic mapping” between different words with the same meaning such as ‘author’ in one ontology and ‘creator’ in another ontology in order to establish interoperability and machine readability. McGuinness (2003) also investigates ontology tools/systems, and advocates their use for supporting collaboration for distributed teams. Uschold and Gruninger (2004, 59) show, with the aid of a diagram, the continuum from informal to formal taxonomies/ontologies, and the kind of tools that are used in each part of this continuum. Uschold and Gruninger (2004, 62) describe ‘*Ontology-based Specification:*’, stating:

“There is a growing interest in the idea of ‘Ontology-Driven Software Engineering’ in which an ontology of a given domain is created and used as a basis for specification and development of some software. The idea is to create an ontology that characterizes and specifies the things that the software system must address, and then use this ontology as a (partial) set of requirements for building the software… In the case of ontology-based specification, the ontology is used as the *basis* for software development. For example, the development of an entire suite of product design software (including viewing and presentation tools, data bases, and even marketing and accounting tools that are used to track product sales) could be driven from the same ontology. This would ensure easier interoperability among software systems whose relationships are typically only implicit. The benefits of ontology-based specification are best seen when there is a formal link between the ontology and the software. This is the approach of the so called ‘Model-Driven Architecture’ created and promoted by the OMG (Object Management Group), as well as ontology software which automatically creates Java classes and Java Documents from an ontology. When the ontology changes, the code is automatically updated. A large variety of applications may use the accessor functions from the ontology. Not only does this ensure greater interoperation, but it also offers significant cost reduction for software evolution and maintenance.”

Modelling systems investigated for this thesis must provide interaction with an ontology. This is an essential requirement because of the objective of improving Sharing of Information outlined in 1.4.1, and developed in section 3.5. Ontologies provide the infrastructure for Step 1 of the translation process. The systems investigated below also provide a mechanism for Step 2 translation of domain level modelling into open standard representation and vice versa. Ontology based systems provide a mechanism for translation/communication with external applications (Step 3). Vanguard System was used for this step 2 and 3 translation within a prototype system, but other tools/systems such as below would also be needed for a full scale implementation.

The following systems are not an exclusive list but they all use open standards for information representation and translation, so this ensures there should be a development path, whatever changes there may be in the software market. This use of open standards also ensures that a system can link with most environments used by others. This enables a translation and visualisation approach to ensure that new models can be added using the existing ontology/ies, and that design changes in the ontology and translation can enable modelling of different problems. So if there is a new problem to be modelled there are two ways to achieve this, adapt the ontology and/or adapt the translation and modelling approach.

Some ontology/modelling tools investigated are :-

- ACUITy (Adaptive Work-Centered User Interface Technology) is an ontology-based system and approach to modelling and implementing user interfaces built on top of Jena. ACUITy has been developed by General Electric, and its aim is to be used in decision support systems (Aragones et al., 2006).

- KAON is explained by Volz et al. (2003) as "an open-source ontology management infrastructure targeted for business applications. It allows for creation and management of ontologies, and provides a framework for building ontology-based applications." Volz also discusses issues in the development of an ontology based software environment, and explains why such an environment is required.

- The Metatomix Semantic Toolkit M3t4[9] is a set of standards-based plug-ins for the open-source Java Eclipse[10] development framework, which allows software engineers to create and modify semantic applications. Metatomix M3t4 provides for creation and editing of Resource Description Framework (RDF), and Web Ontology Language (OWL) resources.

- TopBraid Composer[11] is a visual modelling environment for creating and managing domain models and ontologies with the Semantic Web standards RDF Schema and OWL. The design and implementation of TopBraid Composer is lead by Holger Knublauch who was formerly the designer and developer of Protégé. This tool includes a diagrammatic editor for integrating applications and data sources, so is a useful tool for diagrammatic programming, and for assisting with interoperability (discussed in the next section - 3.5), and therefore for collaboration.

Protégé, Eclipse[12] and Jena[13] are used as construction tools to provide a development environment for creation and customisation of modelling systems. These systems are also built on open standard information formats; this means the systems can be interoperable, and enables people or organisations to make use of any of these systems without worrying about being locked in to that particular system. Protégé is used throughout this research but other tools have been investigated, and also would be used for future work (section 8.2.1 - 'Ontology Development').

The modelling tool Vanguard System was chosen because it handles Units and uncertainty well something that was necessary for the DATUM engineering modelling project with Rolls-Royce as partners (Scanlan et al., 2006). The same advantage was the case for the PhD because of the need for extensibility such as risk and uncertainty modelling. Other advantages of Vanguard System were ease

[9] Metatomix (2011) *m3t4 Dashboard* [online]. Available from: http://wiki.m3t4.com/homepage.action [Accessed 21 April 2011].

[10] Eclipse (2011) *Eclipse - an open development platform* [online]. Available from: http://www.eclipse.org/ [Accessed 21 April 2011].

[11] TopBraid Composer (2011) *The Complete Semantic Modeling Toolset* [online]. Available from: http://www.topquadrant.com/products/TB_Composer.html [Accessed 21 April 2011].

of installation and use, ease of linking to spreadsheets and databases, facilities for web enabling of models, and the ease with which formulae can be entered and linked with a high level programming language where necessary (similar to spreadsheets but with a tree based user interface). Experimentation with Protégé showed it was possible to translate the Protégé tree into a Vanguard System tree. This fits in well with the stepped translation to be developed. The open standard nature of Protégé made it possible to use this software without being locked in to it. Tools such as TopBraid Composer can provide additional higher level functionality such as an improved user interface and more tools for user interaction and modelling by end-users so is future work.

There was a need for the DATUM project and for the PhD to minimise programming, so Jena was not used, but Leaver's (2008) MSc project used this effectively, and there was regular contact with the developers of ACUITy (Aragones et al., 2006), to examine how that more software centred approach was used. Given more time that approach could have been used. Metatomix M3t4 was also used effectively as a high level tool to interact with Jena. So results of research with Jena and/or ACUITy and Metatomix M3t4 would have shown similar results to the approach of using Protégé and Vanguard System.

These types of tools improve with research and development each year so reproducing this research is becoming easier. New ways of modelling at high level with involvement of end-users is thus practical.

Table 2 illustrates the layered architecture on which these systems are built. This thesis uses this layered architecture approach to implement the translation for User Driven Modelling/Programming, which was illustrated in Figure 11. So the main benefit of researching these tools was the inspiration they provided for translation process design and implementation, from a domain level of representation to a code level. Table 2 in section 3.6 shows tools, technologies, and languages that can assist in this, and where they are based in a hierarchy from low level information centred interaction to high level user centred interaction (bottom to top), and computing focused to human focused representation (left to right). Table 2 also shows how each tool fits in with Naeve's (2005) analysis based on "characteristics of the three different semantic stages" of "Semantic Isolation, Semantic Coexistence, and Semantic Collaboration".

[13] Jena (2011). *Jena - A Semantic Web Framework for Java* [online]. Available from: http://jena.sourceforge.net/ [Accessed 21 April 2011].

An interactive version of this table with links to information and examples for these tools and technologies is available here[14]; this also explains all the acronyms.

Naeve (2005) describes Semantic Isolation where databases are available but hidden behind web portals, though the portals advertise their address. Semantic Coexistence is achieved by databases being structured in such a way that it is possible to search them without having to know their location. Naeve gives the example of RDF Schema - RDF(S), this standardises the structuring of the information across RDF(S) databases. RDF(S) provides standardised elements for the description of ontologies, so assisting to enable Semantic mapping. Semantic mapping enables Semantic Coexistence, and Semantic Collaboration due to Semantic mapping enabling agreement on terms. For the table above the argument presented is that high level user centred interaction (bottom to top), and computing focused to human focused representation (right to left), enable Semantic Coexistence. The tools in the top left are built from those below and to the right of them so the Semantic Coexistence is built from Berners Lee's (2000) Layered Architecture shown in Figure 3. Naeve (2005) argues the need for semantics that are understandable to humans as well as machines. That is an important objective of the research outlined in this thesis as without semantics that are understandable to humans, it is not possible for non programmer domain experts to undertake collaborative modelling.

3.5 Ontologies for Interoperability and Reuse of Models

This section builds on the research covered in section 2.3 - 'Structure and Implicit Knowledge', and section 2.3.2 - 'Semantic Web and Ontologies' where Berners-Lee et al. (2006) argued the need for agreeing open representations and visualising and searching them to aid collaboration. This is tested and applied to engineering modelling.

Many large companies, such as Rolls-Royce and Airbus (collaborative partners during this research), have outsourced the management and support of their IT systems to third parties. Strict management processes and procedures for the acquisition and implementation of new systems have been introduced. "A side-effect of this policy is a tendency for employees to make extensive use of spreadsheets and macro programming languages for information storage, analysis, and manipulation" (Scanlan et al., 2006). This paper goes on to explain, "These applications establish themselves as a legitimate part of the business processes of the organization despite the uncontrolled nature of their development." This is a problem as the spreadsheets have much functionality without visualising all relationships between variables clearly. The spreadsheets are also developed in an unplanned ad hoc way, so limiting interoperability, re-use, and collaboration. This problem and possible solutions are examined in more detail in section 5.1.1 - 'Spreadsheet Modelling'.

[14] Hale, P. (2011) *Language and Tool Mapping* [online]. Available from: http://www.cems.uwe.ac.uk/~phale/#LanguageToolMapping [Accessed 14 June 2011].

Most large organisations have key operational knowledge and information dispersed across different types of information systems. Ciocoiu et al. (2000) emphasize that as it becomes necessary to translate between more systems, the number of paths for translation increases exponentially. To improve interoperability, it is therefore necessary to provide either a translator or multiple translators, and the translators can be based on taxonomies/ontologies. Ciocoiu et al. (2000) write, "One of the major problems facing enterprises today is the lack of interoperability among the various software applications that the enterprise uses." They also mention that ontology tools could assist with concurrent engineering and design. Ciocoiu et al. explain how an engineering ontology can be made more rigorous in order to facilitate interoperability. This allows representation of, say, a product structure and its manufacturing processes together. A single node then is the only representation of that node within the model, with all its relationships depicted as arcs emanating/terminating at the node.

Uschold and Gruninger (2004, 58) argue that lack of semantics in data communicated between systems leads to "brittle systems that are limited in flexibility and expensive to maintain." Gruber (1993b), Berners-Lee and Fischetti (1999), and McGuinness (2003) all agree on the usefulness of ontologies for assisting with understanding and agreement on terminology, thus assisting with interoperability and collaboration. This interoperability is necessary for Semantic Coexistence and Semantic Collaboration. This use of ontologies for interoperability and knowledge exchange was discussed in sections 2.3.1 and 2.3.2. Even without agreement on ontology terminology, the use of open standards, and translation and mapping capabilities can assist with providing some interoperability through what McGuinness (2003) terms 'platform interconnectivity'. Uschold and Gruninger (2004, 61) also argue the usefulness of ontologies, describing an approach to 'common access to information':

Problem:

"information required by multiple agents, expressed in wrong format"

Solution:

"ontology used as agreed standard basis for converting/mapping"

Benefits:

"interoperability, more effective use/re-use of knowledge".

The ontology management techniques above assist people to argue and agree terminology, to assist with interoperability based on collaboration. Uschold (2003) argues "*The more agreement there is*", (between humans), *"the less it is necessary to have machine processable semantics.*" Or this can also be seen in reverse as the less machine processable the semantics is the more need there is for human agreement on terms.

Cheung et al. (2007) argue that open source development can avoid vendor lock-in, eliminate unnecessary complexity, give freedom to modify applications, and provide platform and application independence. The reason for preferring open-source interoperable systems over proprietary applications is given by Cheung (2005) as "there is no single management tool or data exchange format that can satisfy all requirements and overcome all the obstacles involved within a collaborative product development environment". So it is important to be able to see and alter source code to assist with interoperability and collaboration. This open source approach can be combined with use of open standards ontologies as advocated by Cheung (2005) as discussed in section 2.4 - 'Interaction for Modelling with Ontologies'.

3.5.1 Engineering Domain Specific Standards

Upper ontologies that use agreed terminology and ways of representing models/programs can aid interoperability, Uschold and Gruninger (2004) discuss this. Investigation of agreed semantics is important to 2 of McGuinness' (2003) 7 uses of simple ontologies; 1 '**controlled vocabulary**', and 4 '**"umbrella" structures from which to extend content**'. Uschold and Gruninger (2004, 62) examine STEP (STandard for the Exchange of Product model data) for interoperability of systems such as CAD, and process planning software; and Process Specification Language (PSL) for exchange of process/workflow/production planning, scheduling, and simulation information.

In addition to the use of open standard ontology languages, a further layer of agreed semantics can be used for the domain of manufacturing modelling. The Process Specification Language (PSL)[15]of the National Institute of Standards and Technology (NIST) is appropriate for agreed representation of manufacturing process semantics. This makes PSL appropriate for this thesis as a means to ensure the storage of process models. Process models created by users with the User Driven Modelling approach could then be made interoperable at the level of semantics. PSL defines a neutral representation for manufacturing processes. Process data is used throughout the life cycle of a product, from early indications of manufacturing processes indicated during design, through process planning, validation, production scheduling and control. In addition, the notion of process also underlies the entire manufacturing cycle, coordinating the workflow within engineering and manufacturing.

Time dependant ordering of processes is essential, in addition to the 'type-of' relationships represented for other sub-ontologies for this research. In XML, order is represented by position within a file, but higher layer ontology representation languages such as RDF Schema RDF(S), and OWL can represent sequences explicitly, PSL adds an engineering specific way of representing the sequence/process flow. A processing sequence can be represented in PSL; this is illustrated in the appendix with an example[16]. This example visualises a section from an illustrative PSL process sequence, and is rendered using an XSL (eXtensible Stylesheet Language) stylesheet. This representation illustrates that the standard, could be expanded to a full ontology, and could allow visualisation, navigation, and interactivity. PSL adds a layer of engineering semantics to generic semantics for communication between process

[15] National Institute of Standards and Technology NIST, (2011) *A Few PSL Basics*... [online]. Available from: http://www.mel.nist.gov/psl/ [Accessed 14 June 2011].

[16] Hale, P. (2011) *Process Specification Language Example* [online]. Available from: http://www.cems.uwe.ac.uk/~phale/XMLDemonstrators/psl.xml [Accessed 14 June 2011].

modelling tools. PSL 'Items' are declared as classes and used as instances. Process Specification Language and the NIST project are covered in Ciocoiu et al. (2000), they describe the use of PSL as a translator for communication between process planning and scheduler applications and their users. As Semantic Web languages were used and produced in this thesis it is important to find a Semantic Web representation for PSL, and to examine Semantic Web representation of design and business standards for industry. PSL can use XML, RDF, RDF(S), and its own semantics to add a layer of engineering meaning to these Semantic Web languages, for communication between process modelling tools, and for use in defining ontologies. Further information on how PSL uses Semantic Web languages to represent processes is in the appendix. PSL and other controlled vocabularies/ontologies are investigated by McGuinness (2003). The advantage of investigating both generic and domain specific standards is that it provides a route for the translation process for User Driven Modelling/Programming, from generic to specific uses and applications, and vice versa.

STEPml is a library of XML specifications based on content models from the STEP (STandard for the Exchange of Product model data) standard. STEPml XML specifications are automatically generated from STEP schemas. It is a standard for transfer of business information concerning the design, manufacture and support of goods. STEPml relating to XML is explained by Chan et al. (2003), who investigate automated conversion of STEP data into XML. Further explanations and an example[17] of representation and visualisation of STEPml are available in the appendix. It would be possible for the ontology used in this thesis to be integrated with product and process ontologies being developed as part of STEP and PSL (Process Specification Language) projects. The development of web technology and its popularity have created opportunities in the application of STEP. This makes it possible to link to a wider range of information and make this available on the web. Visualisation and translation of ontologies to PSL and STEPml could aid in communication between process engineers and designers.

For project management, PMXML (Project Management Extensible Markup Language) has been developed to make it possible for project management tools to communicate. It is competing against Microsoft MS Project data format and there is also the possibility of it being merged into UBL (Universal Business Language)[18] which has a wider use in e-commerce and business.

3.5.2 - Web and Interoperability Standards

Uschold (2003) states, "The assumption that there will be terms whose meaning is publicly declared and thus sharable is critical to making the Semantic Web work." Hence, the importance of such standards and agreement as discussed in section 3.5, and 3.5.1. Such standards are needed both for machine interoperability and for human communication, and both interoperability and human communication benefit from agreements on ways to represent terms, and agreements on meaning of

[17] Hale, P. (2011) *STEPml* [online]. Available from: http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/STEPml.htm [Accessed 14 June 2011].

[18] OASIS (2011) *Universal Business Language (UBL)* [online]. Available from: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl [Accessed 14 June 2011].

terms. In section 2.3.1 - ‘Ontologies for Modelling and Simulation’ research of Horrocks (2002) was discussed on how ontology languages can assist with interoperability. This interoperability provides the infrastructure for interoperable Semantic Web based applications. Uschold (2003) explains that ontologies, ‘semantic mapping’, and translation provide ‘semantic integration’; this benefits interoperability and collaboration. The use of Semantic Web languages as programming languages would assist greatly with interoperability as these languages are standardised for use in a wide range of computer systems (as explained by Berners-Lee and Fischetti (1999), discussed in 2.3.2 - ‘Semantic Web and Ontologies’. Anderson’s (2007) Joint Information Systems Committee (JISC) report explains that as an application becomes more popular, more people use it in order to communicate with others who use it. This enables exposing information using web technology, for re-use; Section 3.6.1 - ‘Semantic Web and Web 2.0 Collaboration’ contains more explanation of mechanisms for this. Use of interoperability standards can mitigate the problem that those using popular applications can only communicate with each other, and not with those who use other applications. This problem was discussed in section 3.5. The use of Semantic Web technologies and standards for collaboration is analysed in section 3.6.1 - ‘Semantic Web and Web 2.0 collaboration’.

3.6 Ontologies and Semantic Web and their role in Modelling

This section builds on the research covered in sections 2.3 - ‘Structure and Implicit Knowledge’ and 2.4 - ‘Interaction for Modelling with Ontologies’. This section takes the general research of 3.4 - ‘Ontology Based Modelling Solutions’ for a specific engineering context, in order to describe how the solutions explained in that section can be developed and used for modelling systems to aid decision support, and collaboration. Shim et al. (2006) emphasize that modern decision support systems need to support teams. This section discusses how Semantic Web/ontology based collaboration can enable cohesive ontology and model editing. Eng and Salustri (2006) talk of cohesion in software systems, and the ultimate aim of transparent systems that allow people to concentrate on the problem they want to solve, with minimal need for awareness of the systems and interfaces they are using. This collaboration using linked and interoperable systems helps people reach agreement on the meaning of terms, and encourages end-user modelling/programming by enabling simplified development of online applications. This end-user modelling/programming based collaboration could be an efficient way of managing large ontologies with multiple users. There is a need for Semantic Web applications in order to increase the amount of Semantic Web information that can be searched. This could result in a virtuous circle of Semantic Web applications creating Semantic Web information, and so justifying the creation of more Semantic Web applications to access it. This research advocates the use of Semantic Web applications for modelling and end-user programming, and integration into business applications.

McGuinness (2003) recommends the Semantic Web as a platform for conceptual modelling by non-experts linking to frame-based ontologies:

“A language should be usable with existing platforms and should be something that non-experts can use to do their conceptual modeling. The web is clearly the most important platform with which to be compatible today, thus any language choice should be able to leverage the web. Additionally, frame-based systems have had a long history of being thought of as conceptually easy to use, thus a frame paradigm may be worth considering."

So this thesis answers a requirement for accessible modelling by non-experts. A modelling environment needs to be created by software developers in order to allow users/model builders/domain experts to create collaborative and interoperable models. This modelling environment will be created using an open standard language such as XML (eXtensible Markup Language), and layers of semantics built upon XML. As discussed in section 2.4 Cheung et al. (2005) put forward the importance of XML for interoperability and knowledge re-use. This assists with the interoperability goals explained in section 3.5. As the high (user) level translation, a Semantic Web based modelling environment depends on tools developed in order to assist the user, provide an interface and manage the user interface. These tools are written by developers using lower level languages, in order to enable modelling by other developers and eventually end-user modellers.

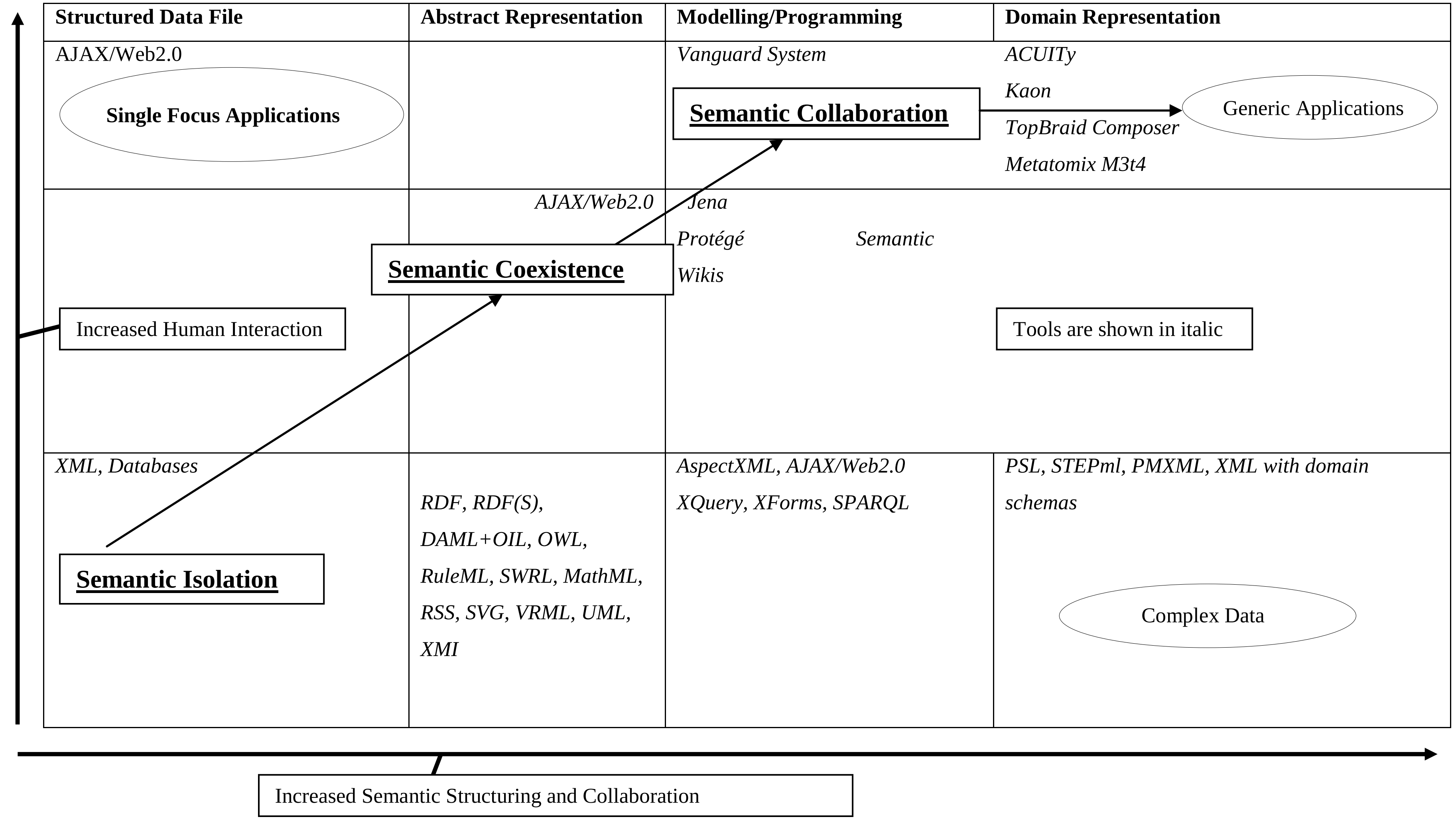
Section 2.3.1 - 'Ontologies for Modelling and Simulation' discussed the definition of an ontology by Gruber (1993b) "An ontology is an explicit specification of a conceptualization." Fensel et al. (2001) explain Gruber's ontology definition. They explain that conceptualization refers to an abstract model of a phenomenon in the real world which identifies its relevant concepts. Explicit means the types of concepts and the constraints are explicitly defined. Formal means the ontology is machine understandable. Uschold and Gruninger (2004, 59) also explain and expand on Gruber's definition in a similar way, and explain the application of ontologies. Gruber (1993a, 1) further explains the ontology definition, "For knowledge-based systems, what 'exists' is exactly that which can be represented". This illustrates the need to state everything needed for an ontology based model in a machine understandable way. Gruber (1993a) examines the role of formal ontologies in enabling machine understanding of information. Gruber discusses representation in ontologies for machine understanding, and difficulties in ensuring interoperability and collaboration due to the problems of agreeing terminology and system representation. The representation must also be understood by humans. Cheung et al. (2005) cite Davies et al. (2002) in explaining that ontologies "provide a shared and common understanding of a domain that can be communicated between people and application systems". Conceptualization is a simplified version of reality; therefore, there is no need to represent anything other than what is required for the model(s). The main difficulty is ensuring the ontology is represented sufficiently to enable machine understanding but also simplified enough to enable representation, modelling, and visualisation to humans.

Research in the use and visualisation of Semantic Web information can provide the tools that end-user programmers have been lacking until recently, and these tools can be used for modelling; Cheung (2005) makes this point. Horrocks et al. (2003) discuss the structure, syntax, benefits and use of ontology languages for information representation and how this assists automated agents. Horrocks et al. also talk of defining properties as general rules over other properties and of defining operations on datatypes. For this thesis, research such as that of Horrocks et al. assists with provision of a visual rule and equation editor and progress towards Semantic Web programming. An editing facility to model these equations and constraints, so that errors could be prevented, will improve the usability of visual modelling systems created. McGuinness (2003) refers to such a facility as ‘consistency checking’. Sutton (2001) illustrates how representing and translating knowledge into a knowledge based system for decision support is likely to be very difficult. Most people ‘just do’ a task and therefore never write down instructions for others; Cheung et al., (2007) also make this point. Eng and Salustri (2006) refer to a dimension from “tacit to articulatable” knowledge. Naeve (2005) argues that where knowledge is tacit it is vital to keep track of the individuals or groups who have this tacit knowledge, and that also the ‘Human Semantic Web’ can help elevate tacit knowledge to explicit. This problem of tacit knowledge highlights the difficulty of getting information into a knowledge base when it may be either only in individual’s minds, or completely unstructured. This issue and possible solutions are investigated in section 4.4 - ‘Ontology Tools’.

The web is a useful environment for enabling people to add their knowledge in both a less structured Web 2.0 way (development of less structured but interactive web tools/programs), and a more structured Semantic Web way. The greater interaction in the Web 2.0 approach at least makes it more likely that Semantic disagreements will be spotted, but it takes the structuring in the Semantic Web approach to then show the meaning of terms more clearly and unambiguously so that agreement or disagreement about and mapping of terms can be reached. This Semantic Web approach then makes possible Naeve’s (2005) ‘Semantic Collaboration’ through and also defining the ‘Human Semantic Web’ that Naeve advocates. This Semantic Collaboration then enables moving on from the Web being an environment only for simple tasks to one where sophisticated programs and models could be run that enable calculation and decision support.

This combination in approaches of enabling greater human interaction, and more definition of semantics is illustrated in Table 2. Language and Tool Mapping. Table 2 outlines positioning of software to decide where each software tool fits in the translation methodology to be devised for this research.

Table 2. Language and Tool Mapping


Single Focus Applications
Semantic Collaboration
Generic Applications
Semantic Coexistence
Increased Human Interaction
Tools are shown in italic
Semantic Isolation
Complex Data
Increased Semantic Structuring and Collaboration

Structured Data File	Abstract Representation	Modelling/Programming	Domain Representation
AJAX/Web2.0		*Vanguard System*	*ACUITy* *Kaon* *TopBraid Composer* *Metatomix M3t4*
	AJAX/Web2.0	*Jena* *Protégé Semantic Wikis*	
XML, *Databases*	*RDF*, *RDF(S)*, *DAML+OIL*, *OWL*, *RuleML*, *SWRL*, *MathML*, *RSS*, *SVG*, *VRML*, *UML*, *XMI*	*AspectXML*, *AJAX/Web2.0* *XQuery*, *XForms*, *SPARQL*	*PSL*, *STEPml*, *PMXML*, *XML with domain schemas*

Thus Increased Semantic Structuring and Collaboration from left to right, combined with Increased Human Interaction from bottom to top makes it more possible to undertake modelling and programming because the information is then well mapped and structured, and made available for visualisation and human interaction. AJAX/Web 2.0 technology spans more than one part of Table 2 depending on the emphasis of whether to structure it and/or enable greater interaction. This reflection also assisted in establishing the place of Vanguard System within Table 2 for the purpose of modelling/programming. To reach the top right of the diagram requires layered use of technology as per the diagram shown in Figure 3 referenced from Berners-Lee (2000) and McGuinness (2003). This layering of technology is needed in order to translate from the computer centred representations in the bottom left to the human centred representations and modelling in the top right. Human centred representations are too abstract for computers and computer centred representations are too abstract for humans. Therefore the technologies in the top right are not superior to those below and to the left as they need to be built on those technologies. Further there is more than one way to reach the top right, e.g. along the diagonal arrow from Naeve’s (2005) Semantic Isolation through Semantic Coexistence to Semantic Collaboration, or by moving up then right, or right then up. Following the diagonal arrow based on Naeve’s analysis is best for planning and building such a project from the start, but the other forms of navigation might be the best way to build on an existing project that has already been moving in a particular direction, that is not on this diagonal arrow. This arrow indicates a path for projects where the aim is to produce a solution that’s as generic as possible for modelling of complex variable problems. Where problems are simpler and less generic an alternative is to develop solution(s) in the top left of the diagram, ignoring needs to be flexible and generic and solving simple problems in a simple way. This is the approach that is useful for mobile phone apps for example. If a new problem arises a new app is developed.

Uschold (2003) argues that “the Web is evolving from being primarily a place to find things to being a place to do things as well.” Uschold argues that to enable this, an evolution of the Semantic Web “will take place by (1) moving along the semantic continuum from implicit semantics to formal semantics for machine processing, (2) reducing the amount of Web content semantics that is hardwired, (3) increasing the amount of agreements and standards, and (4) developing semantic mapping and translation capabilities where differences remain.” Uschold defines the Semantic Web as being machine usable and associated with more meaning. Uschold notes that “in order to carry out their required tasks, intelligent agents must communicate and understand meaning”. Miller and Baramidze (2005) explain that “Finding information in the new Semantic Web will likely become some hybrid of information retrieval, navigation and query processing”. This is researched and implemented within this thesis with example applications (chapters 5 and 6). Research examined in section 2.3.1 - ‘Ontologies for Modelling and Simulation’ provides an infrastructure for this; and research in 2.3.2 - ‘Semantic Web and Ontologies’ provides a mechanism. Miller et al. (2001) explain the technology behind web-based simulations, and argue the need for demonstrating the application of web-based simulations for major projects. Kuljis and Paul (2001) evaluate progress in the field of web simulation. They argue the need for web-based simulations to be focussed on solving real-world problems in order to be successful. This pragmatic approach allows for examination of the problems of definition of terms and of provision of editing facilities. This thesis examines and demonstrates solutions to a ‘real world’ problem of aerospace process modelling and costing.

Uschold and Gruninger (2004, 58) emphasize that “people use terms differently and mapping and translation must take place across different communities.” Uschold (2003) also examines this issue. Ciocoiu et al. (2000) explain that growing complexity in manufacturing hinders the ability to share information as the meaning is affected by its context. Engineers may have different names for the same thing, e.g. wing skin stiffeners may be referred to as stringers, but rib stiffeners are never called stringers. There is a relationship of stringer to stiffener, which needs to be defined, and this definition depends on the context; Green et al. (2007) discuss these terminology problems. This problem shows the need to achieve Naeve’s (2005) aim of ‘Semantic Collaboration’ to agree terminology where possible or map terms to each other when they have a shared meaning. This was the direction of the thesis from the beginning but Naeve articulates and defines what such achievement means. A classification scheme or ontology is necessary in order to make communication precise. Such an ontology can also be used to help non-specialists understand the terminology of a particular domain. A methodology for use of ontologies is developed in section 4.7 - ‘Building a System for Ontology and Semantic Web based Modelling’.

3.6.1 Semantic Web and Web 2.0 Collaboration

Section 3.5.2 - ‘Web and Interoperability Standards’ examined how an interoperable infrastructure could be provided for web based collaboration. Anderson (2007) in a Joint Information Systems Committee (JISC) report explains how Semantic Web and Web 2.0 are related, as Berners-Lee’s intention in the early development of Semantic Web technologies was for pages to be interactive. Anderson’s JISC report talks of Web 2.0 trends towards the “End of the software release cycle, Lightweight programming models, Software above the level of a single device, and Rich user experiences.” Sections 4.5 and 4.6 illustrate how such development techniques are used to create the prototype system for this thesis.

This approach of using Semantic Web and Web 2.0 techniques for end-user programming/modelling is researched by Leaver (2008), and demonstrated in a project created by Leaver, and called Bitriple. Bitriple aims to enable end-user functionality for Semantic Web/Web 2.0 style web-based ontology construction and search. This application added a facility to edit an ontology/ies, and instances based on the ontology/ies, and progress towards development of applications and querying of ontologies. Leaver’s (2008) application was also used within this thesis in order to create an online wing ontology and demonstrate searching of this ontology. Having tested the Bitriple ontology creation capabilities it also became obvious that the ontologies already created in Protégé and translated for Vanguard System could also be translated to run within the Bitriple web-based application. Bitriple uses open standard semantic representations in RDF/XML, a language already used extensively for the stepped translation within this thesis. The Bitriple application illustrated that using Semantic Web and Web 2.0

technologies in combination to enable end-user modelling/programming is possible. Large amounts of Semantic Web information could be created and stored quickly using this application. Therefore, this indicates that an aim of enabling creation of Semantic Web information, with a limited requirement to understand how it is stored, has been met. So to users, Bitriple is a straightforward web application and could be used without the need to learn about Semantic Web structuring. This application enables creation of Semantic Web information that then could be an incentive for creation or linking of other Semantic Web applications, and generate more opportunities for end-user programming/modelling. A screenshot from the Bitriple application, of ontology creation for an aircraft wing, is shown in Figure 38 section 6.4.4.1.

The ontology development problem should be aided by publishing the ontology and allowing tagging of content by users; the advantages of this in creating a shared understanding of what things mean is explained by Anderson (2007). Anderson explains that tagging by web users can generate some understanding and agreement about terms and an improved search facility, even without a formal ontology, or as a way to assist in the development and improvement of an ontology. Anderson’s JISC report explains how the technologies used are enabling user-centred web applications, and the use of the web as a development platform. The report advocates that “As a Web 2.0 concept, this idea of opening up goes beyond the open source software idea of opening up code to developers, to opening up content production to all users and exposing data for re-use and combination”. Anderson (2007) establishes the need for communities to build ontologies. Software applications are needed that allow users with little software knowledge to edit and update ontologies themselves. Anderson talks of ‘harnessing collective intelligence’ by means of interactive collaborative software, he calls this ‘distributed human intelligence’. Cayzer (2004) argues for provision of mechanisms to allow web page creators to tag their pages easily and as a natural part of the page creation. Cayzer also argues for Semantic Web based extensions to improve structuring of user-generated information to aid inferencing. Al-Khalifa and Davis (2006) and Schmitz (2006) use this approach of user tagging combined with centralised ontology development. These are the reasons for translating and uploading models to the web for this thesis. The plan for applying this research is described in section 4.8 - ‘Enabling User Driven Modelling/Programming’.

3.6.2 Central Ontology Management

There is a need for uniting of the approaches of top down ontology definition with that of the bottom up approach of allowing all users to define the ontology. Naeve (2005) discusses a bottom up approach where there is a set process of deciding what can be agreed on, what cannot, and on documenting both. Hunter (2002) explains how taxonomies can be the basis of the definitions for an ontology, and that commercial software is available. Hunter puts forward examples of the Ministry of Defense technology taxonomy, and the Boeing online ontology. Hunter explains the necessity of clear communication amongst engineers, and advocates ontologies/taxonomies for this purpose. Hunter gives examples of Airbus and Rolls-Royce need for ontologies/taxonomies, in order to solve problems from the need for compliance with strict rules and engineering tolerances for components going into complex structures such as aircraft, which have multiple suppliers. Zhao and Liu (2008) examine the need for sharing product information between partners as a product model, and how agreement through ontologies, Semantic Web, and standards can assist this. Ciocoiu et al. (2000) explain that taxonomies/ontologies of manufacturing concepts and terms can assist with avoiding misunderstandings and can aid communication. McGuinness (2003), and Uschold and Gruninger (2004) provide useful guides on how ontologies can assist in linking distributed data. Knowledge based systems need to allow a variety of people in different disciplines to share knowledge across functional, departmental, and interdisciplinary boundaries. Consideration is needed of the further problem that certain knowledge should be shared with others outside the organisation such as suppliers, and customers.

This ontology based approach has been used in the DATUM project (Scanlan et al., 2006), for representation of the information held in a database and process-planning tool used by Rolls-Royce aerospace, and this solution has been implemented at Rolls-Royce. This involved automatically producing a tree/graph based representation of information requested by the user for the DATUM project. This information was then visualised in decision support software (Vanguard System). A program written within Vanguard System for this thesis questioned the user via dialog boxes and produced tree based representation of the information, as selected by the user. This tree was colour coded, to show the categories of information produced, and enable navigation of this knowledge. This information could also be output as XML and other structured formats, and linked to stylesheets to create a Web tree-based representation.

In Figure 12, clicking the page option of the chosen material 'AMS4127 LAH AL6061 Forging and Rolled or Rings' shows the sub-tree below. This is the representation of the information relating to this material grouped into categories to represent groups of attributes. These attributes are colour coded according to the meaning of the attribute list with the top 2 representing the superclasses of this material and the 3rd and 4th group representing the material attributes and material costs. The use of tooltip comments provides an easy way to view the list.

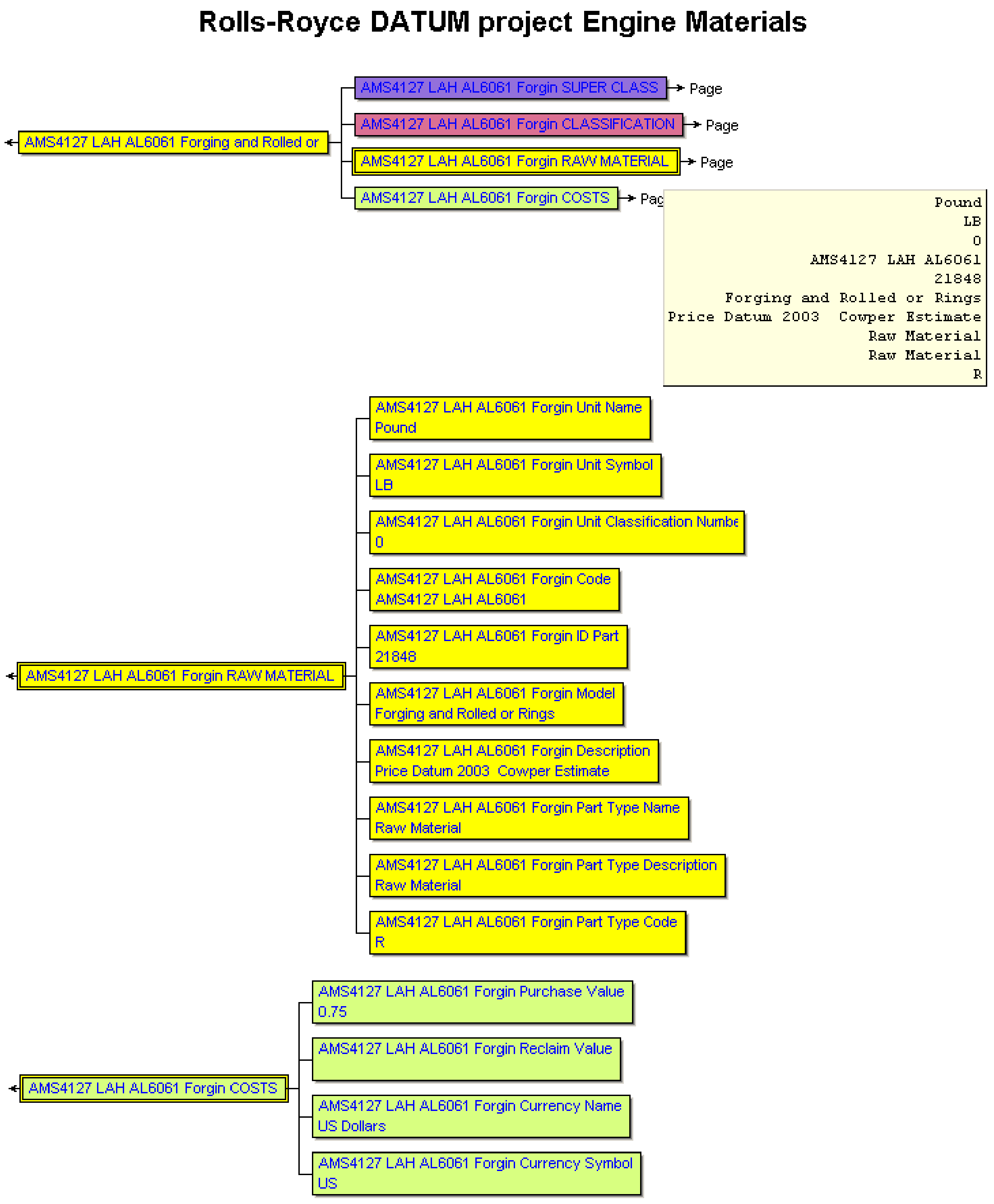

Rolls-Royce DATUM project Engine Materials
AMS4127 LAH AL6061 Forging and Rolled or
AMS4127 LAH AL6061 Forgin SUPER CLASS
Page
AMS4127 LAH AL6061 Forgin CLASSIFICATION
Page
AMS4127 LAH AL6061 Forgin RAW MATERIAL
Page
AMS4127 LAH AL6061 Forgin COSTS
Pag
Pound
LB
0
AMS4127 LAH AL6061
21848
Forging and Rolled or Rings
Price Datum 2003 Cowper Estimate
Raw Material
Raw Material
R
AMS4127 LAH AL6061 Forgin RAW MATERIAL
AMS4127 LAH AL6061 Forgin Unit Name Pound
AMS4127 LAH AL6061 Forgin Unit Symbol LB
AMS4127 LAH AL6061 Forgin Unit Classification Numbe 0
AMS4127 LAH AL6061 Forgin Code AMS4127 LAH AL6061
AMS4127 LAH AL6061 Forgin ID Part 21848
AMS4127 LAH AL6061 Forgin Model Forging and Rolled or Rings
AMS4127 LAH AL6061 Forgin Description Price Datum 2003 Cowper Estimate
AMS4127 LAH AL6061 Forgin Part Type Name Raw Material
AMS4127 LAH AL6061 Forgin Part Type Description Raw Material
AMS4127 LAH AL6061 Forgin Part Type Code R
AMS4127 LAH AL6061 Forgin COSTS
AMS4127 LAH AL6061 Forgin Purchase Value 0.75
AMS4127 LAH AL6061 Forgin Reclaim Value
AMS4127 LAH AL6061 Forgin Currency Name US Dollars
AMS4127 LAH AL6061 Forgin Currency Symbol US

Figure 12. Visualising and Interacting with database/ontology information

Further development of this kind of prototype is explained in section 6.5.2 - ‘Step 2 - Modelling Tool’. Cheung et al. (2007) used a similar approach of using ontology and XML based languages to communicate between different tools; this was described in section 2.4 - ‘Interaction for Modelling with Ontologies’.

Each time the user makes a request or a decision this causes the production of a tree or branch to represent this. A trigger is passed around the tree or branch as it is created. This is how the tree was read and constructed for the Rolls-Royce database application link, and can be made to work with any tree-based data structure. The user interface to enable this is connected to and reads from the ontology.

The ontology is held in a relational database. It would have been easier to construct a new faceted ontology to classify each item under one or more headings and structure this in an open ontology.

However there was insufficient time in this project to translate the Rolls-Royce database to this ontology; so in later prototypes Airbus wingbox (wing structure) information is used instead. The collection of cost knowledge and creation of cost objects represents a major bottleneck. One of the critical factors concerning the speed with which this happens is the complexity of software required to code the knowledge. The advantage of the system outlined in this thesis is that knowledge can be collected and interacted with visually by end-users, and without the requirement to know a programming language. If specialist knowledge or deep software skills are required, this has a severe impact on the rate at which cost knowledge is formalized and deployed. The use of Lisp, Java, or other similar programming languages requires a significant level of training. Competence in such languages typically takes years of experience (Scanlan et al., 2006). Where languages such as these were translated to and used within this thesis, they were hidden behind a visual taxonomy layer, which allows the user to interact with the information without writing code. Only programmers would want to edit the code as text. A second aspect of the solution to the information gathering problem is to hold information in open standard representations, and share this amongst interoperable systems so information can be represented just once and translated when necessary. This was discussed in section 3.5 - 'Ontologies for Interoperability and Reuse of Models'. Once information is visualised and can be interacted with as above, the next stage is to make it possible to model/program with it.

3.7 Meta-Programming as a Model Creation Technique

Model-Driven Programming and Meta-Programming together with Semantic Web and End-User Programming techniques are vital ingredients of the User Driven Programming/Modelling approach used in this thesis. The use of Meta-Programming in the User Driven Modelling/Programming approach is described in 4.8. This section (3.7) builds on the research covered in section 2.6 - 'Model-Driven Programming for Better Model Production'. Dmitriev (2006) explains the problem to be solved in order to improve model production as "limitations of programming which force the programmer to think like the computer rather than having the computer think more like the programmer." Meta-programming (Dmitriev, 2006) is a useful way of allowing for language independent software development, and can aid in providing a high level front-end to programming languages. "Meta-programming is the writing of programs that write or manipulate other programs (or themselves) as their data" (Wikipedia, 2009). Fischer's (2007) examination of the overall meta-design concept was described in section 2.5 - 'Visualisation and Interaction'. The idea behind use of meta-programming in this thesis is that instead of writing programs to do a task a domain expert needs the program for, the Meta program developer creates an environment which all domain experts, in this and similar fields can use to create their own solutions. The developer then only needs to maintain and improve this programming environment, and can concentrate on this task; the domain expert can concentrate on solving the problem at hand without having to ask the developer to create the code on his or her behalf.

Dmitriev (2006) advocates reducing dependency on languages and environments by enabling programmers to develop their own specific languages for solving each domain problem:

“If we are going to make creating languages easy, we need to separate the representation and storage of the program from the program itself. We should store programs directly as a structured graph, since this allows us to make any extensions we like to the language. Sometimes, we wouldn’t even need to consider text storage at all. A good example of this today is an Excel spreadsheet. Ninety-nine percent of people don’t need to deal with the stored format at all, and there are always import and export features when the issue comes up. The only real reason we use text today is because we don’t have any better editors than text editors. But we can change this... Text editors... don’t know how to work with the underlying graph structure of programs. But with the right tools, the editor could work directly with the graph structure, and give us freedom to use any visual representation we like in the editor. We could render the program as text, tables, diagrams, trees, or anything else. We could even use different representations for different purposes, e.g. a graphical representation for viewing, and a textual representation for editing. We could use domain specific representations for different parts of the code, e.g. graphical math symbols for math formulas, graphic charts for charts, rows and columns for spreadsheets, etc. We could use the most appropriate representation for the problem domain, which might be text, but is not limited to text. The best representation depends on how we think about the problem domain. This flexibility of representation would also enable us to make our editors more powerful than ever, since different representations could have specialized ways to edit them.”

This visual tree/graph diagrammatic approach provides a way to create programs that create programs so enabling the 3 step translation process used in this thesis, and this enables translations between people, between systems, and between languages. This translation could enable those who are not currently programmers to create models at their domain level using domain specific systems created for them by programmers, and with visual management of formulae and versions. The mechanisms for this are recursive translation of the tree/graph code representation to multiple models and languages, where necessary aided by user/modeller choices, as demonstrated in chapter 6.

3.7.1 Model-Driven Programming

Spahn et al. (2007) explain that end-users are domain experts not IT professionals, and because they cannot program their own solution, this is requiring them to communicate their needs to IT developers. Spahn et al. argue for the empowerment of users to customise software by providing an abstraction layer to hide technical details and allow for concentrating on business needs. Model-Driven Programming and the Semantic Web are explained by Frankel et al. (2004), and they discuss how these techniques can be combined (section 4.2.1). Bringing together model-driven programming with the Semantic Web can enable diagrammatic programming, and translation to structured and searchable Semantic Web output, this eases visualisation and interaction problems at each stage of translation. Visualisation and interaction with ontologies was examined in section 3.3 and is important for model-driven programming research in this thesis. Frankel et al. investigate translation of UML and entity relationship diagrams that use graphical notations and store in formats such as XMI[19] [20] (Example visualisation in appendix), and translations into OWL.

As models and ontologies in this research are to be cohesive and closely linked, visualisation and interaction techniques can be applied to either or both. Model-Driven Programming can be an important technique for dealing with complexity. Gray et al. (2004) explain how this technique can assist in the development of software for a large avionics system, and also investigates program and model transformation. Gray et al. (2004, 361) explain Program Transformation/Translation for Model-Driven Programming:

"Research into horizontal transformation concerns modification of a software artifact at the same abstraction level. This is the typical connotation when one thinks of the term *transformation*, with examples being code refactoring at the implementation level, and model transformation and aspect weaving at a higher design level. Horizontal transformation systems often lead to invasive composition of the software artifact. In contrast, vertical transformation is typically more appropriately called *translation* (or synthesis) because a new artifact is being synthesized from a description at a different abstraction level (e.g., model-driven software synthesis and reverse engineering). Vertical translations often are more generative in nature."

Gray et al. (2004, 367) also investigate "interpreters that traverse the internal representation of the model and generate new artifacts", e.g. XML files and source code, though their main emphasis is on transformation of existing code. Gray et al. explain that a model/developer does not create the transformation/translation rules. For this thesis this role is assigned to the 'System Creator' who creates the translation system for the 'model builders' and 'model users', these roles were examined in section 3.3.1 - 'Users and De-abstraction for Translation'. This thesis concentrates on vertical generative translations for creation of new artefacts, in order to assist end-user programming/modelling.

Coutaz (2007, 2) explains how Model Driven Engineering and Service Oriented Architecture can be combined. Coutaz also explains that "An interactive system is a graph of models related by mappings and transformations." This would fit in well with the structure of RDF, which is also a graph structure, an interactive editable tree/graph of models could be produced, in order to relate models and sub models, and ontologies, and sub ontologies all to each other. Baclawski et al. (2001) and Kogut et al. (2002) explain how UML can be used as a tool to produce ontologies. Kogut et al. make the point that UML was originally designed for human-human communication, but is being driven to become more formal and 'machine-processable' and is now being used to generate code and schemas. Baclawski et al. translate from UML diagrams to ontologies represented in graph based languages; thus demonstrating the use of translation to aid diagrammatic visualisation and editing for ontology creation, and enabling closing of gaps between UML and ontology languages and modelling. Section 4.2.1

[19] OMG (Object Management Group) (2011) *MOF 2.0 / XMI Mapping Specification, v2.1.1* [online]. Available from: http://www.omg.org/spec/XMI/ [Accessed 14 June 2011].

[20] Hale P. (2011) *XMI (XML Metadata Interchange)* [online]. Available from: http://www.cems.uwe.ac.uk/amrc/seeds/Modelling.htm#XMI [Accessed 14 June 2011].

examines ways UML tools could facilitate translation, interaction, and visualisation for ease of use. UML tools are beginning now to use and translate to and from Semantic Web languages; indicating a step towards co-operation and merging of ontologies and modelling.

3.8 Conclusions

The overlapping fields investigated for the theory are shown below :-

- Information Structuring and Navigation
- Visualisation and Interaction for Modelling
- Ontology Based Modelling Solutions
- Ontologies for Interoperability and Reuse of Models
- Ontologies and Semantic Web technology and their role in Modelling
- Meta-Programming as a Model Creation Technique

A methodology for using these different strands of theory that overlap, and for creation of a tool to fill the gap in the area of research where all these overlap is shown in 4.1 Creation of Modelling System - Figure 13.

In the examination of the theory for chapter 3, a conclusion was that this approach is suitable for tree/graph based problems, and for modelling where nodes can be linked by formulae. This does not make it suitable for all engineering modelling. For example fluid dynamics, and finite element analysis are different types of modelling. The theory is suitable for process modelling (both engineering and business), and for representing taxonomies. So although the research was aimed towards engineering, the focus is not on all engineering, and can include other fields.

The examination of the theory behind User Driven Programming/Modelling indicated that the hypothesis needed revising to - it is possible to create an End-User Programming environment, usable by computer literate non-programmers, which can have a wide variety of uses especially for tree/graph/network based modelling including process modelling, and representation of taxonomies.

Chapter 4 explains the research design and methodology based on this User Driven Modelling/Programming theory. Section 4.4.1 - ‘Improving and Building on Modelling Capability and End-User Interaction’ examines use of translation for enabling model building and use. Chapter 5 prototypes this methodology and chapter 6 implements it.

Chapter 4 - Research Design: Methodology

4.1 Creation of Modelling System

The methodology outlined here involves reflection and refinement, related to, and in step to match research practice to and with the software development cycle for implementation. The methodology needs to meet the modelling/programming objectives outlined in Chapter 1, of Maintenance, Extensibility, Ease of Use, and Sharing of Information. The intention of the research into User Driven Modelling (UDM), and more widely User Driven Programming (UDP), is to enable non-programmers to create software from a user interface that allows them to model a particular problem or scenario. This involves users entering information visually via a diagram. The research involves developing ways of automatically translating this information into model/program code in a variety of computer languages. To achieve this, visual editors are used to create and edit taxonomies to be translated into code. To make this possible, it is also important to examine visualisation, to create a human computer interface that allows non-experts to create software. This methodology is examined throughout this thesis against the Maintenance, Extensibility, Ease of Use, and Sharing of Information criteria.

There are many computer literate people who do not have the time to learn, or access programming tools, but nevertheless try to accomplish programming type tasks (Scaffidi, 2005); so instead, many of them model problems using spreadsheets (Scanlan et al., 2006). This thesis research examines an aspect of end-user programming - User Driven Modelling/Programming, addressed towards the kind of computer literate end-user programmer/modeller just mentioned. This takes further the end-user programming aspect of spreadsheets and similar tools. To enable easier manipulation of the kind of complex information that is often held and managed in spreadsheets, an approach based on diagrammatic visualisation of the model is employed to enable navigation and communication of models. This allows navigation of the equations that represent a model by following a ‘family tree’ of relationships between equations, and between models. This makes collaboration easier by ensuring people can navigate and interpret models created by others. Gruber (1993b) observes that a further advantage is that the underlying structure on which models are constructed can be represented and stored using open standard representations, to enable its availability for collaboration. In this way equations (formulae) can be stored in an ontology, visualised for ease of understanding, and made available for calculation in models.

The methodology involves visualising connections between individual calculations and allowing results from calculations to connect and link to form an overall model. Therefore, this methodology could be generalised to any situations where calculations connect to form a model. Even where calculations are not represented but information needs to be visualised (e.g. scientific taxonomies, engineering product data structures); the same visualisation techniques can be applied, to ease navigation and therefore collaboration. Complex models can be made more understandable when displayed in diagrammatic form. Nurminen et al. (2003) evaluate a system called NICAD that uses calculation rules in this manner. Nurminen et al. emphasize that what successful expert systems have in common is that they put user needs at the centre of a fast and agile development process. The authors explain that users prefer usability over automation, and that users should drive the more difficult tasks where they are needed and leave routine tasks to the system. Ko (2007) explains that end-users' goals relate to the problem domain not to code production so they should be allowed to focus on their goals; therefore it is important to visualise the whole program execution not just the output.

The intention is to demonstrate a way to construct diagrammatic representations of cost using the example of an aircraft wingbox. The wingbox is the structure or skeleton of the wing. These diagrammatic representations will be achieved by visual representation of items and equations that make up wingbox cost. These items and equations can be represented in standardised categories used in engineering - 'materials', 'processes', 'cost rates' etc. These categories are standard for engineering and the methods for representing items and equations that relate the items can be expressed in standard mathematical form. Therefore using the same methodology and same categories it would be possible to represent other items and equations in the same way. So this methodology is reusable for costing other engineering components including those outside aerospace. The costing method is also recursive because components and sub components can be costed separately or together and top down or from bottom up. This methodology has the potential to be applied to any calculation based modelling problem.

The User Driven Programming approach advocated in this thesis has the advantages that it is using a modelling approach for creating modelling solutions and involves creating systems to create systems. This makes it possible to solve the problem by breaking it down into stages and allowing software developers to concentrate on the most complex software problems and domain experts to be able to concentrate on their domain problem. The standardisation possible in this approach can allow software developers to create modelling systems for generic purposes that can be customised and developed by domain experts to model their domain. This methodology can be facilitated by :-

- Modelling Tools - Building an end-user interface and extending the translation capabilities of UML (Unified Modelling Language) and/or other modelling tools (Johnson, 2004).
- Spreadsheets - Improving the structuring and collaboration capabilities of spreadsheets, and enabling customisation of spreadsheet templates for particular domains and users.
- Ontology Tools - Extending the modelling capabilities and equation calculations in ontology tools and providing an end-user interface.
- Semantic Web/Web 2.0 - Extending the capabilities of Semantic Web and Web 2.0 style web-based development tools to allow collaborative modelling.

These possible solutions are not mutually exclusive and their combination could be the best way of providing usable collaborative modelling tools for computer literate end-users and domain experts. The link between these alternative ways of advancing current research is translation and User Driven Modelling/Programming.

Figure 13 shows the solutions, and how these could make User Driven Modelling/Programming possible :-

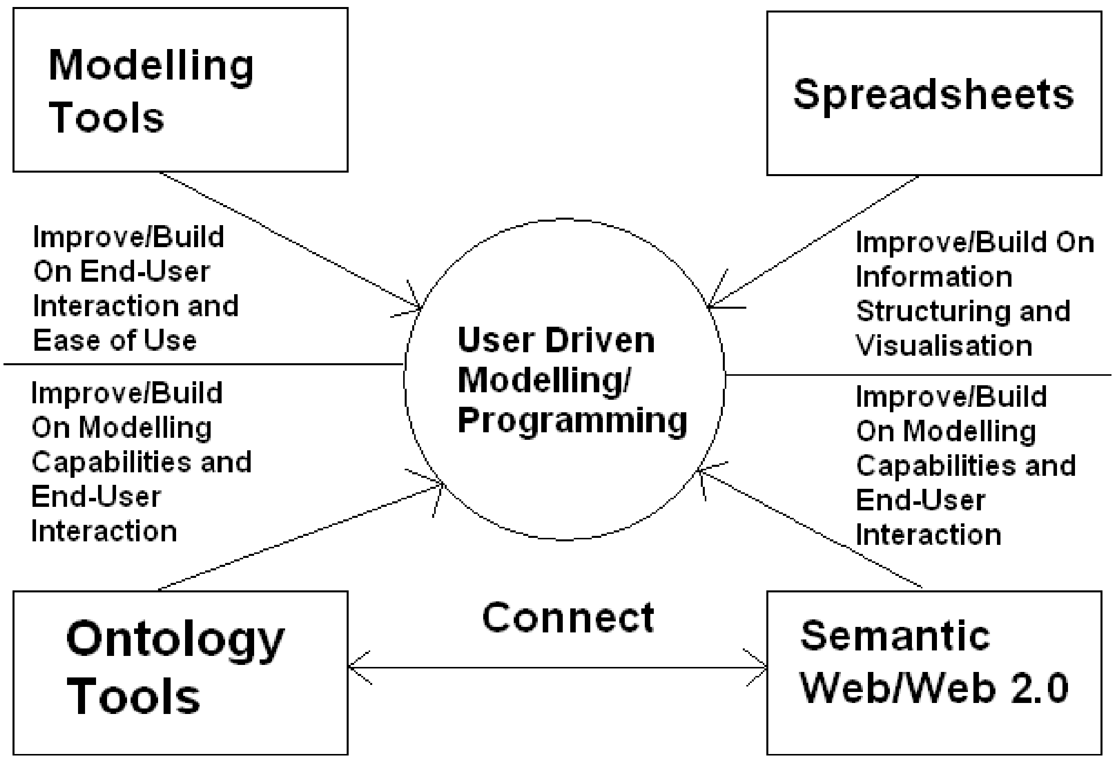

Modelling Tools
Spreadsheets
Improve/Build On End-User Interaction and Ease of Use
Improve/Build On Information Structuring and Visualisation
User Driven Modelling/ Programming
Improve/Build On Modelling Capabilities and End-User Interaction
Improve/Build On Modelling Capabilities and End-User Interaction
Ontology Tools
Connect
Semantic Web/Web 2.0

Figure 13. Methodology Diagram - Enabling User Driven Modelling/Programming

This methodology is developed from the overlapping research examined for Chapter 2, the literature review, and the theory based on the overlapping theoretical work developed in Chapter 3. The circle in the middle - User Driven Modelling/Programming is the overlap, and therefore is the focus of this thesis, and that which needs to be implemented to enable end-users to create tree/graph/network based models for process modelling.

4.2 Modelling Tools

4.2.1 Improving and Building On End-User Interaction and Ease of Use

This section examines the way software developers could make use of conceptual modelling type solutions to enable users to program their own modelling solutions. Examples of such solutions are ontology/taxonomy tools, UML (Unified Modelling Language), Scientific/Engineering Modelling tools, and Business and Process Modelling tools. It is important to develop a user interface layer to such tools that can enable development by non-programmer domain experts, by providing to these domain experts clear visualisation and interaction with their domain that can be translated into the most relevant UML, process, business, and/or scientific models as appropriate. McGuinness (2003) advocates conceptual modelling to provide ease of use for end-users, “conceptual modeling support, graphical browsing tools, etc. all may be important.” In section 3.6 - ‘Ontologies and Semantic Web and their role in Modelling’, McGuinness examined how ontology tools can assist with such end-user conceptual modelling.

De Souza (2007) argues that the goal of human-computer interaction (HCI) will evolve from making systems easy to use to making systems that are easy to develop. Building an end-user interface and extending the translation capabilities of tools such as UML (Unified Modelling Language) and/or engineering and scientific modelling tools could make them more usable by end-users. UML tools are used mainly by software developers, and engineering/scientific modelling tools are mainly used by domain experts who have software skills.

There are important gaps in the functionality of UML tools for user centred design. Palanque and Bastide (2003) identify these gaps, “For the team of methodologists (Rumbaugh, Jacobson, Booch) that shaped the UML, User Centred Design was not a central concern.” These gaps are of even greater importance when attempting to make it possible for people who are not programmers to create software. Johnson (2004) ascertains that UML tools need to be extended to better enable modelling of collaborative tasks. In section 3.3 - ‘Visualisation and Interaction for Modelling’, Repenning’s (2007) examination of how enhancements to UML could aid end-user programming was discussed. Engels (2007) also explains that UML should be extended to allow development of user interfaces in order to assist end-users to program. UML tools could also assist software developers in creating a modelling environment suitable for domain experts to use to solve their problems. Achieving these aims would require a major change in UML tools to enable modelling of user interaction as the core concern. For example Abraham and Erwig (2007) integrate spreadsheet modelling into the UML modelling process. Enabling users themselves to create software using UML type tools would require development of a new type of UML tool specifically designed for users. This would be compact and simple, but provide enough capabilities to ensure user’s designs are robust. This would also fill a gap left by engineering and scientific modelling tools, which are powerful but do not have collaboration, communication, and ease of use as central concerns. Frankel et al. (2004) explain this, and comment on collaboration and technologies for co-operative systems that combine UML, ontology environments, and software environments into an overall system or Model Driven Architecture. Vernazza (2007) demonstrates Model Driven programming with a Semantic User Interface, as discussed in section 3.3. It should be possible to target this research towards engineers and scientists who are well used to building models from equations; economists and financial analysts are also used to such techniques.

Examples in chapter 6 demonstrate the use of modelling tools e.g. Vanguard System linked with an ontology tool Protégé through User Driven Modelling/Programming translation techniques.

4.3 Spreadsheets

4.3.1 Improving and Building on Information Structuring and Visualisation

Improving the structuring and visualisation capabilities of spreadsheets is necessary for improving their collaboration capabilities. This could enable better customisation of spreadsheet templates for particular domains and users.

Users can specify a calculation in mathematical terms using a formula. The spreadsheet then calculates the result of the formula. Users can change the formula if it is incorrect without any need to write code or re-compile. This accounts for the popularity of spreadsheets. Ko (2007) explains the problem of programs which were intended to be temporary and owned by a particular person becoming central to a company; this often happens with spreadsheets. So it is necessary to create collaborative tools that allow users to develop software in a way they will be familiar with from their use of spreadsheets. Section 5.1.2 gives a critical evaluation of a spreadsheet created as part of an Airbus aerospace project; this gives a practical illustration of the problems that need to be tackled. Burnett et al. (2007) state, “end-users are using various languages and programming systems to create software in forms such as spreadsheets, dynamic web applications, and scientific simulations. This software needs to be sufficiently dependable, but substantial evidence suggests that it is not.” Creation of correct spreadsheets is difficult when the structure of the relationships in the spreadsheet is not clearly visible; this is why it is important to develop clear representations and visualisations. Crapo et al. (2002) argue that many users of spreadsheet software can model problems accurately if they are provided with visualisation capabilities to help them build, track, and understand the information relationships, and therefore the problem modelled. Crapo explains that visualisation helps users maintain a hierarchy of sub models at different stages of development and to navigate effectively between them. The way spreadsheets are defined by formulae can aid in allowing editing and tracking of information in models, provided that the relationships between formulae are adequately represented. Hanna (2005) explains that a spreadsheet program is defined by formulae and has purely declarative semantics with the order of evaluation determined purely by the dependencies between cells. However, Hanna criticises the impoverished semantics of spreadsheets that “severely limit the ability of programmers (even expert ones) to construct reliable, correct, maintainable programs with well known consequences”. Practical problems resulting from these semantic representation deficiencies are explored in section 5.1.1. One possible solution, demonstrated by Erwig et al. (2006) is automatic generation of correct spreadsheets, and solving of errors of meaning (semantic errors); though in this thesis the main concentration is on visualising the semantics and structure of models/programs.

This thesis concentrates on visualisation in order to make the meaning clearer to the human modellers. Therefore, it is necessary to improve the information structuring, and visualisation capabilities of spreadsheets in order to encourage more reliable modelling and collaboration. Spreadsheets are a useful application of the technique of ‘Programming by Example (or Demonstration)’; this is used in the

‘Record Macro’ functionality (also discussed in section 5.2). Programming by Example is expanded on by Cypher et al. (1993) and Lieberman (2000).

4.4 Ontology Tools

4.4.1 Improving and Building on Modelling Capability and End-User Interaction

Ontology tools can be made more usable for User Driven Modelling/Programming by extending the modelling capabilities and equation calculations, and providing an end-user interface. Ontology tools can be linked also with modelling tools and spreadsheets through implementation of a User Driven Modelling/Programming translator (explained in Section 4.6 - System to be Developed). Ontology tools can be linked through this translation mechanism to client Semantic Web/Web 2.0 tools or directly to such tools on the server to provide facilities such as improved visualisation and editing. This was illustrated in Figure 13 section 4.1.

Information is scattered within organisations and often not held in such a structured way as to be easily accessed by employees or software. Reasons behind this situation were discussed in section 3.6 - ‘Ontologies and Semantic Web and their role in Modelling’. This problem is examined by Lau et al. (2005) using the example of McDonnell Douglas (now part of Boeing), that demonstrated how difficult it is to gather unstructured knowledge. Therefore, it is important that research is undertaken into methods of capturing, structuring, distributing, analysing, and visualising information. Even where documents are represented using XML or other structured languages, it is important to structure the contents and semantics using an ontology, Erdmann and Studer (1999) experiment with querying ontologies made up of XML documents. If a usable ontology is created, or brought together, it is then possible to automate the building of models from the ontology.

Macías and Castells (2004) use the approach of defining model based user interfaces using an ontology. This approach was also used in this thesis, and the ontology defines both the user interface, and the software model. Formulae (equations) are represented in the ontology and made available to modelling systems, and examples created demonstrate how this information can be directly translated to software or meta-language software to represent and run the models (chapter 6). Shim et al. (2002) examine translation from a user’s model to equations and explain that “converting a decision-makers’ specification of a decision problem into an algebraic form and then into a form understandable by an algorithm is a key step in the use of a model.”

This thesis involves automatically producing software for a Semantic website from visual representations of the problem. The core of this modelling infrastructure is automated generation of models created with World Wide Web Consortium (W3C) and other standards based languages, and the visualisation of information represented in such W3C standard ways. Modelling languages such as Alloy, explained by Jackson (2006), could be developed as an interface to the ontology from which models are built. Transformation from a model building environment to program code has been investigated by Gray et al. (2004) and was examined in section 3.7.1 - ‘Model-Driven Programming’. Experienced programmers could build a modelling environment that could then be used by non programmers to create models or solve other software problems. Hanna (2005) uses this approach and makes use of a declarative functional language Haskell (Hudak et al., 2007) to build user environments, (discussed in section 3.3). MathML and semantics built on this could assist in this process by providing an open representation of functions as XML (eXtensible Markup Language). Functions entered by the model developer can then be translated to this open representation and translated to programming languages and/or read by programming languages. The representation of functions and information can usually be illustrated diagrammatically. This is why it is important to provide translation capabilities between different representations of modelling problems to visualise them in the context the user expects.

4.5 Semantic Web and Web 2.0

4.5.1 Improving and Building on Modelling Capability and End-User Interaction

Improving and building on modelling capabilities and end-user interaction can be achieved by extending the capabilities of Semantic Web and Web 2.0 style web-based development tools to allow collaborative modelling. Semantic Web/Web 2.0 tools can be made more usable for User Driven Modelling/Programming by extending the modelling capabilities and equation calculations and providing an end-user interface. Semantic Web/Web 2.0 tools can be linked also with modelling tools and spreadsheets through implementation of a User Driven Modelling/Programming translator (as is also the case for ontology tools). As was illustrated in Figure 13, Semantic Web/Web 2.0 tools can be linked through this translation mechanism to Ontology tools or directly to Ontology tools on the server to provide improved structuring of information.

In this thesis, online spreadsheet tools were developed and extended to allow better collaboration and visualisation. During this part of the research, Google developed Google Spreadsheets[21], and other spreadsheet like Web 2.0 tools were also developed. While online spreadsheet type software created for this thesis and other tools such as developed by Google are a useful extension of spreadsheets onto the web, and they encourage collaboration, they still need further development and customisation in order to structure and visualise information to make collaboration effective when dealing with complex models. This would involve more implementation of other Web 2.0 ‘Mashups’ and interactive diagrammatic modelling. An advantage of web spreadsheets over traditional spreadsheets is that they enable easier customisation and development of their capabilities, and linking to ontology tools through open source languages. It is also then easier to link the spreadsheet to online visualisation and modelling tools, this is explained in chapter 5 on development of early examples and prototypes, and chapter 6 on development of the final implementation.

4.6 System to be Developed

The system explained here aims to solve problems of Maintenance, Extensibility, Ease of Use, and Sharing of Information by linking and advancing Modelling tools, Spreadsheets, Ontology tools and Semantic Web/Web 2.0 tools and techniques. The system outlined is an attempt to help users who are experts at their own careers and not computer professionals, to develop models that aid in their work. This is a common situation for designers and other engineers. The research and implementation is later demonstrated through the wingbox (wing structure) example, used in a spreadsheet in chapter 5. This illustrates the approach of provision of a system to enable users to create and/or use and share their own models.

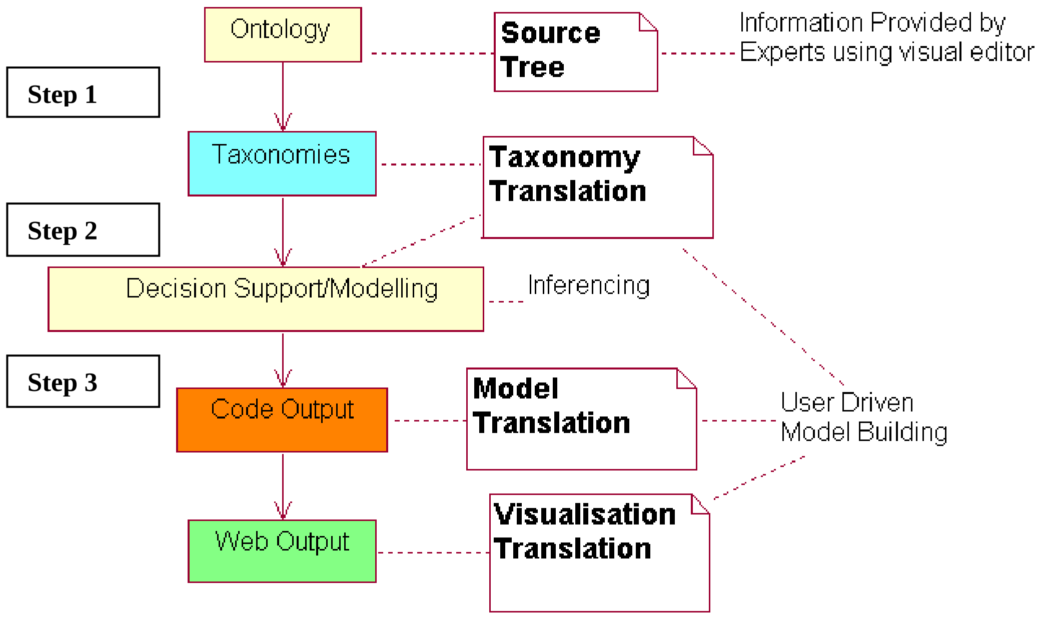

Ontology
Source Tree
Information Provided by Experts using visual editor
Step 1
Taxonomies
Taxonomy Translation
Step 2
Decision Support/Modelling
Inferencing
Step 3
Code Output
Model Translation
User Driven Model Building
Web Output
Visualisation Translation

Figure 14. Translation Process Chain

This research involves adapting or creating software systems to provide the visual editor for the source model/tree, and model builders would create a model by editing this. By doing so they would create a generic model for a particular modelling subject.

This translation process involves a stepped translation from a source tree, through visualisation, interaction and model calculation, to a result tree. The role of the ontology and taxonomies/sub ontologies for providing the source tree for this translation was explained in Figure 9 section 3.3.2. The result tree can be code and/or Semantic Web output. The result tree/model is a more specific subset of the source tree/model. This output is enabled by provision of translation software to translate the ontology into a decision support and modelling system. The model users could then use this decision support and modelling system to create their models. User choices specify what specific subset of the generic model will be produced to apply for their own analyses. This thesis concentrates on provision of a translation mechanism to convert information or models into many computer languages (primarily

[21] Google (2011) *Welcome to Google Docs* [online]. Available from: docs.google.com/ [Accessed 14 June 2011].

web-based), and to visualise this information. Examples of this are shown in chapters 5 and 6 based on wingbox cost modelling.

A model as defined by a user could be translated to a model that is more suitable for a computer to interpret. Software could then follow any relationships defined in the model, make any calculations or decisions, and so provide the results. Recursion can be used to enable the computer to follow a representation of the problem without having to care about the names of objects. This is particularly true for tree/graph representations. This process was described in section 3.3.1 and illustrated by Figure 10. Trees are defined recursively because their structure is recursive, so it is natural to traverse and copy them recursively. This approach deals with hierarchies and relationships, but for requirements outside this scope 'Aspect Oriented Programming' (Elrad et al., 2001) and (Murphy et al., 2001) could be used to capture and translate these requirements. Aspect Oriented Programming can be used to identify functions that can be used in calculating results without the requirement of a detailed underlying ontology/taxonomy (Elrad et al., 2001). So Aspect Oriented Programming is best used where software functions cannot be neatly attached to particular objects or nodes in a hierarchy. These are known as cross-cutting concerns as they may affect several nodes. A diagrammatic representation of the cross-cutting concerns could then be translated into a computer representation (as investigated by Gray et al. (2004)), but further research on this topic is outside the scope of this thesis, though the possibility was illustrated in Figure 8 section 3.3.

A further translation is necessary from the decision support program/model (step 2) to a result model (step 3) that should be created to express the results to a user. This model would be a categorized full description of all the results from the program. This results model should be represented using open standard information languages such as XML or languages derived from these. This enables the widest possible re-use of the information on different hardware and software systems, and translation up and down ontology layers. The result model could be represented diagrammatically or as categorized and linked web pages/applications. The full translation is as below :-

Source Model (Human Friendly Representation)➔ Source Model (Computer Friendly Representation)➔ Computer Program➔ Result Model (Human Friendly Representation)

If users can define the source model, remain largely unaware of how the result model is produced, can understand the result model, and it meet their expectations, the translation will be successful. Decisions a user makes can affect both the content and the presentation style of the results received.

The system created for this thesis consists of applications combined in order to represent a layered architecture of :-

Database➔ Ontology Engine➔ Ontology Visualiser➔ Calculation Engine ➔ Inputs Visualiser ➔ Results Visualiser

The ontology created for this thesis in Protégé has formal definitions of 'is-a' relationships, and formal instances. Therefore, it is much more machine readable than those at the human readable end of the ontology scale such as thesauri. In fact, it is tested as machine readable by the automated conversion process from the Protégé representation to the decision support modelling system used for calculations. The decision support system can be made to read any of the trees from Protégé and it can split or combine these trees or branches as needed for the model being used, and in response to decisions made by the user, as mentioned in section 2.4. In Section 2.4, it was mentioned that Miller and Baramidze (2005) argued for use of interrelated ontologies rather than one single ontology; this approach is used for this thesis. Development of these interrelated ontologies was described in section 3.3.1. The intention is to ensure models and ontologies are linked systematically and in a generic way. Logical constraints are informal, and failures are caught by the modelling tool that will flag any illogical expressions. It would be better to represent such logical constraints in the ontology also, as is done in the most rigorous machine readable ontologies, this was not finished due to time constraints and because it was not necessary for validation of the concept, so is to be future work (explained in section 8.2).

It is possible to create an extra layer of visualised semantics to enable users to specify commands in structured language. This approach of adding extra layers is the way this visual programming works. Users provide the information the program needs at the visual interface layer and program code is created automatically. The layers provide the bridge between abstract ideas and computer code. If this approach is taken to its logical conclusion, it would be possible to allow the user to specify what the computer should do. Then each layer would communicate this to the layer below until the computer performs the action required. A simple example of this approach is the use of spreadsheets. Users can specify a calculation in mathematical terms using a formula. The spreadsheet then calculates the result of the formula. Users can change the formula if it is incorrect without any need to write code or re-compile. This accounts for the popularity of spreadsheets. However, spreadsheets do not provide the centralised and structured data-store required for a distributed collaborative system. Therefore, this research concentrates on combining the wide applicability of generic spreadsheet modelling with structured and adaptable modelling and visualisation.

It is important to enable changes to the design of the information source and its structure as necessary, even when it contains information. This makes possible continuous improvement of the information and its representation together. Clear visualisation of the structure makes out of date and duplicate information obvious, so it can be changed by the end-users of the information. This provides for maintenance of information quality without necessitating end-users to understand relational database design; though relational databases can still be accessed by software specialists for more in depth and less frequent structural changes. Figure 15 shows the way iterative research development and refinement is used both in this research method and in the implementation to ensure that changes can be made systematically as necessary and without disrupting the projects.

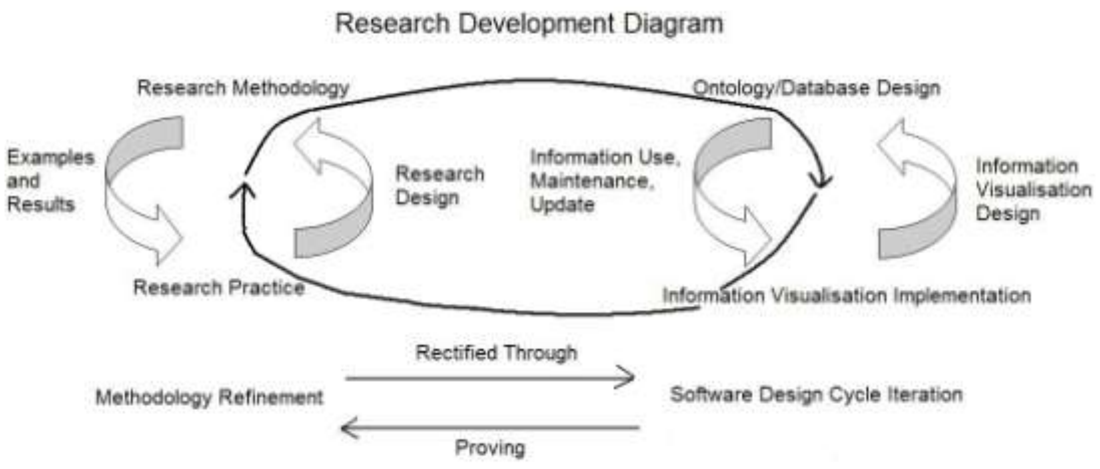

Research Development Diagram
Research Methodology
Ontology/Database Design
Examples and Results
Research Design
Information Use, Maintenance, Update
Information Visualisation Design
Research Practice
Information Visualisation Implementation
Rectified Through
Methodology Refinement
Software Design Cycle Iteration
Proving

Figure 15. Research Development Iterations

Each idea is implemented and tested continuously. There is reflection and refinement within the Methodology and Implementation, within the Software Design for demonstrating the Implementation, and between the Methodology and Implementation and the Software Design. Each iteration of development is a consequence of application of the research and methodology so far undertaken to the process of development. This process is described in chapters 5 and 6.

4.7 Building a System for Ontology and Semantic Web based Modelling

As mentioned in chapter 1, the use of the Semantic Web in this thesis is a means for open standard representation of information, transformation into different representations as required, and for provision of a high level interface as a tool for model creation, and translation to program code. An ‘elaborator’, is used, this is a translator that converts the diagrammatic representation of the problem into software. Translations could be performed into any programming or meta-programming language or open standard information representation language, the visualisation of the model created can be displayed on the web. This translation builds on research in program and model transformation. The translation software performs transformations as required between different programming languages and visual model views. This is prototyped for this thesis, and it is important to further this research in order to establish a user base, and make the translation generic.

The focus is on combining the development of dynamic software created in response to user actions, with object oriented, rule based and Semantic Web techniques. This helps solve problems of ‘impedance mismatch’ of data structure between object oriented and relational database systems identified by Ambler (2003). The information for the examples is highly structured and visualisation of this structure in order to represent the relationship between things clarifies the semantics. The meaning can be seen not just by the name of each item but also by the relationship of other items to it. For this thesis an ontology is used that consists of several related taxonomies/sub ontologies. This ontology provides a design costing capability, but the ontology and the techniques used to build it could be re-used for other purposes. In fact the model demonstrated in chapters 5 and 6 needs to be restricted from generic to wingbox costing specific, as it is constructed and used the model and tree become more specific as information is added. Therefore the core of this solution is generic, other problems could be modelled in just the same way, over time going from generic to specific as the model/tree is specified. The use and visualisation of the ontology enables editing of an ontology centric model without the need to edit code. A light-weight ontology is used for the thesis, and this is evaluated for usefulness before deciding whether it would need to be more structured. McGuinness (2003) mentions that “building the more complicated ontologies may be cost prohibitive for certain applications.” Issues about visualisation of light weight ontologies are examined by Fluit et al. (2003). Fluit et al. examine scalability of ontologies, and visualisation, they recommend visualising ontology schemas and instances in a related coherent way. Hunter (2002) evaluates engineering ontologies and gives examples (a more expansive discussion is in section 3.6.2). An important reason for creating an open standard central ontology is that it can be accessed by many different applications. Research of others in this field assists with interoperability of ontologies (Corcho and Gómez-Pérez, 2000) compare and contrast ‘traditional’ and web-based ontology languages, and (Corcho et al., 2003) examine ontology tools and techniques, as discussed in section 3.4. Noy and McGuinness (2004) describe ontologies by means of examples, and outline a methodology for their systematic construction. The open standard OWL (Web Ontology Language) is used for this thesis; the role of OWL was explained in sections 2.3, 3.3, and 3.4. Bechhofer and Carroll (2004) explain the three classes of OWL documents, and the sublanguages that define them - Lite, DL, and Full. These languages use logic representation increasingly from Lite towards full, so the benefits of simplicity need to be balanced with what is necessary for Semantic Collaboration, and modelling.

The detail and accuracy of the information that can be provided to define engineering products varies along the product development chain. The most important time to identify opportunities for cost reduction are early in the product life cycle before most of the costs are committed, so it is important to analyse the new design as soon as possible so that it can be costed by a process of analysis and refinement. Scanlan et al. (2006) make this point. This also implies however that the costing tool must be continuously refined as designs are refined, and/or new designs and products created. An ontology based modelling approach can assist with this as it allows the software design to be adjusted by redefining the ontology structures. Visualisation of these structures can ease this task and so increase the range of people who can undertake it. Semantic Web visualisation can make the models accessible and searchable, by a wide range of users. Chapter 5 illustrates the problems that occur when costing is undertaken without a sufficiently structured and editable ontology, such an ontology, and a modelling system based on it is prototyped in chapter 6.

The wingbox ontology is made up of taxonomies for each domain such as parts, processes, and materials. Wherever possible, agreement on the terminology and method of use for each taxonomy/sub ontology should be sought, or a published standard used e.g. Process Specification Language (PSL) of the National Institute of Standards and Technology (NIST), covered in section 3.5.1. This standardisation of terminology can make the translation simpler and enable interaction with other systems. This kind of solution is referred to by Uschold and Gruninger (2004). This standardisation makes it possible to expand the modelling capabilities to generic problems.

4.8 Enabling User Driven Modelling/Programming

This methodology is built on the combination of research in end-user programming, Semantic Web and Web 2.0, and modelling and visualisation examined in chapter 3. The thesis approach builds on previous work undertaken for Rolls-Royce aerospace to allow designers and manufacturers to visualise and share cost information (Scanlan et al., 2006). This was part of the DATUM (Design Analysis Tool for Unit-cost Modelling) project undertaken with Rolls-Royce to make costing by analysis possible early in the design of a new product. This project involved Rolls-Royce Aerospace, University of the West of England and Southampton University. During this project, one task undertaken in relation to this thesis was to automatically produce tree/graph representations of information requested by users. Information held in relational databases was visualised and exported in structured languages. The author was part of the DATUM project until 2006 when it relocated to Southampton University. The essence of the research methodology (developed in section 3.3.2) is that a high level visual interface is used to create a Meta-program, which can communicate the wishes of users. This requires the definition of taxonomies to provide the library of information to be used in the decision support modelling. A model builder could visually and diagrammatically populate and maintain this without needing to know programming languages. This kind of modelling would then enable the designers to influence the construction of decision support models dynamically. Once this is possible the model is responsive, instead of commissioning for software development, the domain expert model builder can communicate with a visual interface, and this translates to controlling code, which writes code to achieve what is required. Thus, the model builder is using a modelling system (created by a software developer) to create modelling systems. A program/model is expressed in terms of a diagram that represents each domain of information required, and this diagram holds structured language definitions. Separation of content of the information from any constraints of language and format enables this.

This user driven approach is similar to the way spreadsheets allow structured formula to be entered using a formula wizard. However, spreadsheets allow for construction of models that are hard to maintain and adjust as the scenario which they model changes. This is because of the difficulty that humans or computer applications have in accessing information when the information is held in cells rather than nodes of a taxonomy or other diagrammatic representation. Schrage (1991) explains how difficult it can be to find the underlying assumptions that are represented in a spreadsheet scenario. The difficulty of tracking the information and assumptions in a spreadsheet make it hard to integrate the model with other software applications. However, the large numbers of domain experts who undertake spreadsheet development indicates their desire to be creators of software models. Improvements are required in order to ease problems of Maintenance, Extensibility, Ease of Use, and Sharing of Information as are recommended in section 4.3, and these improvements will be developed in section 5.1.1.

The methodology involves creation of an elaborator that can output code, in various computer languages. This avoids the need to use loosely structured systems such as spreadsheets and allows people to develop models themselves. The model can be translated from a high level representation to a computer language or a Meta-programming syntax such as MetaL (Lemos, 2009) or Simkin (Whiteside, 2009). The elaborator needs only a few pieces of information. All information other than that dependant on user interaction, including the names of each node and its relationships to other nodes, needs to be held in a standardised data structure, e.g. a database or structured text file(s). A visual interface to this ontology is required so that model builders can maintain and extend it. The interface and model linking can follow a similar structure to that used for web networks; Anderson (2007) characterises this, “The Web is a network of interlinked nodes (HTML documents linked by hypertext)”. The automated translation provides an infrastructure of linked nodes and equations for web modelling. This enables Semantic Web Programming as advocated by Berners-Lee and Fischetti (1999).

Each node (elaborator) needs to be provided with the following pieces of information :-

1) A trigger sent as a result of user action. This is a variable containing a list of value(s) dependant on decisions or requests made by the user, last time the user took action. Each time the user makes a request or a decision; this causes the production of a tree or branch to represent it. This trigger variable is passed around the tree or branch as it is created. The interface to enable this is connected to and reads from the ontology.

2) Knowledge of the relationship between each node and its immediate siblings e.g. parents, children, attributes. So the elaborator knows which other elaborators to send information to, or receive from.

3) Ability to read equations. These would be mathematical descriptions of a calculation that contains terms that are items in the ontology. The equation would be contained within an attribute of a class, e.g. the class 'Material Cost' would have an attribute 'Material Cost Calculation' that holds an equation.

4) Basic rules of syntax for the language of the code to be output.

The way the elaborator finds the information held in **2** and **3** is dependent on the action that is taken in **1**. Thus, if a suitable ontology is created, the basis of the rules of construction of the code to be created are defined **4**, and the user has made choices, the user needs to take no further action and just wait for the necessary code to be output.

The system created will make use of Program Transformation, which allows for writing in one representation or language, and translating to another. Lieberman (2007) advocates this to enable automation of software production; (section 2.7.2 - 'Better Models'). This is particularly useful for language independent systematic information representation, and creation of models can be made more easily accessible and editable by drag-and drop editing of nodes. Use of Semantic Web technologies is a means for open standard representation of collaborative models, transformation into different representations as required, and for provision of a high-level interface as a tool for model visualisation and system creation. This enables application of the theory of distributed constructionism explained by Resnick (1996); allowing people to learn from building and sharing models. The outputs are made available on the web, as this is cheap and accessible to use and allows for distributed visualisation, and co-operation to enable new insights. The methodology involves creation of a translator that converts between representations of the problem as appropriate to different people and subject matter, and is customisable by users. Translations will be performed into programming and meta-programming languages and open standard information representation languages. The visualisation and models created will be displayed and interacted with via the web.

Figure 16 shows the methodology behind the Semantic Web modelling. The diagram explains the Semantic Web modelling process, at all stages from ontology to results visualisation. These stages will then be prototyped.

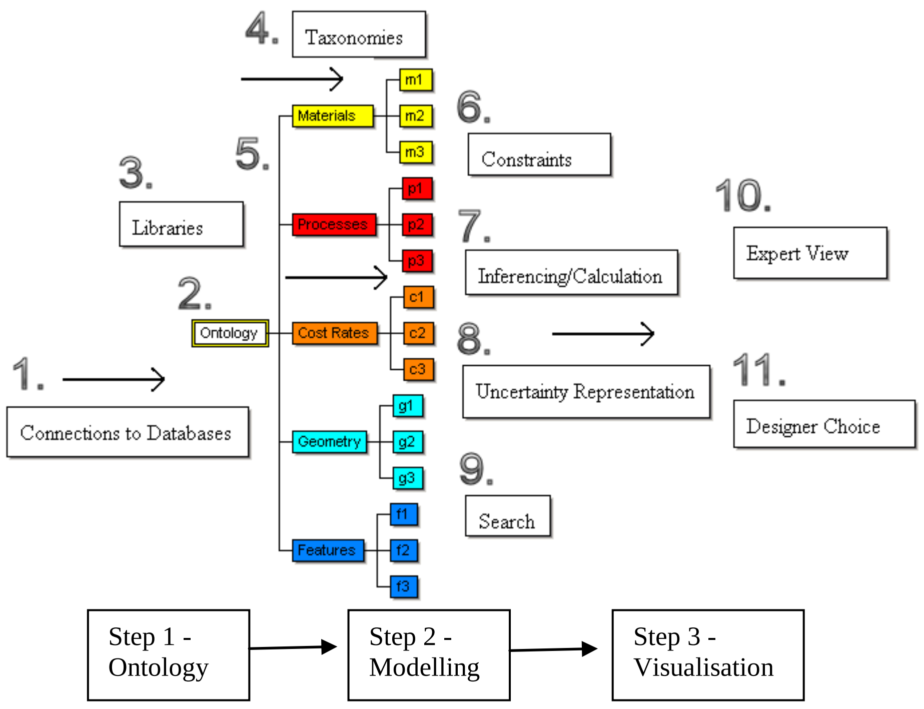


Figure 16. User Driven Modelling/Programming System

1. Connections are established between the ontology system and any databases, spreadsheets, or other systems that hold relevant information for that modelling problem.

2. The ontology is created using RDF/OWL (Section 3.3.2 discusses Bechhofer and Carrols' (2004) explanation of using RDF to encode tree structures, and section 4.7 includes a reference to their explanation of the different varieties of OWL. Above this ontology an interface is built to allow domain experts to edit the ontology.

3. Libraries are created in a partnership with domain experts.

4. Taxonomies/sub-ontologies are populated by model builders who want to use them for their modelling problem. These are based on the libraries created in step 3. The colour coding demonstrates the way sub-ontologies are linked into the overall ontology and each colour coded to differentiate between each sub-ontology and to show their position clearly within the overall ontology. This colour coding becomes especially important in a result tree as a result tree is likely to include calculations that take input from more than one sub-ontology, thus a branch may contain nodes of various colours e.g. a cost calculation based on Materials, Processes and Cost Rates.

5. Taxonomies are colour coded for ease of understanding. A link is created between the ontology tool and a decision support and calculation tool, which reads information from the ontology tool.

6. There are 2 sorts of constraints that can be used in order to make it easier for users to build and adapt models. These are constraints on the way the ontology and models are built, and user interface constraints to reduce the scope for error.

7. The colour coding makes calculation clearer because all taxonomies can be used and reflected in any calculation, this produces a multicoloured result tree/graph that represents the entire calculation history. User choices affect how items are related for the calculation; choices could be made manually or via a search. Colour could also be used to represent cost, time, and/or uncertainty (Bru et al., 2004).

8. Each node can also represent uncertainty, and prototypes have included uncertainty expressions in the calculations.

9. The result tree can be represented on the web and in other programs, this allows for further searching, navigation, processing and evaluation of results. Visualisation techniques and the use of searchable languages such as XML and SVG (Scalable Vector Graphics) can assist in this.

10. and 11. Experts such as designers can interact with the ontology, the model, and results, there is a two way feedback mechanism where the expert can make changes at any stage (steps 1, 2, or 3), and this filters into changed results. This can then support a cycle of results and rework.

Visualisation is important in allowing users to interact with the models created and see the results of any calculations and transformation into the result model. Examples of visual interfaces are shown in chapters 5 and 6 and also described in (Bru et al., 2002) and (Bru et al., 2004). The elaborator (de abstractor) needs to follow a structured ontology to establish how related concepts represented visually can be represented in equivalent code. The visualisation can be either as a colour coded tree or an interactive SVG diagram of a component to be modelled. Letondal (2005) researches this problem in a similar way and for graphical programming objects states:

“1. their ‘content’ may be defined by a *formula*,

2. their methods may be edited, modified, copied to define new *methods*

3. their graphical components are accessible as *program objects*

4. graphical attributes may be defined to implement *visualization functions*”

4.9 Conclusions

Development of prototypes will be undertaken in order to explore whether the research and methodology are realisable for the development process. Thinking about and examining this methodology at the same time as producing results via prototypes that can be refined and tested, enables refinement and testing of the methodology and the theory. Research practice is thus improved by the refinement of the research development and implementation. Chapter 5 prototypes and demonstrates the development and use of the User Driven Modelling/Programming methodology using examples.

Ch 5 - Development: Early Prototypes

5.1 Illustration of the Problem to be Addressed

This chapter illustrates problems that result from building a complex costing system within a spreadsheet, and the need for development towards and provision of alternative ways to represent such systems. Section 5.1 is based on work undertaken for Airbus to establish a way of representing information relating to the design and production of a wingbox (wing structure). The approach of developing decision support models for design and costing, using a spreadsheet is evaluated. It is argued that this approach is insufficient for providing generic and reusable models. The rest of the chapter examines ways to improve this by bringing together Ontology, Semantic Web, Modelling, and End-User Programming approaches to meet the aims and objectives discussed in chapter 1. So a contrasting approach will be explained of using open standards ontologies and software. The chapter begins with an explanation of the spreadsheet approach. This chapter gives a critical evaluation of the ACCS project spreadsheet; the project is explained in (Scanlan et al., 2002). ACCS was an Aerospace Composite Costing System created in a project developed between University of the West of England and Airbus Aerospace in which the author took part. Later in this chapter the first attempts are made to build an ontology based modeling system, this is illustrated with prototypes.

5.1.1 Spreadsheet Modelling

Spreadsheet modelling for the Airbus wingbox problem involved investigation of problems of Maintenance, Extensibility, Ease of Use, and Sharing of Information discussed in chapter 1. These problems can lead to errors. Scanlan et al. (2006) observe, “By their nature, large spreadsheets are difficult for a third party to comprehend as their inherent flexibility for editing allows users to generate a complex web of cell references which are arduous to audit.” Panko (2000) and Paine (2003) also throw light on this problem. Also, the spreadsheet may add to the problem, by hiding the detail behind an elaborate and visually attractive interface, but which does not clearly visualise the spreadsheet structure/content. Scanlan et al. explain that “if the author of such an application leaves the organization, it is commonly abandoned as colleagues are reluctant to master its complexity, and take ownership of it.” Paine ascertains that spreadsheets have almost no features for building applications out of parts that can be developed and tested independently. Panko (2000) suggests that “Given data from recent field audits, most large spreadsheets probably contain significant errors.” The most recent audit he cites found errors in at least 86% of spreadsheets audited. Panko reports that 90% of the spreadsheets audited in a study carried out by Coopers and Lybrand were found to have errors, Erwig et al. (2006) also cite a figure of 90% from (Rajalingham, 2001). Erwig et al.’s examination of spreadsheets was discussed in section 2.6. Scanlan et al. (2006) joins Erwig et al. (2006) in arguing the vulnerability of spreadsheets to mistakes, brittleness of complex spreadsheets, and difficulties in auditing and validating their contents. The studies by Paine, and Panko show that the chances of any given spreadsheet cell containing an error are somewhere between 0.3 and 3%, so that a spreadsheet of only 100 cells has about a 30% chance of having one error or more. An error in one cell can cascade to many other cells and so cause misleading results. Aragones et al. (2006) state, “Desktop spreadsheet users are very creative in their adaptations, but distributed spreadsheets have the problem of distributed, inconsistent inputs and distributed results. There is no easy way to aggregate the collective wisdom of user experience.” As discussed in section 4.3 - ‘Spreadsheets’, Hanna (2005) examined the declarative nature of spreadsheets; this makes them suitable for end-user programming. However, he criticises the lack of structure which makes the semantics of spreadsheet models hard to understand and interpret. This can make it difficult to maintain, re-use, collaborate on and extend spreadsheet models. Declarative semantics provide a means for end-user programming as long as this is within a structured environment, and as long as a sufficiently understandable graphical front end is provided that visualises the structure and semantics to allow interaction. Wakeling (2007) investigates creation of a spreadsheet using Haskell (Hudak. et al., 2007) in order to introduce spreadsheet users to functional programming. Hudak et al. explain the history of Haskell, its support for XML and Web scripting languages, and Haskell Graphical User Interface (GUI) research.

5.1.2 ACCS Example

5.1.2.1 Project Aim

The ACCS (Aerospace Composite Costing System) spreadsheet estimation tool was created for a project involving Airbus, to provide Airbus with a comparative cost for the manufacture of the various wingbox parts using carbon-fibre composite. This example (Figure 17 - described in 5.1.2.2) illustrates a design that is difficult to cost because of a change in process i.e. use of composites rather than metal for manufacture of a wingbox. In early study of design options, parametric cost models are often used to statistically relate cost to factors such as weight and manufacturing process. Composites have different cost drivers than metals so this invalidates the use of parametric models based on metals in order to cost composite structures, and products. Gutowski (2001) examines web-based costing of composites. (Scanlan et al., 2002 and 2006) investigate parametric cost estimating, and generative cost estimating. Brundwick (1995) and Duverlie et al. (1999) investigate issues in parametric modelling. Parametric costing is generally not suitable for new processes, whose characteristics are significantly different from those previously used, due to lack of historical information. So parametric models based on processes used already cannot be re-applied for the new processes. Much of the cost of a product is committed during the design process, so an alternative for parametric costing is needed even at an early stage of design (Scanlan et al., 2002 and 2006). A similar project to ACCS was undertaken by Eaglesham (1998); this costed composite manufacture in aerospace, using a database and querying/calculation.

Because of the difficulties applying parametric costing methods to new technologies, the ACCS project was undertaken to create software for the purpose of costing a product where parametric costing was not viable. Airbus specified that this should be a short project to enable costing the manufacture of a composite wingbox, and that a spreadsheet must be used for this because of availability of this package.

Costing of composites is an important area of research, as designers want to make use of the strength and weight properties of composites but need to be confident that the utilisation of such components is feasible and cost effective. Airbus used the spreadsheet frequently.

5.1.2.2 ACCS Implementation

The composite wing spreadsheet is a decision support tool for designers and manufacturers to evaluate the options for design and manufacture of composite wingbox components. It covers four main components - 'Skins', 'Spars', 'Ribs', 'Stringers', and possible manufacturing techniques for each. The spreadsheet model begins by providing the user with a choice of component to cost. Diagrams of generic wingbox components were provided to visualise the general shape and use of these components. Help pages were also provided at all stages of the costing. On choosing a component, in this case a spar, properties of the component are shown with default values. Colour coding is used to indicate to the user what values are editable. The derived values are recalculated to reflect any changes made by the user. When users are satisfied with the definition of the component, they press the 'Define Processes' button to begin choosing and defining manufacturing processes. Figure 17 shows the navigation system the user follows, although this is a simple system, sophisticated code had to be written to keep users to this path, and return them to this path after adding input values or viewing results.

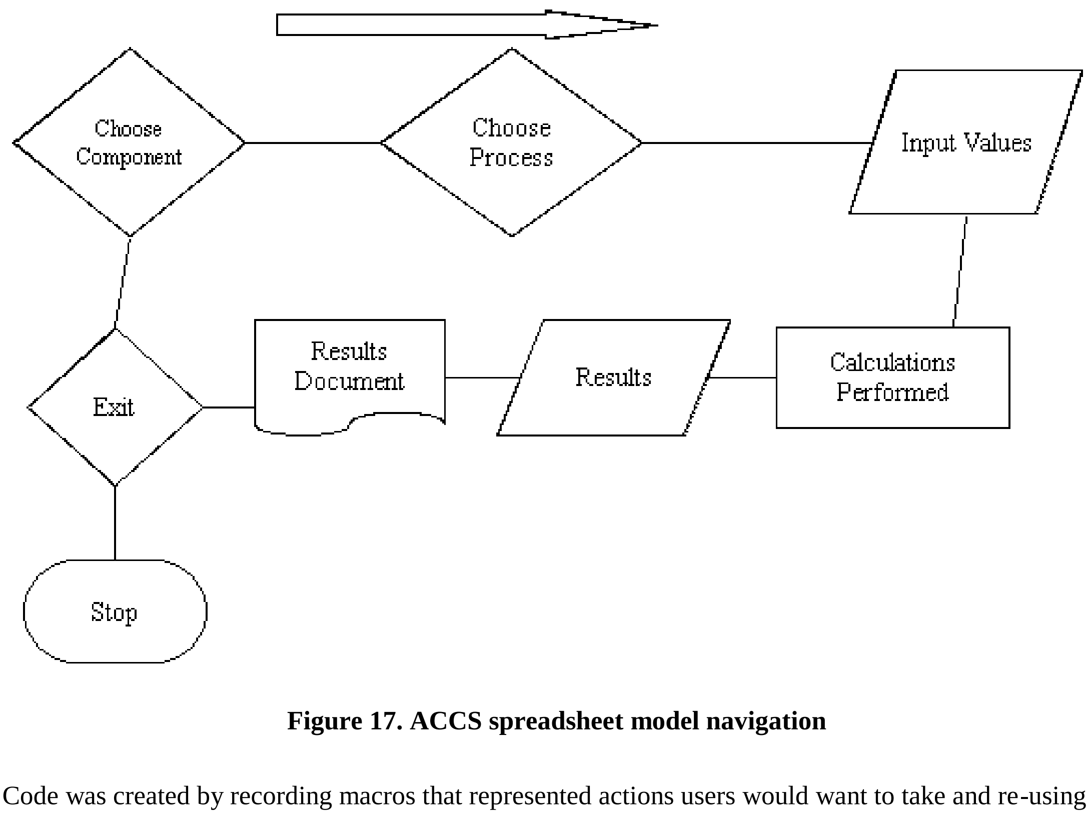


Figure 17. ACCS spreadsheet model navigation

Code was created by recording macros that represented actions users would want to take and re-using the code created from these actions. This is a simple way of using the Programming by Example technique explained in section 4.3 - 'Spreadsheets'.

5.2 Macros and Programming by Example

‘Programming By Example’ (PBE) is a technique where a software agent records user’s behaviour in an interactive graphical interface, then automatically writes a program that will perform that behaviour for the user. Spreadsheet ‘Record Macro’ functionality uses a Programming by Example approach, as mentioned in section 4.3. Macías and Castells (2004) use a Programming by Example approach for creation of model based user interfaces, using an ontology (section 4.4.1). This is part of their work to build systems for end-user programming.

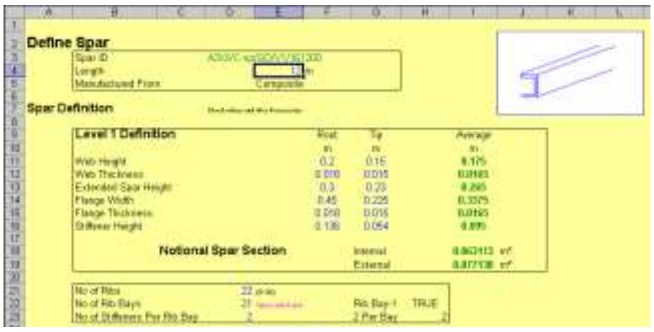
5.3 Visualisation and Interactivity for Translation

Jackiw and Finzer’s (1993) research on translations between visualisations was examined in chapter 2 (section 2.5). This thesis has concentrated on translating graph and tree representations to diagrammatic visualisations, but this translation is valid in either direction so future work will involve reversing the translation; this possibility is examined in section 8.2.2.

5.4 Web Spreadsheet and Semantic Modelling Examples

Reed et al. (2000) show how web-based modelling and simulation can be used for improving the aircraft design process. An approach of providing collaborative spreadsheets with a central information source was prototyped for this thesis. This was in order to link spreadsheet use with web-based modelling and simulation and enable sharing of information, and collaboration for modelling.

Installing spreadsheets on client computers does not provide multiple user access. The example below provides a centralised spreadsheet that makes use of a structured database to store the information, which it relies upon. A further advantage of this is that the database can be accessed by other applications. Figure 18 is an example page relating to wing spar manufacture[22]. Java based spreadsheets were also investigated, but later developed into a more visualised tree-based approach as demonstrated in section 6.5.3.2 based on the example research shown by Figure 4.



Define Spar

Spar ID	[illegible]
Length	[illegible] m
Manufactured From	Composite

Spar Definition [illegible]

Level 1 Definition	Root	Tip	Average
	m	m	m
Web Height	0.2	0.15	0.175
Web Thickness	[illegible]	0.015	[illegible]
Extended Spar Height	0.3	0.23	0.265
Flange Width	0.45	0.225	0.3375
Flange Thickness	[illegible]	0.015	[illegible]
Stiffener Height	0.136	0.054	[illegible]

Notional Spar Section Internal [illegible] m²; External [illegible] m²

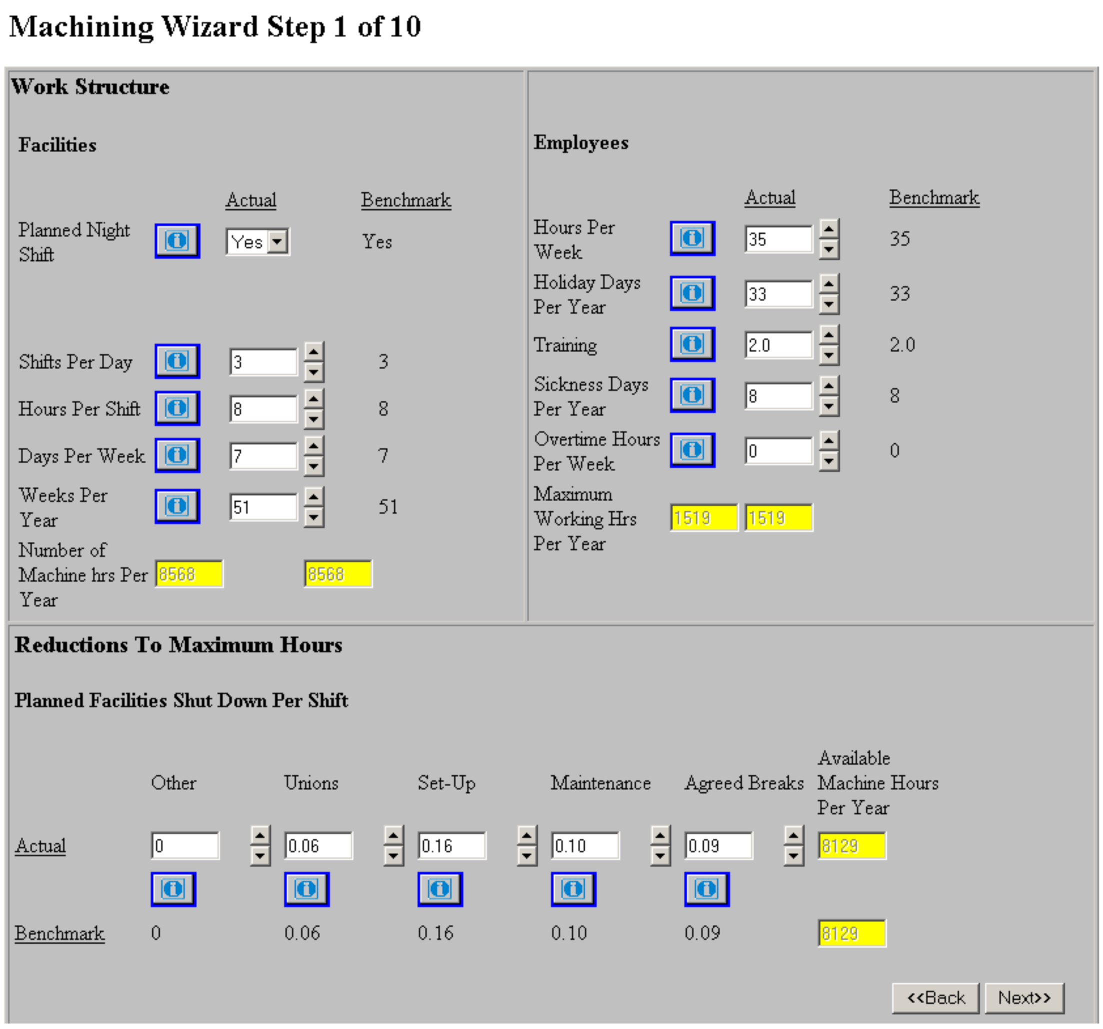
No of Ribs	22 [illegible]			
No of Rib Bays	21	Rib Bay 1	TRUE	
No of Stiffeners Per Rib Bay	2	2 Per Bay		21

Figure 18. Distributed Spreadsheet Spar Definition

[22] This was created using ExcelWriter (2011) [online]. Available from: http://officewriter.softartisans.com/
[Accessed 22 June 2011]. This software creates web spreadsheets or converts existing spreadsheets for use in a distributed web environment.

This spreadsheet enables re-use of the manufacturing information for other manufacturing problems, as the spreadsheet can link and contribute to a manufacturing ontology.

Figure 19 shows an early attempt to design a web-based representation of an Airbus spreadsheet built on an ontology infrastructure. This builds on the research leading up to Figure 18 by defining through an ontology, the creation of the spreadsheet and of controls for interaction with the spreadsheet and for defining of all the default values via the ontology. The web pages that reproduce the functionality link to the ontology, and respond to decisions made by the user. The controls are constructed appropriately according to data type e.g. JavaScript constructs the up/down increment buttons for increasing values as integer or decimal steps, and enables direct editing of the box whist applying the appropriate editing constraints for that data type. The information used to produce these spreadsheets is held in DAML+OIL XML files (DAML+OIL was explained by Horrocks (2002) as discussed in section 2.3.1 - 'Ontologies for Modelling and Simulation'). This approach of adding layers of structure to a basic XML file is used in ontology editors, and in Semantic Web and other web standards. These files can be edited to alter the information on which the spreadsheet is based. So the spreadsheet could be edited and reconstructed purely through editing the DAML+OIL code without changing the JavaScript code required for the interface creation. This makes it easier to manage the information and reuse it for a different problem. It also eases development of software for modelling different problems using the same approach. Figure 19 also demonstrates how information from the ontology can be read into the controls and how calculations are made, using the ontology information. The entire contents of what is shown in Figure 19 including all the labels, values and controls are defined in DAML+OIL. JavaScript software was developed to read the DAML+OIL and construct the interactive controls, visualisation, and interface for users to interact with and calculate from the source DAML+OIL information.



Machining Wizard Step 1 of 10

Work Structure

Facilities

	Actual	Benchmark
Planned Night Shift	Yes	Yes
Shifts Per Day	3	3
Hours Per Shift	8	8
Days Per Week	7	7
Weeks Per Year	51	51
Number of Machine hrs Per Year	8568	8568

Employees

	Actual	Benchmark
Hours Per Week	35	35
Holiday Days Per Year	33	33
Training	2.0	2.0
Sickness Days Per Year	8	8
Overtime Hours Per Week	0	0
Maximum Working Hrs Per Year	1519	1519

Reductions To Maximum Hours

Planned Facilities Shut Down Per Shift

	Other	Unions	Set-Up	Maintenance	Agreed Breaks	Available Machine Hours Per Year
Actual	0	0.06	0.16	0.10	0.09	8129
Benchmark	0	0.06	0.16	0.10	0.09	8129

Figure 19. DAML+Oil Web Program linking to ontology information

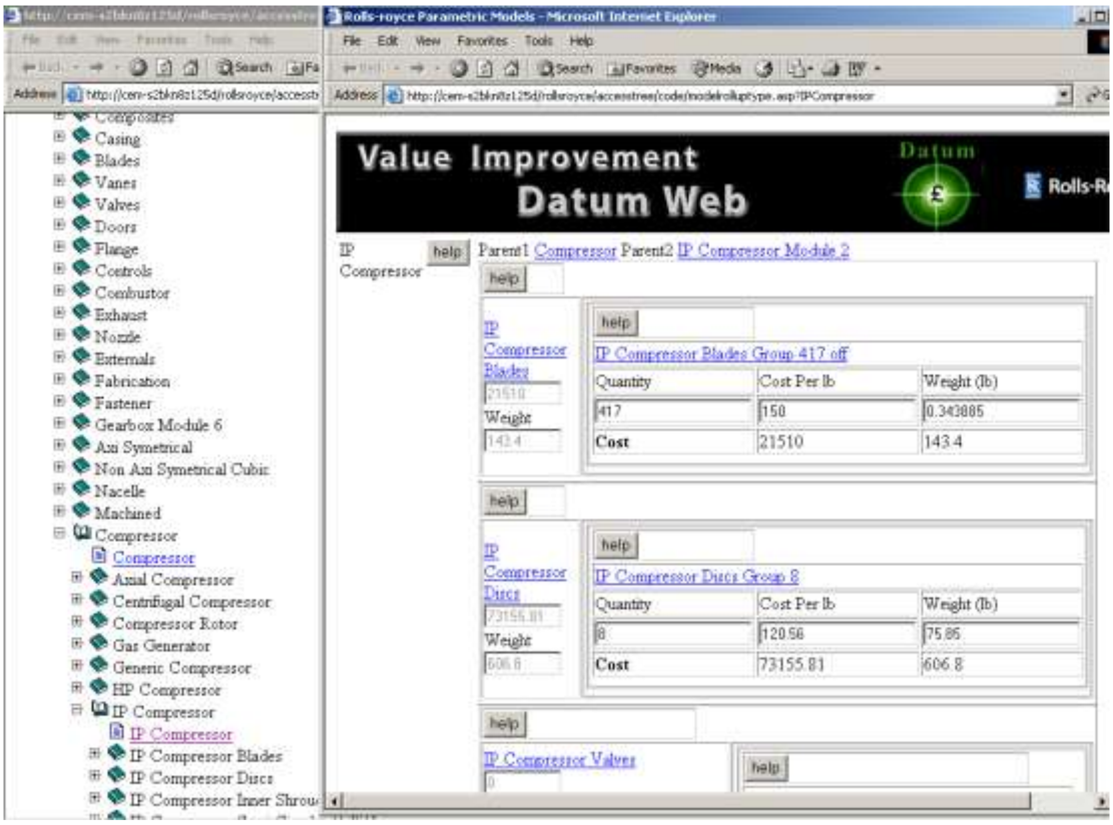
The yellow text boxes show calculated values. These calculations are made using JavaScript code that reads the equations from the DAML+OIL representation into functions. This causes a loss of flexibility as only a programmer can create and develop use for new equations that may be needed for the calculations. Editing the code is reasonably easy for any programmer, and does not require compiling or any special software. However this is still too much to expect from non-programmers. Another type of problem occurs with the control of program flow. Decisions that the user makes affect later options, but a non-programmer cannot change the structure of this flow of decisions. The advance that the Figure 19 implementation makes over the Figure 18 implementation is the incorporation of some the functionality available from spreadsheet languages such as Excel (Visual) Basic into the ontology, read by the JavaScript in a partially generic way.

Although the HTML and JavaScript interface was abandoned in exchange for a more maintainable XForms implementation, this work demonstrated it was possible to structure information appropriately in linked taxonomies/sub ontologies to create a structured ontology that is constructed using an open standard language. The use of XForms also enables users to maintain the ontology, in step with the application. The example proved the feasibility of providing a web application. Web software was also used for the production of parametric models, based on common data held in a relational database or XML on a server. This is explained with an example in section 5.5.2 - ‘Better Representation and Structure’.

5.5 Necessary Improvements

5.5.1 Better Visualisation

Proper visualisation of the structure was missing from the previous test implementations. In Figure 20 the full program structure is visualised as a tree hierarchy. The user selects the component from an XML based menu and queries are run which return the appropriate costing (simple parametric). If a component is chosen, in this case ‘IP Compressor’, the information related to this component including its parents and children is shown. This was limited to a few common attributes ‘Quantity’, ‘Cost Per lb’ (test values), and ‘Weight’ but this could have been extended to provide a more detailed list. The example uses test quantities and costs only. The example is based on aircraft engine structure and components; knowledge from discussions with Rolls-Royce was used to create the hierarchy.



IP Compressor Blades Group 417 off

Quantity	Cost Per lb	Weight (lb)
417	150	0.343885
Cost	21510	143.4

IP Compressor Discs Group 8

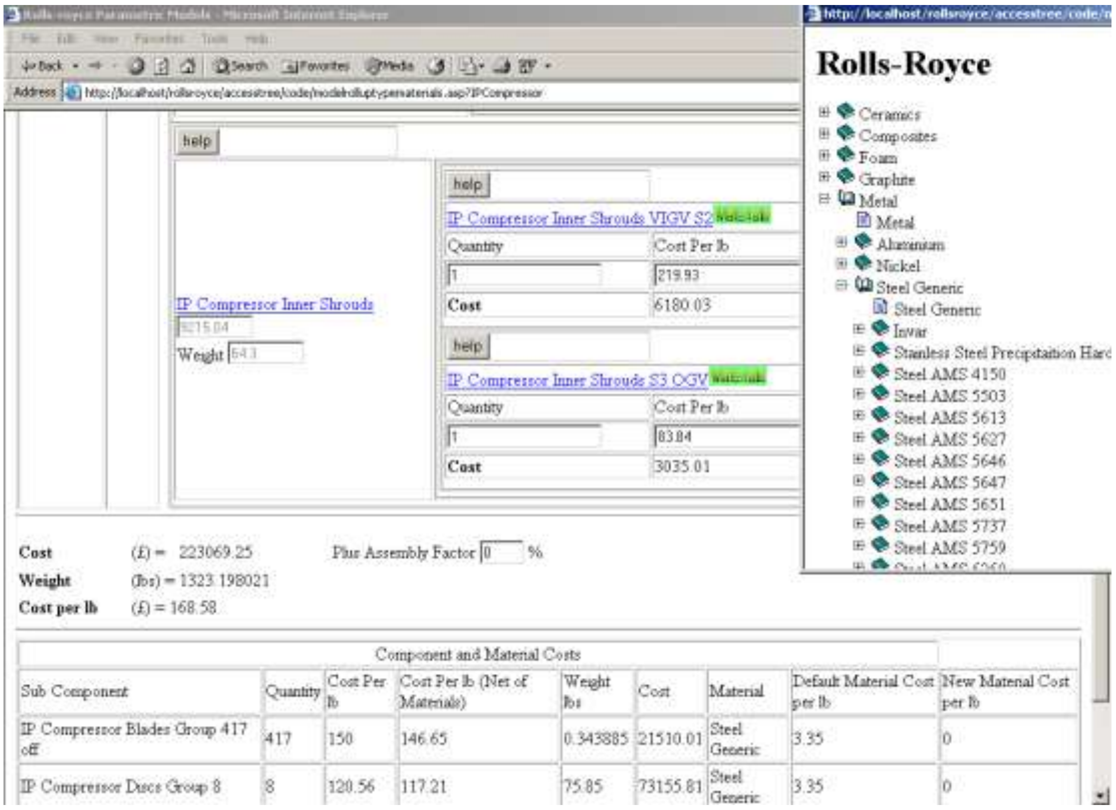
Quantity	Cost Per lb	Weight (lb)
8	120.56	75.86
Cost	73155.81	606.8

Figure 20. Ontology of Engine Components and related information

The tree menu used in Figure 20 and Figure 21 uses a stylesheet created by De Andreis[23]; this menu is used to link and provide navigation for all the related sub ontologies and models. This example was

[23] De Andreis, E. (2009) *2MXtree XML-based tree* [online]. Available from: http://manudea.duemetri.net/manudea/xtree/default.asp [Accessed 25 July 2011].

extended to allow for the choice of Materials for each Component from a similarly structured Materials hierarchy The menu appears in response to the user clicking on the Material Button to choose a material. Figure 21 shows this and also illustrates how costs can be calculated and totalled.


http://localhost/rollsroyce/accesstree/code/modelrolluptypematerials.asp?IPCompressor

help

IP Compressor Inner Shrouds
9215.04
Weight 54.3

help

IP Compressor Inner Shrouds VIGV S2

Quantity	Cost Per lb
1	219.93
Cost	6180.03

help

IP Compressor Inner Shrouds S3 OGV

Quantity	Cost Per lb
1	83.84
Cost	3035.01

Cost (£) = 223069.25 Plus Assembly Factor 0 %
Weight (lbs) = 1323.198021
Cost per lb (£) = 168.58

Component and Material Costs

Sub Component	Quantity	Cost Per lb	Cost Per lb (Net of Materials)	Weight lbs	Cost	Material	Default Material Cost per lb	New Material Cost per lb
IP Compressor Blades Group 417 off	417	150	146.65	0.343885	21510.01	Steel Generic	3.35	0
IP Compressor Discs Group 8	8	120.56	117.21	75.85	73155.81	Steel Generic	3.35	0

http://localhost/rollsroyce/accesstree/code/

Rolls-Royce

- Ceramics
- Composites
- Foam
- Graphite
- Metal
  - Metal
  - Aluminium
  - Nickel
  - Steel Generic
    - Steel Generic
    - Invar
    - Stainless Steel Precipitation Hard
    - Steel AMS 4150
    - Steel AMS 5503
    - Steel AMS 5613
    - Steel AMS 5627
    - Steel AMS 5646
    - Steel AMS 5647
    - Steel AMS 5651
    - Steel AMS 5737
    - Steel AMS 5759

Figure 21. Selection of Material for components

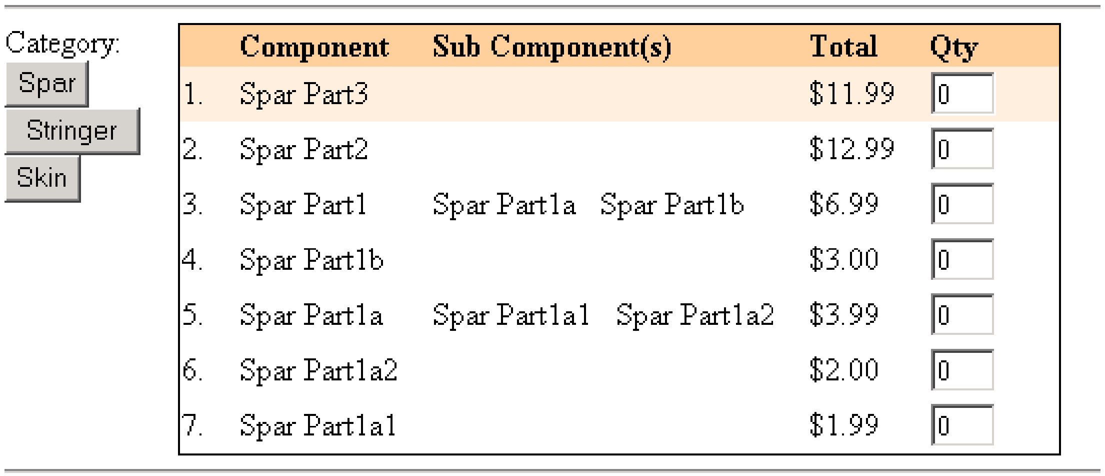
The menu was moved to the right for clear presentation as this shows the menu reused with different information (materials rather than components) and for the purpose of selecting a material for the component already selected. A shopping cart metaphor was useful for summing the costs from models for many components so that a whole engine costing could be created. This application was also the basis for a re-implementation, using XForms for ease of maintenance and ontology development, which is described below.

5.52 Better Representation and Structure

Highly interactive web pages that act like programs to provide a user interface can be used to provide an interactive User Driven Modelling/Programming environment. One methodology investigated is use of XForms[24] web forms to capture the information required for each node in a tree. XForms enables automated production of user-interfaces from XML information; the advantages of this were argued in section 2.7.2 'Better Models' referencing Bishop (2006). The XForms can be made to look like a web page, spreadsheet, or whatever tool the user is most familiar with. This can be used as a way to capture

[24] Bruchez, E, 2006. XForms: an Alternative to Ajax?. *In: XTech 2006: Building Web 2.0* 16-19 May 2006, Amsterdam, The Netherlands.

information from users for an ontology. An example of this is Orbeon Forms[25], which is open source software that can store information in an Exist[26] XML database. XForms can be the basis for interactive applications that are easier to maintain than the HTML and JavaScript forms created earlier in this research (5.5.1). XForms can be used for managing information from XML files. It is possible to add, delete and edit any XML file, so this provides infrastructure for a user interface for end-users to create XML documents online, which can represent models. The most important consideration in use of XForms for end-user programming is to visualise and allow editing of the information structures, rather than depend on the structure being understood only from its text representation. Figure 22 shows example wing component information edited using this system (this contains test values only).



	Component	Sub Component(s)		Total	Qty
1.	Spar Part3			$11.99	0
2.	Spar Part2			$12.99	0
3.	Spar Part1	Spar Part1a	Spar Part1b	$6.99	0
4.	Spar Part1b			$3.00	0
5.	Spar Part1a	Spar Part1a1	Spar Part1a2	$3.99	0
6.	Spar Part1a2			$2.00	0
7.	Spar Part1a1			$1.99	0

Figure 22. XML Based Software for capturing information

5.5.3 Better Interactivity

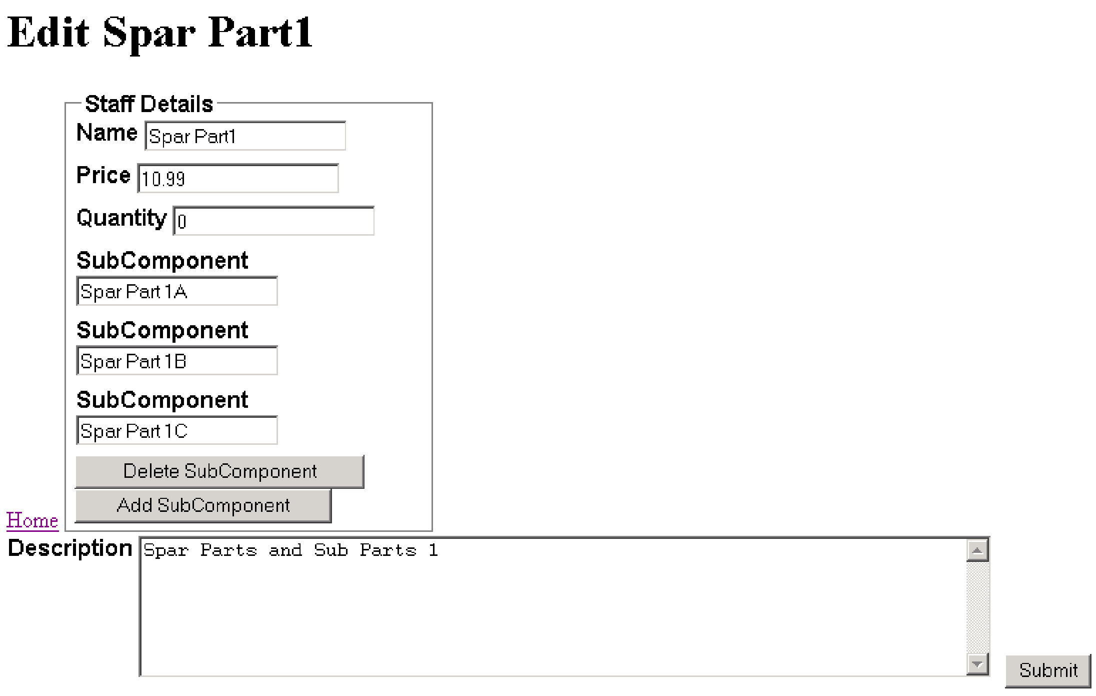
The above application enables selection of components and sub components, and calculates the total cost. XForms was also used to manage staff information, demonstrating that it was particularly easy to reuse XForms from one application, for a different application, by editing the XML and form. This online editing could be expanded to enable online creation and editing of models, in a similar way that Protégé was used in the research to perform this task offline. Future research on online editing is explained in 7.3 - ‘Widening Participation: Online Editable Systems’.

The interactive examples shown earlier in this chapter (5.5.1) demonstrate the kind of calculation and modelling that could now be achieved more easily using XForms. Those examples had used combinations of XML and JavaScript that work in a similar way to XForms and enabled the same kinds of user interaction; but XForms based solutions proved to be more maintainable. XForms can be

[25] Orbeon Forms (2011) *Form-based web applications, done the right way* [online]. Available from: http://www.orbeon.com/ [Accessed 25 July 2011].

[26] Exist (2009). *Open Source Native XML Database* [online]. Available from: http://exist.sourceforge.net/ [Accessed 25 July 2011].

used in combination with XQuery (McGovern et al., 2003) to allow for completely XML based interactive applications, illustrated by Figure 22 and Figure 23[27].



Edit Spar Part1

Staff Details	
Name	Spar Part1
Price	10.99
Quantity	0
SubComponent	Spar Part 1A
SubComponent	Spar Part 1B
SubComponent	Spar Part 1C
Description	Spar Parts and Sub Parts 1

Figure 23. Automated Generation of Web Forms

Provision of an editor would mean changes made by the user are fed back to the XML, so users can change information, and thus change the model without ever needing to see the XML code. The reason for investigation of XForms is that ontologies translated and linked to XForms XML files, schemas and stylesheets, can enable creation of a web-based user interface for modelling and ontology management. This combination of Web 2.0 and XML technology behind an adaptable and extensible user interface can enable experts who may have no knowledge of the Semantic Web or programming to create, manage, use and edit applications. This would enable user-generated Semantic Web information to be produced and so create an incentive for development/linking of further Semantic Web applications for use by end-user modellers/programmers. Such arguments and examples are developed further in 6.4.4.

5.6 Discussion

These prototypes progress towards but do not fully satisfy the aims of improving collaborative software development through the interaction with diagrams without requiring people to learn computer languages.

[27] Hale, P. (2009) Wingbox *XForms Example* [online]. Available from: http://www.cems.uwe.ac.uk/amrc/seeds/FormFaces/Examples/WingBox/index.html
- Explanation - http://www.cems.uwe.ac.uk/amrc/seeds/Ajax/ajax.htm#XForms [Accessed 25 July 2011].

The prototypes address maintenance, extensibility, ease of use, and sharing of information. Maintenance is improved by providing ontologies that models can share, but although the ontologies are implemented using Semantic Web languages they do not yet have a sufficient visual interface. So the systems could be improved by using a visual ontology editor. The implementations do not allow users to edit items and equations, or structure online. While this prevents model users from accidentally changing or overwriting equations (which can happen with spreadsheets, even if cells are protected, users can switch off the protection) it also prevents model builders from editing and extending the models unless they access the Semantic Web and code files and edit them. So this hampers maintenance and extensibility by computer literate end-users, restricting this to programmers. For most users these restrictions improve ease of use, so these examples show useful user interfaces, but development of an ontology based production system for these interfaces would allow model builders to create models also. This would enable more automation and end-user programming and still keep the advantages of ease of use for model users.

5.7 Recommendations

Evidence from the implementations outlined in this chapter indicates that Maintenance, Extensibility, Ease of Use and Sharing of Information problems can be solved by :-

1. Improving the structuring of information and model/program structure using visualised editable ontologies implemented with open standard languages.

2. Making use of structured development tools such as intermediaries between model builders/users and computer code. The user interface and translation capabilities of XForms and XQuery, and Semantic Web technologies can provide interaction with the model and/or ontology.

3. Visualising the model structure fully to engage users and ensure navigation and feedback to users is in a way they would expect and understand.

Chapter 5 investigated the theory explained in chapter 3 and the design and methodology in chapter 4, in order to establish how the research areas could be brought together and implemented. Some progress was made towards developing an ontology based modelling system. Chapter 6 is developed from test applications created to test this ontology based modelling approach, and aims to bring the research ideas based on Semantic Web/Ontologies/End User Programming, and Modelling together into a single working system. Chapter 6 explains this revised implementation.

Ch 6 - Development: Final Prototype Implementation

Chapter 6 describes the implementation of the User Driven Modelling/Programming theory outlined in chapter 3 where Semantic Web/Ontologies/End User Programming, and Modelling overlap, then developed in chapter 4 and prototyped in chapter 5. This is in order to ease problems of Maintenance, Extensibility, Ease of Use, and Sharing of Information. The prototype was developed in order to provide a system that could be used for modelling a complex system that is liable to be changed. To develop and implement a User Driven Modelling/Programming approach an ontology was required, a way to visualise and edit this, and a translation so the model based on the ontology could be automatically updated in step with the ontology developed.

6.1 Background to Projects and Implementation

The prototype is based on projects with Airbus to model wingbox design and manufacture processes and costs and the Rolls-Royce DATUM project (Scanlan et al., 2006) for process and cost modelling applied to engine design and manufacture. Both projects involved extensive collaboration between the UWE process/cost modelling team and teams at these companies, and frequent meetings for feedback.

Both Airbus and Rolls-Royce have deadlines based on projects such as to develop new products. This focuses the companies on meeting the deadlines, but this can hinder long term systematic process modelling system development. So it was necessary to focus on creation of a system that could be used continually, and be reusable for one project after another. This also created an opportunity to make the system reusable across different companies/organisations, and for different types of projects.

6.2 Wing Spar Explanation

The main implementation examples in this chapter are based on a representation of this Wing Spar (wingbox component) and associated processes, though Wing Ribs, Skins, and Stringers (Stiffeners) were also modelled. Figure 24 illustrates the Wing Spar :-

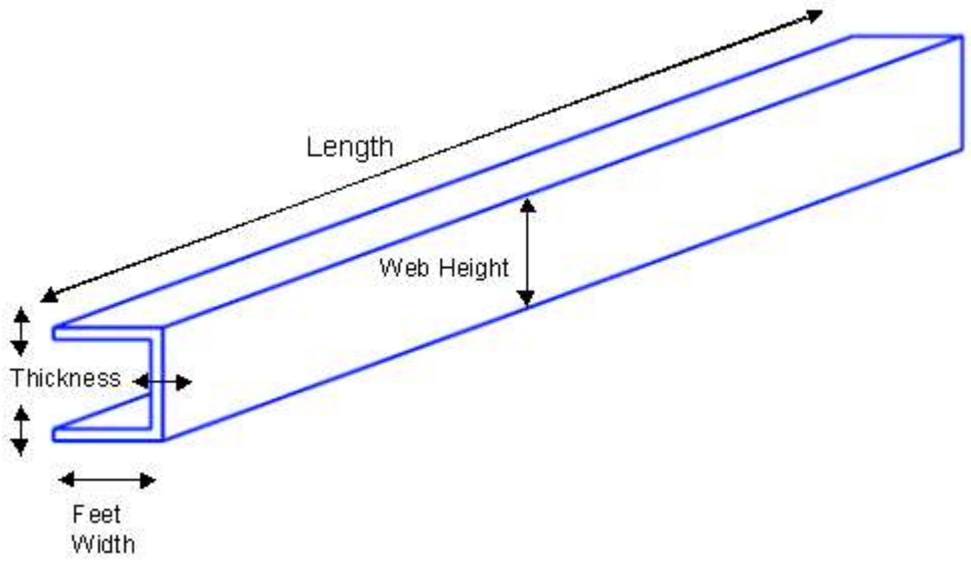
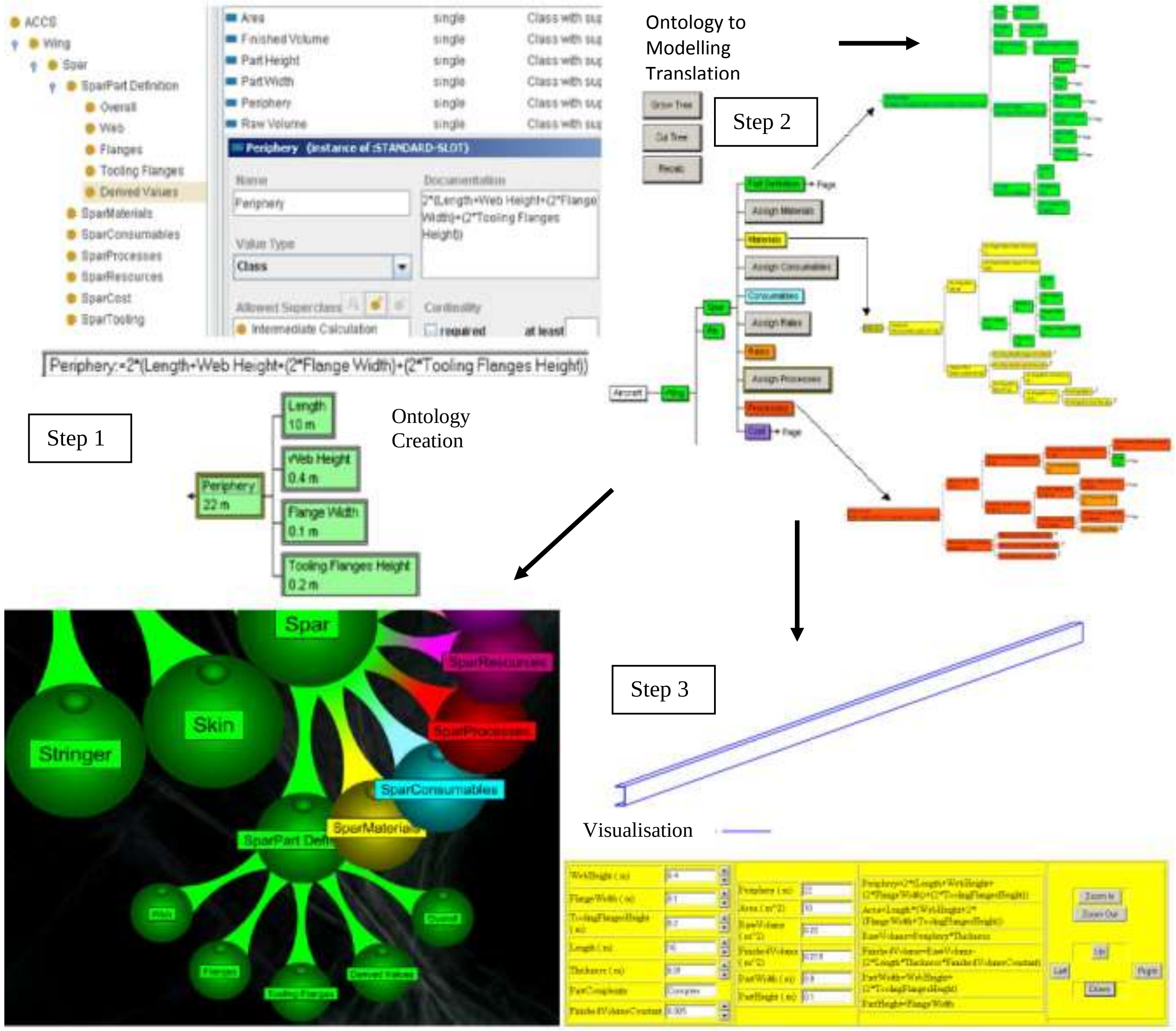


Figure 24. Wing Spar Diagram

To assist with making the spar example and the process for its creation understandable, simpler examples based on a rectangle and a square are demonstrated first.

6.3 Implementation Techniques and Examples

Figure 25 illustrates the implementation of the translation stages. Step 1 is creation of the ontology, which is then translated to the decision support and modelling tool (Vanguard System) for Step 2. Step 2 is illustrated more fully to the right, and this shows colour coding of the taxonomies (sub ontologies) that make up the ontology e.g. parts, processes, and materials. Step 3 involves translations to visualisations for the web (using Semantic Web formats) and alternative representations. Step 3 can also produce program and/or meta-program code.



Area	single	Class with sup
Finished Volume	single	Class with sup
Part Height	single	Class with sup
Part Width	single	Class with sup
Periphery	single	Class with sup
Raw Volume	single	Class with sup

Periphery (instance of :STANDARD-SLOT)

Name: Periphery

Documentation: 2*(Length+Web Height+(2*Flange Width)+(2*Tooling Flanges Height))

Value Type: Class

Allowed Superclass: Intermediate Calculation

Cardinality: required at least

Periphery:=2*(Length+Web Height+(2*Flange Width)+(2*Tooling Flanges Height))

Length 10 m

Web Height 0.4 m

Flange Width 0.1 m

Tooling Flanges Height 0.2 m

Periphery 22 m

WebHeight (m)	0.4
FlangeWidth (m)	0.1
ToolingFlangesHeight (m)	0.2
Length (m)	10
Thickness (m)	0.03
PartComplexity	Complex
FinishedVolumeConstant	0.005

Periphery (m)	22
Area (m^2)	10
RawVolume (m^3)	0.66
FinishedVolume (m^3)	0.218
PartWidth (m)	0.8
PartHeight (m)	0.1

Periphery=2*(Length+WebHeight+(2*FlangeWidth)+(2*ToolingFlangesHeight))
Area=Length*(WebHeight+2*(FlangeWidth+ToolingFlangesHeight))
RawVolume=Periphery*Thickness
FinishedVolume=RawVolume-(2*Length*Thickness*FinishedVolumeConstant)
PartWidth=WebHeight+(2*ToolingFlangesHeight)
PartHeight=FlangeWidth

Figure 25. Stepped Translation and Visualisation

Section 6.3 explains how the translation is achieved, using a simple example illustration. Section 6.4 examines how the aim is to bring together modelling, ontologies, Semantic Web and Web 2.0 techniques to enable User Driven Modelling/Programming through the 3 step process of Step 1 Ontology, Step 2 Modelling, and Step 3 Visualisation (especially over the web); and translation between these steps. Thus section 6.4 contains examination of the technologies required. Examples to illustrate the 3 main steps of the User Driven Modelling/Programming approach will be illustrated in section 6.5 in order to provide implementation and prototyping of the techniques and technologies illustrated in 6.4.

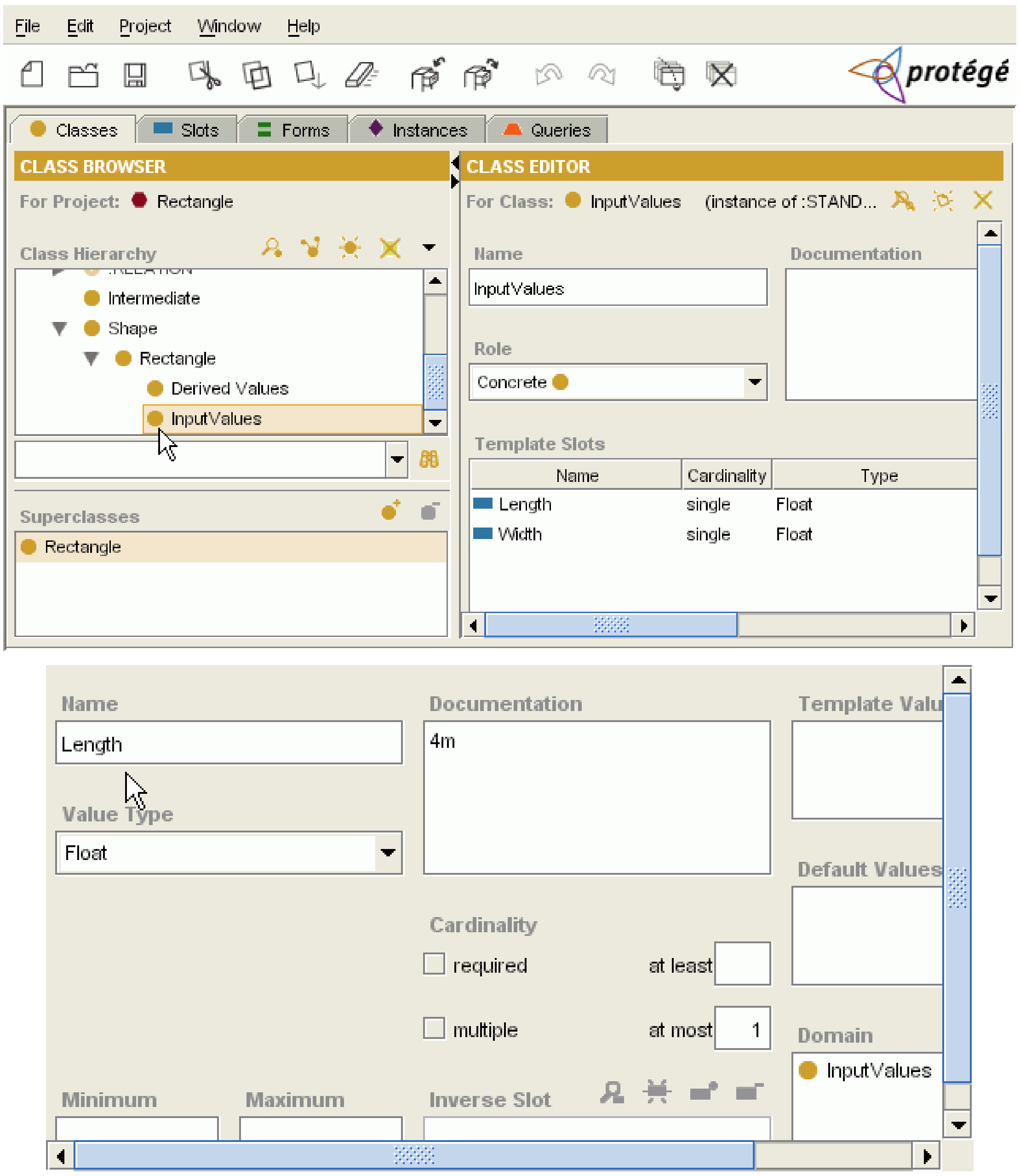
6.3.1 Implementation Simple Example

This simple model explains all the implementation of translation steps. A movie demonstration of this is available here[28].

6.3.1.1 Step 1

A rectangle is defined with attributes for length and width, shown in Figure 26.

[28] Hale, P. (2009) *Implementation Simple Example* [online]. Available from: http://www.cems.uwe.ac.uk/~phale/RectangleDemo/RectangleDemo.viewlet/RectangleDemo_viewlet_swf.html - Explanation - http://www.cems.uwe.ac.uk/~phale/RectangleDemo/RectangleDemo.viewlet/RectangleDemo_launcher.html [Accessed 25 July 2011].



CLASS BROWSER

For Project: Rectangle

Class Hierarchy

- :RELATION
- Intermediate
- Shape
  - Rectangle
    - Derived Values
    - InputValues

Superclasses

- Rectangle

CLASS EDITOR

For Class: InputValues (instance of :STAND...

Name: InputValues

Documentation:

Role: Concrete

Template Slots

Name	Cardinality	Type
Length	single	Float
Width	single	Float

Name: Length

Documentation: 4m

Template Valu

Value Type: Float

Default Values

Cardinality

required at least

multiple at most 1

Domain: InputValues

Minimum

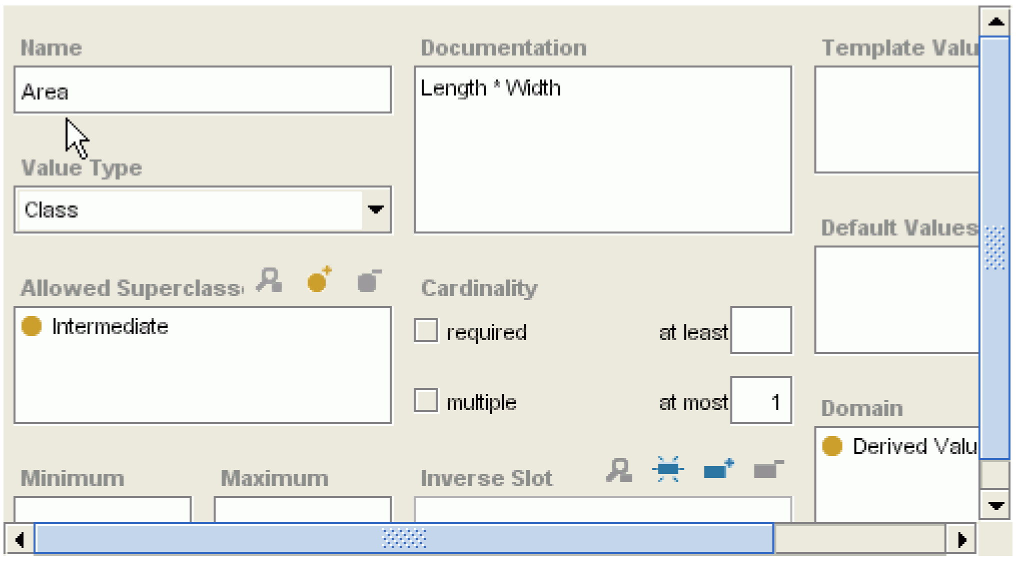
Maximum

Inverse Slot

Figure 26. Rectangle Definition in Ontology

Width is then defined as 2m.

In Figure 27 another class is created for calculation of area.


Name: Area

Value Type: Class

Documentation: Length * Width

Template Valu

Default Values

Allowed Superclass: Intermediate

Cardinality: required at least; multiple at most 1

Domain: Derived Valu

Minimum

Maximum

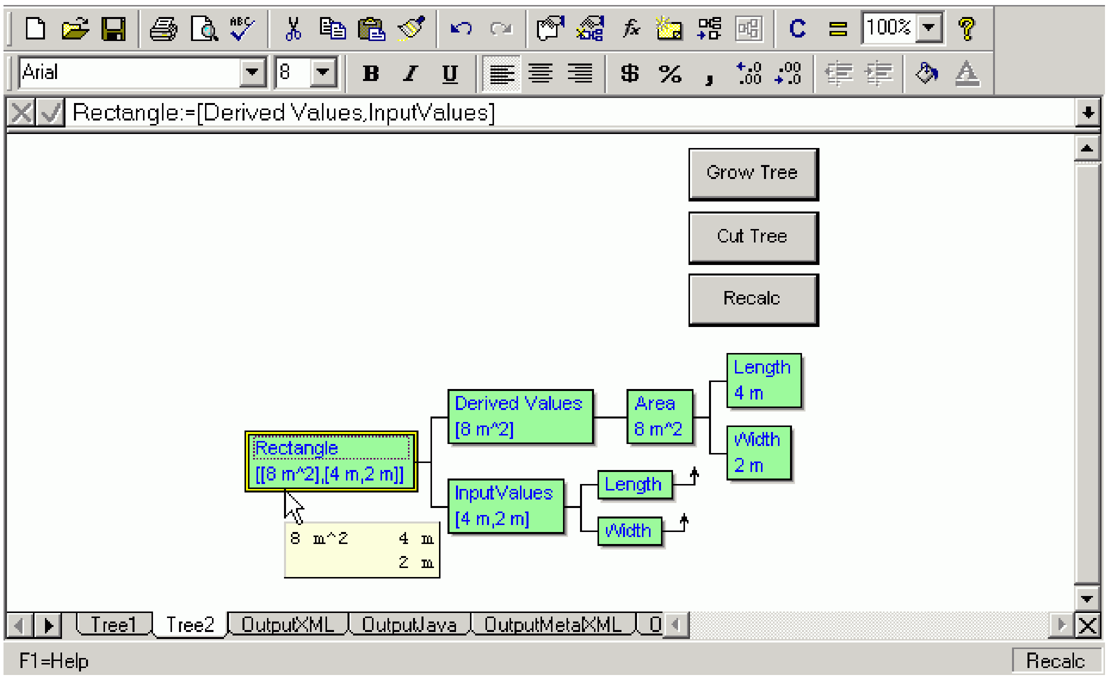
Inverse Slot

Figure 27. Rectangle Area Attribute

In Figure 27 ‘Area’ was assigned a value of ‘Length’ * ‘Width’. This is a simple equation that will be used to calculate the result. This illustrates how modelling calculations are performed. They are all defined by equations that relate attributes of the taxonomy/sub ontology. The taxonomy can be read by the decision support system in Step 2.

6.3.1.2 Step 2

For Step 2 the decision support/modelling system reads the ontology, performs the calculation and deals with units giving the result for area as 8 metres squared. This is shown in Figure 28 below :-


Rectangle:=[Derived Values,InputValues]

Rectangle [[8 m^2],[4 m,2 m]]

Derived Values [8 m^2]

Area 8 m^2

Length 4 m

Width 2 m

InputValues [4 m,2 m]

Length

Width

```
8 m^2      4 m
           2 m
```

Figure 28. Step 2 - Translation and Calculation

The modelling system can then output the results to web-based visualisations, or to program/meta-program code.

6.3.1.3 Step 3

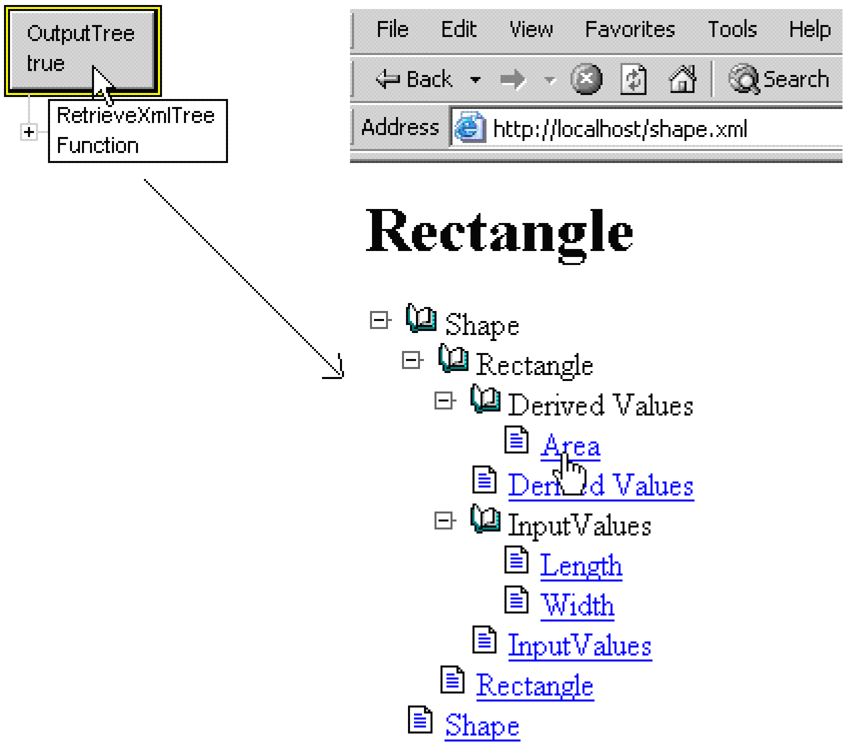
As well as showing the model itself on the web it is possible to translate the model results into other representations and visualisations, so making it as widely accessible as practical. Figure 29 shows the result model translated into XML and visualised as a tree view on the web. The automated menu provides for some of McGuinness (2003) 7 uses of simple ontologies (discussed in section 2.3.2 - ‘Semantic Web and Ontologies’) :-

2. site organization and navigation support

3. expectation setting

5. browsing support

The use and visualisation of Semantic Web languages at all stages in the translation process facilitates 6. ‘Search support’, and 7. ‘sense disambiguation support’; the structure of the ontology is visualised in order to enable users to determine the context of terms.


OutputTree
true
RetrieveXmlTree
Function
File Edit View Favorites Tools Help
Back
Search
Address
http://localhost/shape.xml

Rectangle

- Shape
  - Rectangle
    - Derived Values
      - Area
    - Derived Values
    - InputValues
      - Length
      - Width
    - InputValues
  - Rectangle
- Shape

Figure 29. Results Output as Tree (XML based)

Figure 30 shows an output SVG rectangle diagram that includes interactivity; this has been translated from the tree/graph-based representation. The input values used for the calculation and the diagram itself can be changed via an automatically produced user interface that is related to the taxonomy structure. These changes cause the shape representation and the area to be recalculated.

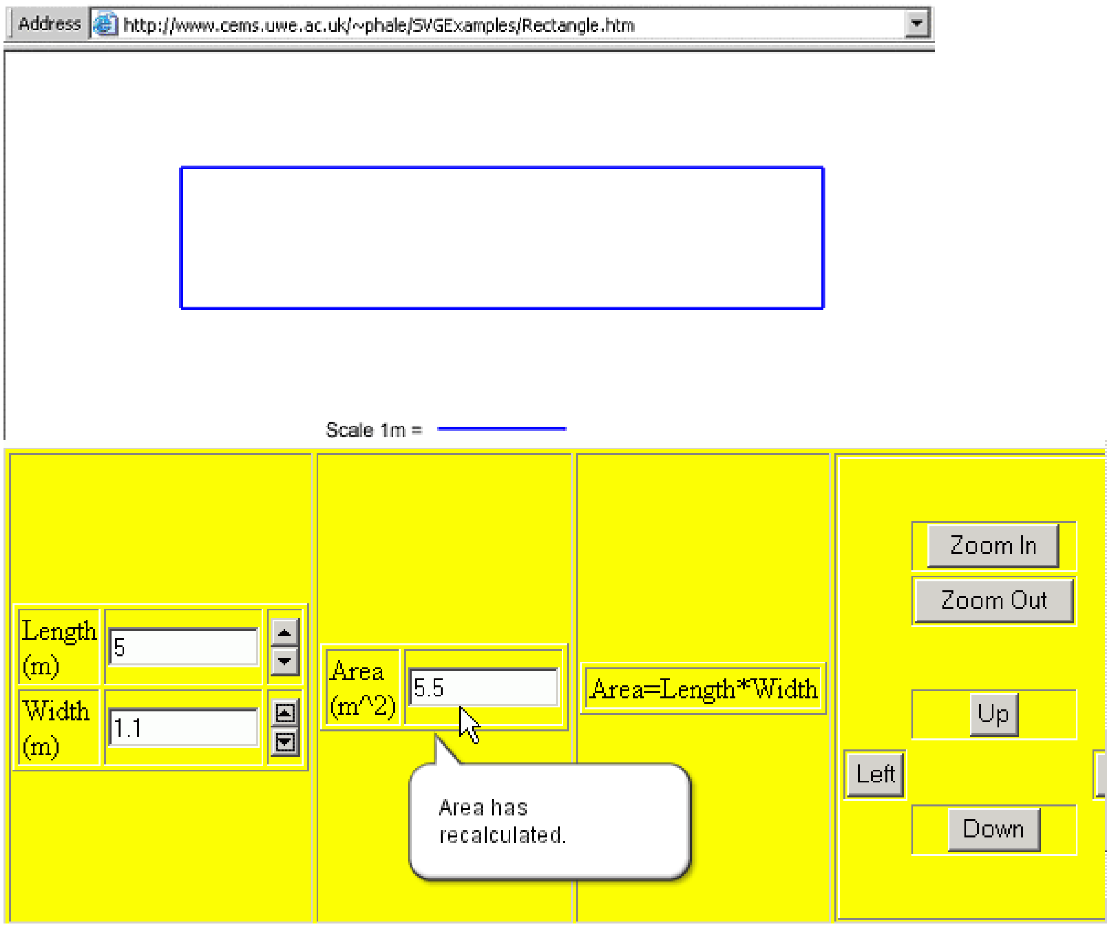


Figure 30. CAD type interface and User Interaction and Calculation

Section 6.4 explains the benefits of this translation process and how they are achieved.

6.3.2 Implementation Cube Example

This example is provided to illustrate the structure and process for creating the ontology, model, and visualisation/representation used for translation process from step 1 to step 3. This also helps to demonstrate how the research provides a solution for generic and reusable engineering modelling, by providing a real but simple demonstration of this modelling approach being used for an engineering model.

The cube model, as for all the engineering/process models is made up of the definition, in this case of the cube, and a colour coded representation of all the processes, materials, tooling, consumables, resources, and rates used for the manufacture of the cube; these are read in from the ontology in response to user choices. This makes it possible to investigate scenarios such as in this case whether to manufacture using welding, or riveting, and different options for use of tooling, consumables, resources, and rates. From investigating different options, different trees are created to represent different paths/options, and from this the production cost tree is created with results and feedback on exactly what made up the process/cost. Figure 31 illustrates how the different sub ontologies/taxonomies are colour coded in order to ensure it is easier to read the meaning of the tree and the interrelationships between the different aspects of the model.

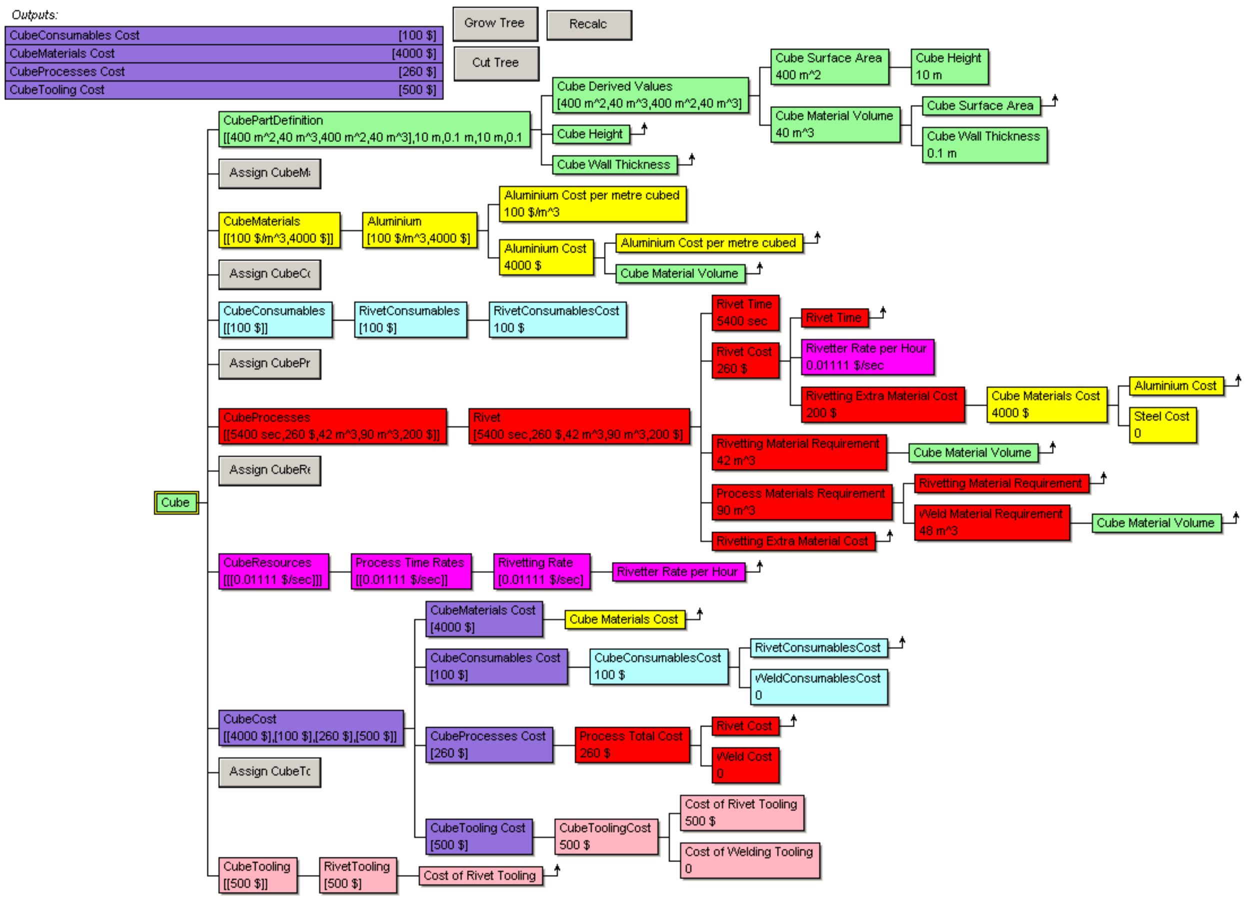
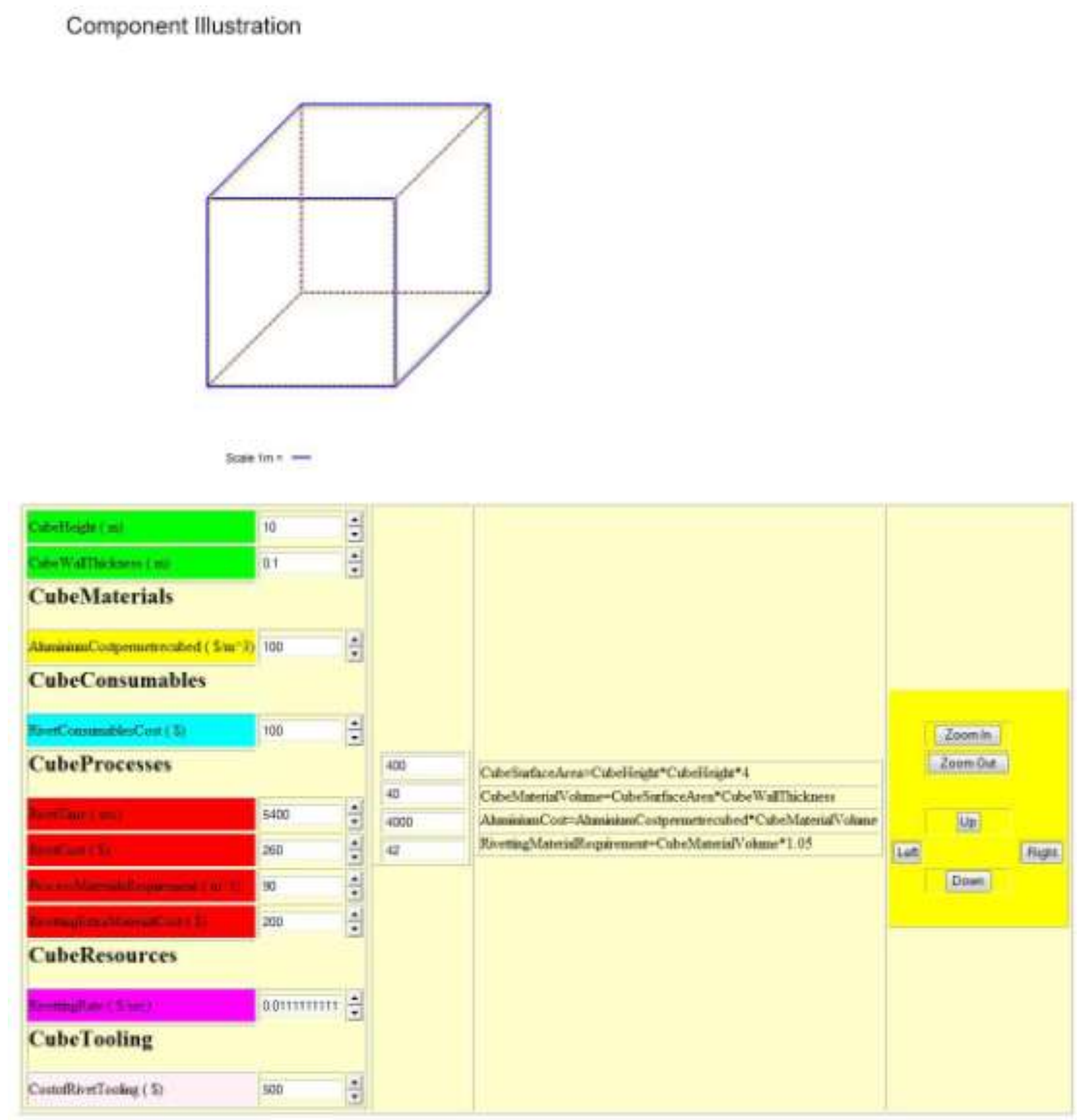

Outputs:
CubeConsumables Cost [100 $]
CubeMaterials Cost [4000 $]
CubeProcesses Cost [260 $]
CubeTooling Cost [500 $]
Grow Tree
Recalc
Cut Tree
Cube
CubePartDefinition [[400 m^2,40 m^3,400 m^2,40 m^3],10 m,0.1 m,10 m,0.1
Cube Derived Values [400 m^2,40 m^3,400 m^2,40 m^3]
Cube Surface Area 400 m^2
Cube Height 10 m
Cube Material Volume 40 m^3
Cube Surface Area
Cube Wall Thickness 0.1 m
Cube Height
Cube Wall Thickness
Assign CubeM
CubeMaterials [[100 $/m^3,4000 $]]
Aluminium [100 $/m^3,4000 $]
Aluminium Cost per metre cubed 100 $/m^3
Aluminium Cost 4000 $
Aluminium Cost per metre cubed
Cube Material Volume
Assign CubeC
CubeConsumables [[100 $]]
RivetConsumables [100 $]
RivetConsumablesCost 100 $
Assign CubePr
CubeProcesses [[5400 sec,260 $,42 m^3,90 m^3,200 $]]
Rivet [5400 sec,260 $,42 m^3,90 m^3,200 $]
Rivet Time 5400 sec
Rivet Time
Rivet Cost 260 $
Rivetter Rate per Hour 0.01111 $/sec
Rivetting Extra Material Cost 200 $
Cube Materials Cost 4000 $
Aluminium Cost
Steel Cost 0
Rivetting Material Requirement 42 m^3
Cube Material Volume
Process Materials Requirement 90 m^3
Rivetting Material Requirement
Weld Material Requirement 48 m^3
Cube Material Volume
Rivetting Extra Material Cost
Assign CubeR
CubeResources [[[0.01111 $/sec]]]
Process Time Rates [[0.01111 $/sec]]
Rivetting Rate [0.01111 $/sec]
Rivetter Rate per Hour
CubeCost [[4000 $],[100 $],[260 $],[500 $]]
CubeMaterials Cost [4000 $]
Cube Materials Cost
CubeConsumables Cost [100 $]
CubeConsumablesCost 100 $
RivetConsumablesCost
WeldConsumablesCost 0
CubeProcesses Cost [260 $]
Process Total Cost 260 $
Rivet Cost
Weld Cost 0
CubeTooling Cost [500 $]
CubeToolingCost 500 $
Cost of Rivet Tooling 500 $
Cost of Welding Tooling 0
Assign CubeT
CubeTooling [[500 $]]
RivetTooling [500 $]
Cost of Rivet Tooling

Figure 31. Cube model example - illustrates choice of process etc.

In this example[29], aluminium was chosen as the material, and riveting was chosen as the process. This example also illustrates how the Vanguard System modelling tool automatically combines units appropriately.

Figure 32 shows the cube translated and visualised using SVG (Scalable Vector Graphics).

[29]Hale, P. (2009) *Component Definition* [online]. Available from: http://www.cems.uwe.ac.uk/~phale/SVGCubeExample/CubePartDefinitionwithCosts.htm [Accessed 25 July2011].



CubeHeight (m) 10

CubeWallThickness (m) 0.1

CubeMaterials

AluminiumCostpermetrecubed ($/m^3) 100

CubeConsumables

RivetConsumablesCost ($) 100

CubeProcesses

RivetTime (sec) 5400

RivetCost ($) 260

ProcessMaterialsRequirement (m^3) 90

RivetingExtraMaterialCost ($) 200

CubeResources

RivettingRate ($/sec) 0.0111111111

CubeTooling

CostofRivetTooling ($) 500

400 CubeSurfaceArea=CubeHeight*CubeHeight*4

40 CubeMaterialVolume=CubeSurfaceArea*CubeWallThickness

4000 AluminiumCost=AluminiumCostpermetrecubed*CubeMaterialVolume

42 RivettingMaterialRequirement=CubeMaterialVolume*1.05

Figure 32. Translation to SVG Visualisation

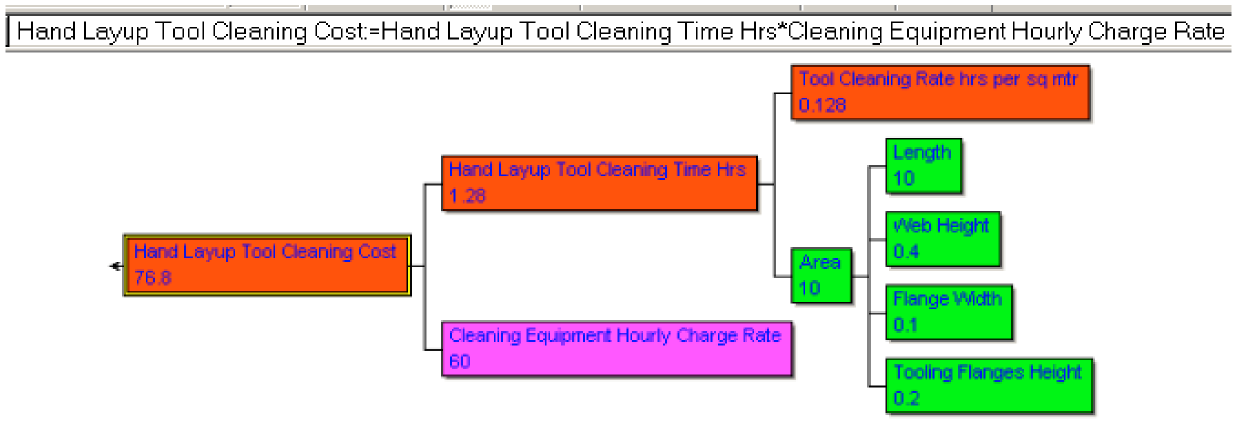
Next the implementation of this research is illustrated with the more complex example of an aircraft wingbox, using the same approach.

6.4 Implementation Results and Benefits

6.4.1 Modelling Tools

Improving and Building On End-User Interaction and Ease of Use

Figure 33 shows how the modelling system created for this thesis can automatically construct and represent a branch in the tree, visualise an equation and calculate a result (into Vanguard System). The information was translated from an ontology, as described in section 3.3.1. Red nodes represent processes, green nodes represent the part definition and magenta nodes represent resources. This illustrates how 3 taxonomies have been automatically linked because they are needed in this calculation. In this prototype, hundreds of calculations have been related to each other. This example illustrates that ‘Area’ was calculated, and that this becomes part of the tree for the ‘Hand Layup Tool Cleaning Cost’, which in turn is passed into other calculations. Hundreds of calculations using information from all the taxonomies are linked as required in this costing example.


Hand Layup Tool Cleaning Cost:=Hand Layup Tool Cleaning Time Hrs*Cleaning Equipment Hourly Charge Rate

Hand Layup Tool Cleaning Cost
76.8
Hand Layup Tool Cleaning Time Hrs
1.28
Cleaning Equipment Hourly Charge Rate
60
Tool Cleaning Rate hrs per sq mtr
0.128
Area
10
Length
10
Web Height
0.4
Flange Width
0.1
Tooling Flanges Height
0.2

Figure 33. Ontology to Model Conversion

Costs and times are fictitious and units were not used in Figure 33. Units can be switched on or off in the Vanguard Studio modelling tool. In an ontology based application for multiple use, agreed standardised representation of units would need to be defined, as discussed by Gruber (1993b) in section 2.3.1.

For the prototype to be extended and applied for practical use, each taxonomy/sub ontology was filled with a structured tree representation of expert’s knowledge in the form of classes, values, and equations. A costing tree can be automatically produced from these taxonomies. Equations created by domain experts, together with choices made by users of the decision support software, determine how these taxonomies are linked for a particular costing. The costing tool user then determines which costing equations are used, by choosing options on dialogue forms. These choices are made whenever multiple solutions are available. The benefit of this approach is that the user interface and calculations will be changed automatically to reflect any changes in the model. So if the problem to be modelled changes, only the information that defines the model needs updating by the user, the user interface and calculation engine will change in response.

6.4.2 Spreadsheets

Improving and Building On Information Structuring and Visualisation

It is particularly important to target modelling that is too complex for use of spreadsheets, and enable sharing of information via an ontology. To achieve this it is necessary to tackle difficulties in maintaining, extending, and reusing spreadsheet type models. Section 3.3 examined possibilities for visual modelling/programming extensions to spreadsheet type formulae based modelling. Section 4.3 explained the need for improving the structuring and visualisation capabilities of spreadsheets to aid collaboration. 5.1.1 detailed the problems that this could help resolve. So such extensions were tested with examples (shown in 5.4) for web based spreadsheet collaboration. The examples in this chapter demonstrate building of a collaborative system that visualises formula in each of the 3 translation steps. So the modelling capabilities of spreadsheets were provided by a decision support system that reads formulae from an ontology in order to aid interoperability and collaboration. The results from formulae calculation are displayed in various ways to show a range of interfaces that could aid interactivity, and a range of translations to different applications, either built for this research or already existing applications (to demonstrate interoperability). Such variation of the interface and collaboration with other applications could be added to web spreadsheet applications such as Google spreadsheets (discussed in 4.5.1). Google Spreadsheets was evaluated for this purpose and online spreadsheet tools including Softartisans and Java based spreadsheets (section 5.4). However, more customised software enabled web modelling that allowed for many different ways of representing and interacting with models, and that did not necessarily have the spreadsheet look and feel; this provided more scope for development of alternative ways of interaction with end-users. This also ensured that if the ontology in Step 1 changes, the models and visualisations in Step 2 and 3 will update automatically, so models can be added just by adding information to the database or ontology. This represents progress towards the goal of allowing User Driven Modelling.

6.4.3 Ontology Tools

Improving and Building On Modelling Capability and End-User Interaction

Figure 34 demonstrates the ontology creation and translation to the modelling tool made for Step1and 2. This shows an equation defined by the user and demonstrated as a relationship tree. Figure 34 is the top half of Figure 25 in order to show Step 1 and 2. This was achieved using an ontology tool (Protégé), and this definition is read directly by decision support software (Vanguard System) that visualises the information and colour codes it. This is a small part of a more complex example, and shows one part of one sub-ontology, that of Parts. For a more complex example a higher level user interface would be required to enable users to define the problem, and a translation step to the computer readable model. The software can translate the source model into a program and calculate results. The result program is then translated again into open standard languages such as XML and to Java for human friendly visualisations viewable as web pages.

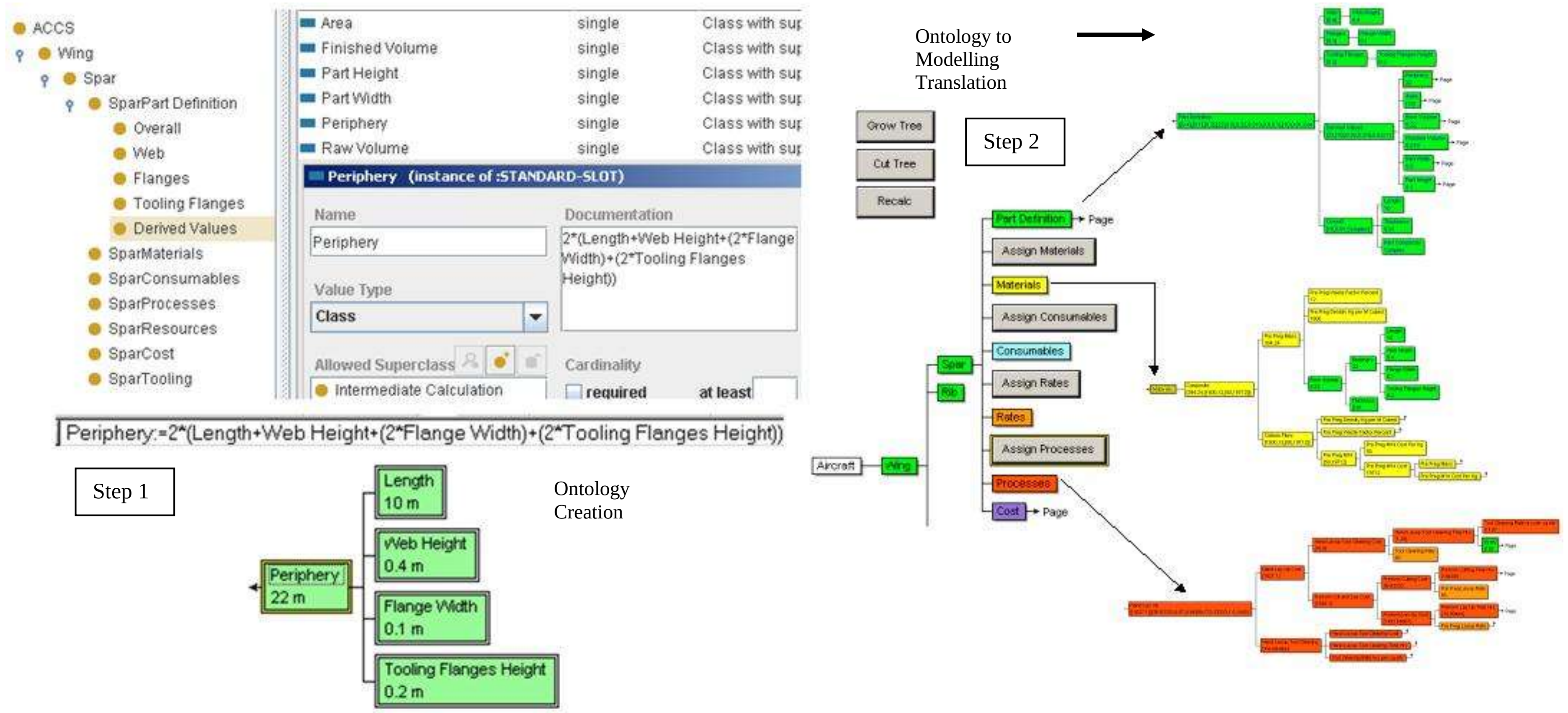


Figure 34. Translation Process Implementation

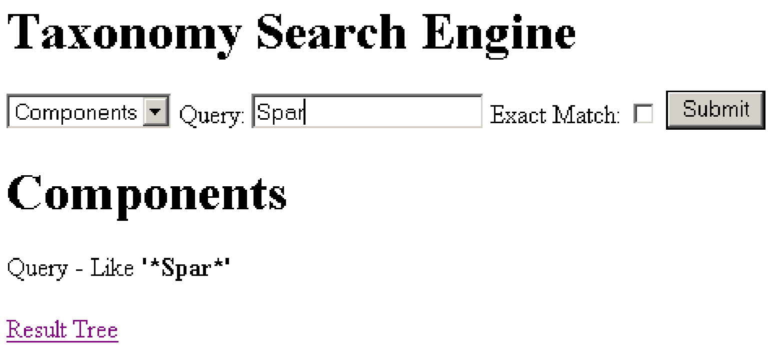
6.4.4 Semantic Web/Web 2.0

Improving and Building On Modelling Capability and End-User Interaction

This section develops further the research illustrated in section 5.5 for using Semantic Web languages and accessing them with applications built to enable end-users to create and interact with Semantic Web information sources.

6.4.4.1 Semantic Search

The example below (Figure 35) illustrates how it is possible to enable refining a search by visualising all the items present in sub-categories of the main category found in the search. McGuinness (2003) explains how ontologies support this functionality and calls this ‘generalization/specialization’ of information. Uschold and Gruninger (2004) describe “ontology-based search” as “Ontology used for concept-based structuring of information in a repository”; and describe the benefit of this as “better information access” This aids the thesis objectives of ease of use and sharing of information. The use of open standards for representing information makes it possible to enable searches that understand the semantics of the information and so can track all of the relationships between items. Figure 35 illustrates the interface for making a search. In this example the user wants to retrieve all the information related to a spar.


Taxonomy Search Engine

Components ▾ Query: Spar Exact Match: ☐ Submit

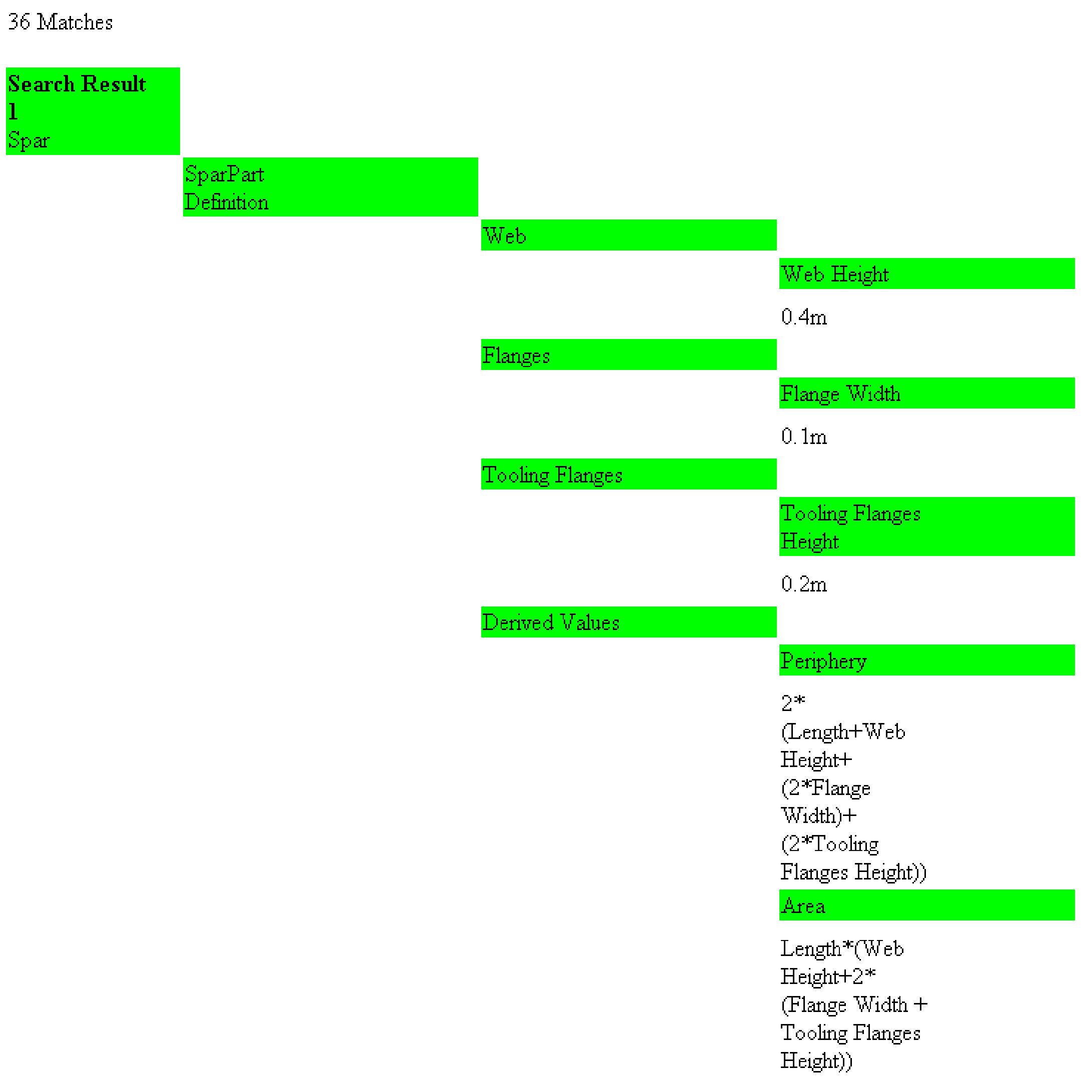
Components

Query - Like '***Spar***'

Result Tree

Figure 35. Semantic Search interface

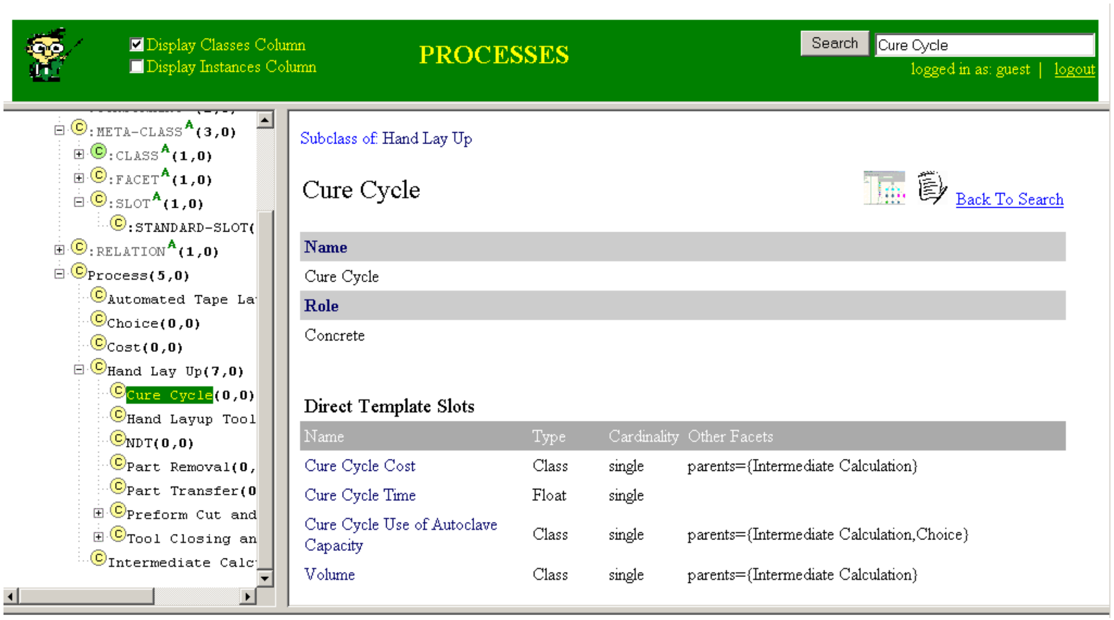
The result is shown as a series of trees for each item that contains the word spar. Each keyword match is the root of a tree. Each tree shows the item found and all its children and attributes. Figure 36 shows an image of the top part of the results, this is part of the branch for the first item returned.


36 Matches

Search Result 1
Spar
- SparPart Definition
  - Web
    - Web Height
      0.4m
  - Flanges
    - Flange Width
      0.1m
  - Tooling Flanges
    - Tooling Flanges Height
      0.2m
  - Derived Values
    - Periphery
      2*(Length+Web Height+(2*Flange Width)+(2*Tooling Flanges Height))
    - Area
      Length*(Web Height+2*(Flange Width + Tooling Flanges Height))

Figure 36. Results from semantic search

The information is held in linked and related taxonomies/sub ontologies so it is not HTML that is being searched but the taxonomy itself. Because the information is held in a structured way, it is much more likely that searchers will find what they are looking for, because the search can follow the relationships represented in the taxonomy. One of the key objectives of Semantic Web research and Web 2.0 is to make this kind of search possible over the web as a whole. The Semantic Web is a longer-term vision for managing information over the web and Web 2.0 is the shorter-term practical implementation of techniques, which can ease current information search and management problems. A web interface has been developed for Protégé (WebProtege[30]). An example of the use of this is Figure 37 where a search is made for information on the cure cycle for composites manufacturing. This search is possible as WebProtege has succeeded in providing a web based interface for displaying and searching ontologies, so providing an additional way to enable web access to the test ontologies created for this thesis.



Subclass of: Hand Lay Up

Cure Cycle

Name
Cure Cycle

Role
Concrete

Direct Template Slots

Name	Type	Cardinality	Other Facets
Cure Cycle Cost	Class	single	parents={Intermediate Calculation}
Cure Cycle Time	Float	single	
Cure Cycle Use of Autoclave Capacity	Class	single	parents={Intermediate Calculation,Choice}
Volume	Class	single	parents={Intermediate Calculation}

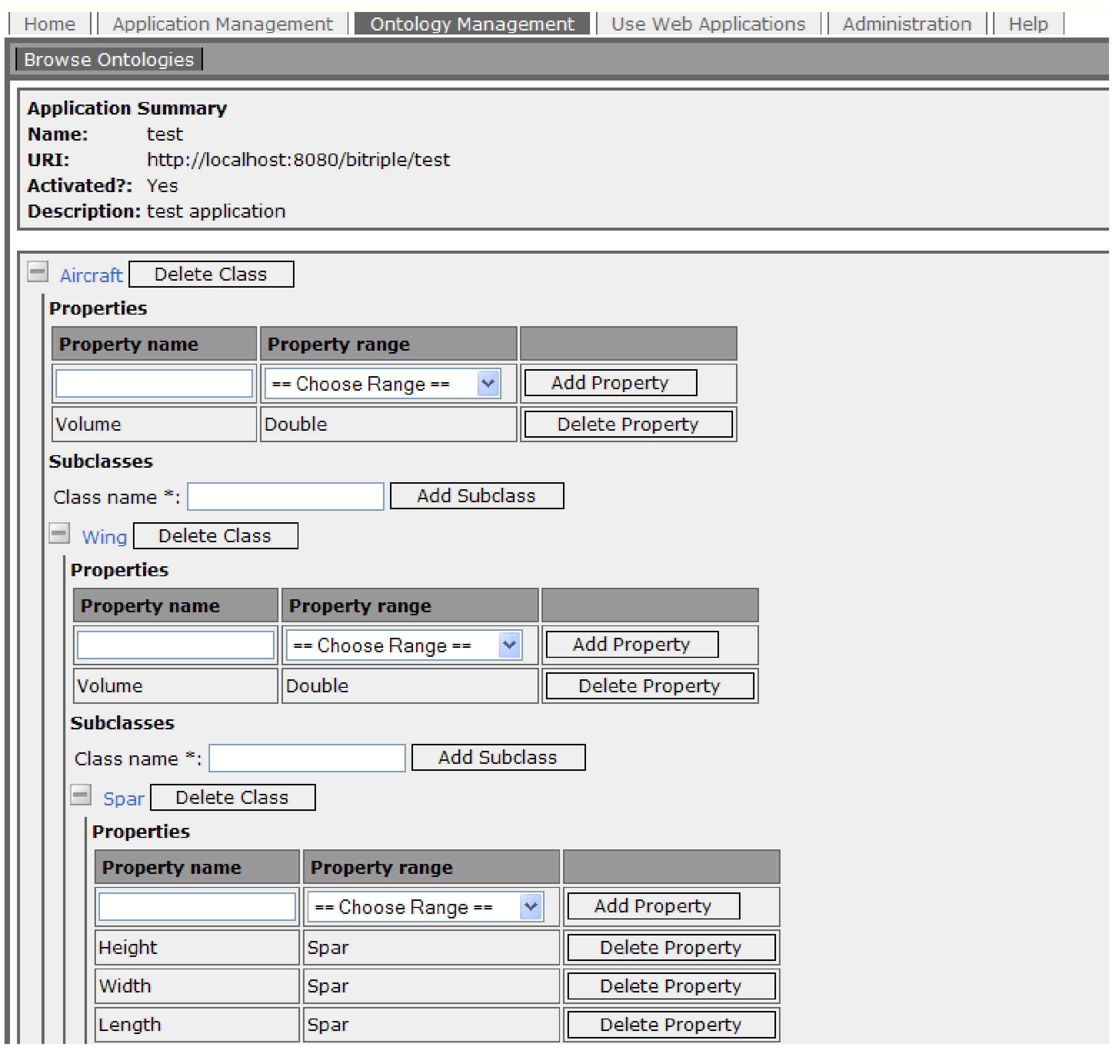
Figure 37. Web Protégé Interface

As discussed in section 3.6.1- 'Semantic Web and Web 2.0 Collaboration', a project was created called Bitriple by Leaver (2008), to enable end-user functionality for this kind of web-based ontology construction and search. The application provides a facility to edit an ontology/ies and instances, and provides tree-based visualisation of the ontology (as shown in Figure 38). This example illustrates creation of an online aircraft wing ontology. Wing component sub/ontologies created using Protégé can be translated for the Bitriple application to be represented as RDF/XML. An application could be built as an extension to Bitriple to perform calculations and modelling using the information stored. This could assist in allowing domain expert end-user programmers/modellers to create models. Such web applications provide an alternative to spreadsheets, and to single computer based programs; and if installed on a network server, such applications can provide a collaborative model development

[30] WebProtege (2011) *Wiki Page* [online]. Available from: http://protegewiki.stanford.edu/index.php/WebProtege [Accessed 22 June2011].

environment. Collaboration can aid people to agree on terminology, and standardisation of calculations used such as for cost rates and currencies. RDF information can be searched with SPARQL[31] (SPARQL Protocol And RDF Query Language), which is used to search the Bitriple application.

A screenshot from the Bitriple application, of ontology creation for an aircraft wing, is shown in Figure 38:-



Application Summary
Name: test
URI: http://localhost:8080/bitriple/test
Activated?: Yes
Description: test application

Aircraft Delete Class

Properties

Property name	Property range	
	== Choose Range ==	Add Property
Volume	Double	Delete Property

Subclasses

Class name *: Add Subclass

Wing Delete Class

Properties

Property name	Property range	
	== Choose Range ==	Add Property
Volume	Double	Delete Property

Subclasses

Class name *: Add Subclass

Spar Delete Class

Properties

Property name	Property range	
	== Choose Range ==	Add Property
Height	Spar	Delete Property
Width	Spar	Delete Property
Length	Spar	Delete Property

Figure 38. Bitriple Ontology Creation Screenshot

6.5 Examples

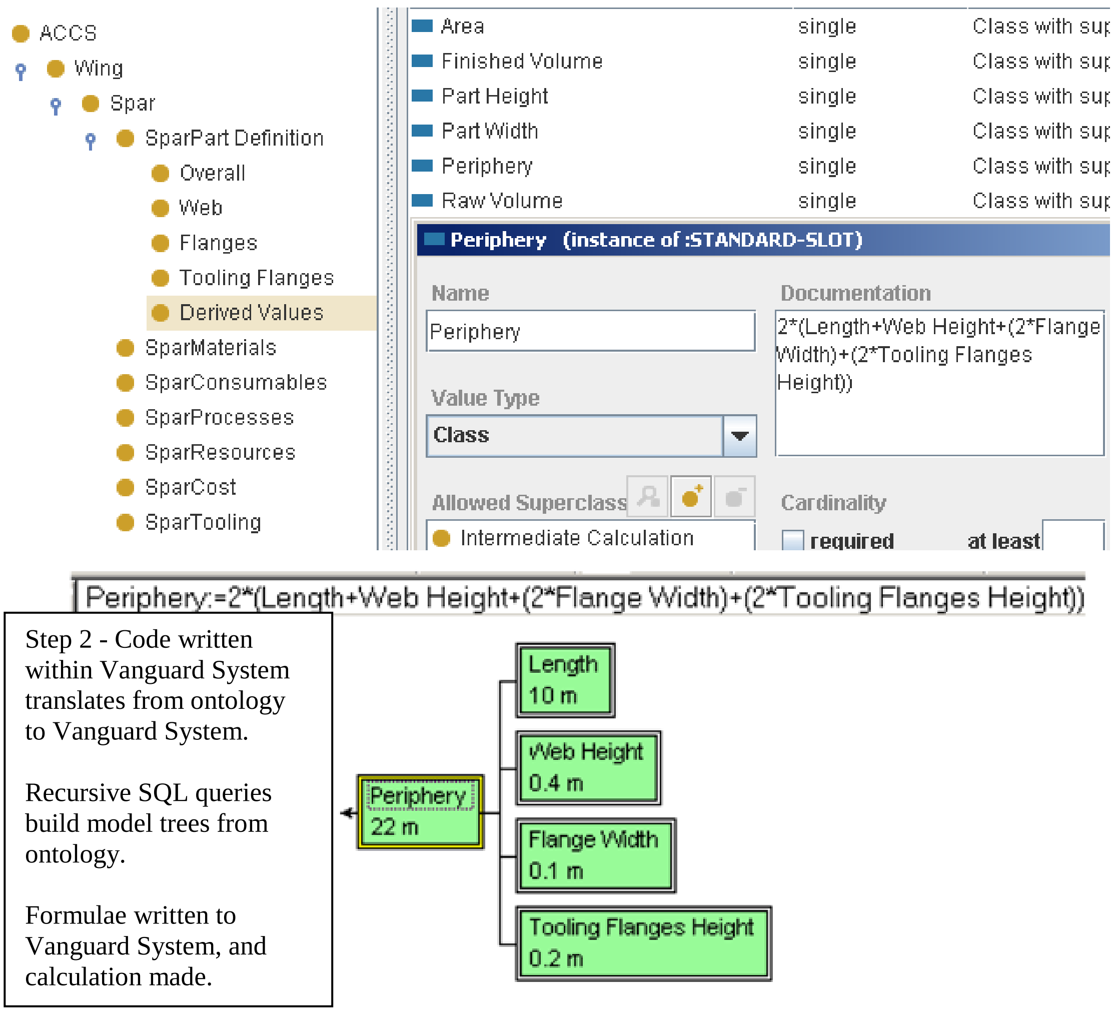
Examples are based on representation of wing components. Spar and associated processes are shown; Ribs, Skins, and Stringers were also modelled.

[31] SPARQL is the query language and protocol for RDF being recommended to the World Wide Web Consortium - (2004) [online]. Available from: http://www.w3.org/TR/rdf-schema/ [Accessed 28 September 2009]. A tutorial has been developed by Dodds (2005) - [online]. Available from: http://www.xml.com/lpt/a/2005/11/16/introducing-sparql-querying-semantic-web-tutorial.html [Accessed 25 July 2011].

6.5.1 Step 1 - Ontology

The purpose of Step 1 is to provide a common ontology from which many models and visualisations in Steps 2 and 3 can be created. As the ontology provides a high level view of information, an editor does not need programming skills to edit the ontology (though easier web-based editing of the ontology would enable wider ontology editing).

Figure 39 is a Vanguard System reproduction of the part definition taxonomy (or sub ontology) from the Protégé ontology. The Vanguard System software adds an extra functionality, which is to calculate, and store the results of equations captured in the Protégé taxonomy. The equation is represented as text in the 'Documentation' field of the 'Periphery' attribute of 'Derived Values' as illustrated in Figure 39 below :-



- ACCS
  - Wing
    - Spar
      - SparPart Definition
        - Overall
        - Web
        - Flanges
        - Tooling Flanges
        - Derived Values
      - SparMaterials
      - SparConsumables
      - SparProcesses
      - SparResources
      - SparCost
      - SparTooling

Area	single	Class with sup
Finished Volume	single	Class with sup
Part Height	single	Class with sup
Part Width	single	Class with sup
Periphery	single	Class with sup
Raw Volume	single	Class with sup

Periphery (instance of :STANDARD-SLOT)

Name: Periphery

Documentation: 2*(Length+Web Height+(2*Flange Width)+(2*Tooling Flanges Height))

Value Type: Class

Allowed Superclass: Intermediate Calculation

Cardinality: required at least

```
Periphery:=2*(Length+Web Height+(2*Flange Width)+(2*Tooling Flanges Height))
```

Step 2 - Code written within Vanguard System translates from ontology to Vanguard System.

Recursive SQL queries build model trees from ontology.

Formulae written to Vanguard System, and calculation made.

Figure 39. Translation of Equation Representation from Protégé to Vanguard System

If choices need to be available to modellers this is specified in the ontology as a default choice with a list of all the possible other choices in a standardised natural language representation, the translator produces a button in Vanguard System in response to this, which lists choices and acts on the user/modeller's selection. These choice/buttons can be represented anywhere in the tree/model as needed.

6.5.2 Step 2 - Modelling Tool

Figure 40 illustrates colour coding showing Parts as green, Materials as yellow, Rates as orange, and processes as red, and how these are combined in calculations, so the calculation branch may contain several colours as required. This gives some idea of the size of the tree created for to test implementation of this research but is still only a mock diagram consisting of tiny parts of the overall tree. Figure 40 also shows how the decision support system tree view of the spar branch from the wingbox cost model is created with information translated from the related taxonomies, colour coded, and where necessary from users' selections (e.g. of materials). The tree, including all the default part definition information for the spar, is produced automatically. The buttons in the tree enable choices to be made by the user about materials, consumables, rates, and processes. Branches are created in response to these choices. The values in the branch nodes can then be changed as required.

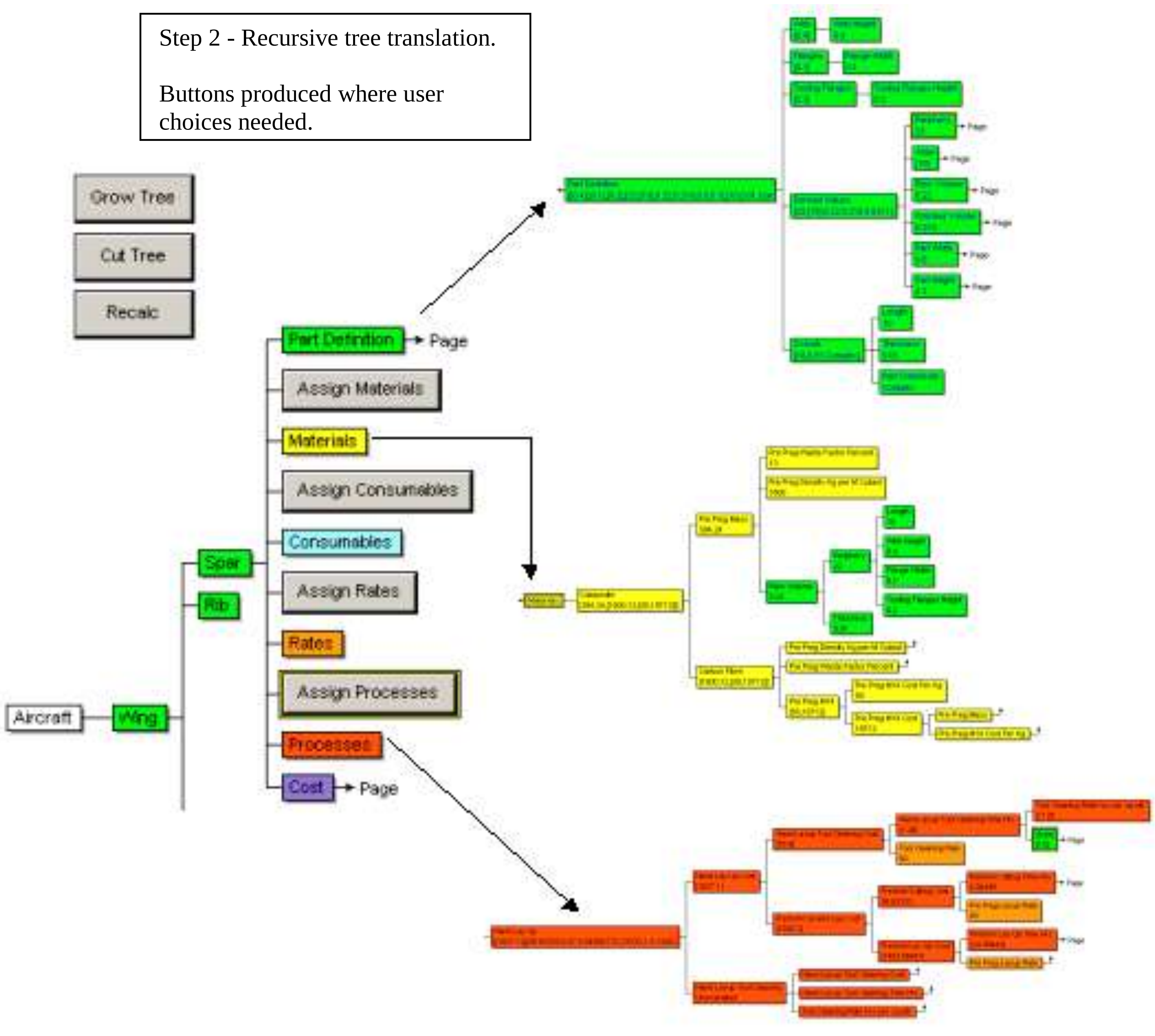


Figure 40. Spar branch automatically created from information source

The user makes choices, so the decision support result tree will be a subset of the information source tree. Choices take a graph/tree and produce a sub graph/tree. Zhao and Liu (2008) are encoding STEP rules and executable statements into OWL and SWRL. Semantic languages such as OWL can represent if-then choices (Elenius, 2005), and some rule based systems have a visual diagrammatic editor which could be used to enable an end-user to represent such problems. In fact if-then choices could have been represented using diagrammatic editing in Protégé's Visual Composer Toolbar[32], so this will be future work (section 8.22).

The trees created have thousands of nodes, Vanguard System visualises large trees by breaking them into individual 'pages' (screens), and indicating with a right arrow where further pages can be viewed. Clicking the 'Part Definition' right arrow will display the corresponding information as illustrated in Figure 41 below. The 'Derived Values' branch contains parameters of the spar that are calculated from the spar dimensions.

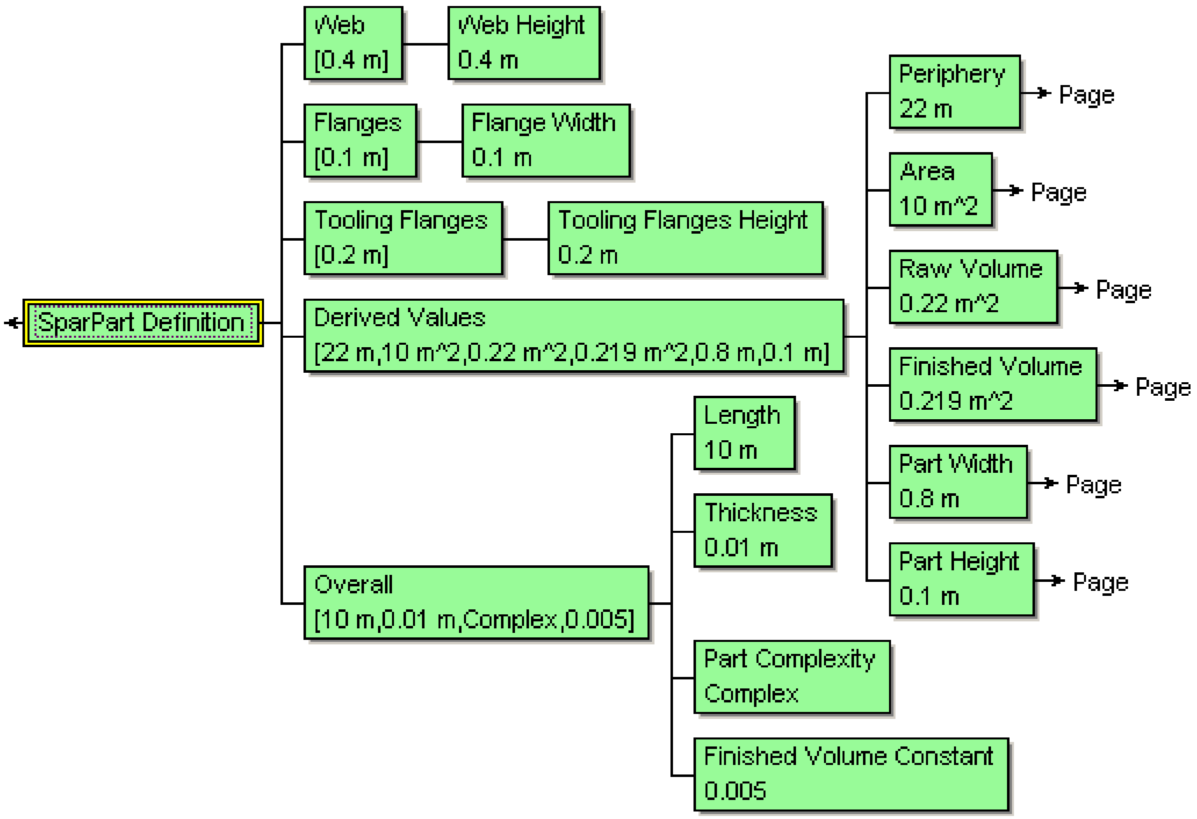


Figure 41. Part Definition Branch

Different types of information indicated by colour coding may be combined in a calculation. This is illustrated in Figure 42. 'Pre Preg Mass' is coloured yellow as it and its attributes are categorised as material attributes, 'Raw Volume' is a part attribute, and so coloured green.

[32] Scicluna, J. (2009) OWL-S Editor User Manual, [online]. Available from: http://staff.um.edu.mt/cabe2/supervising/undergraduate/owlseditFYP/OWL-S_Editor_User_Manual.pdf [Accessed 25 July 2011].

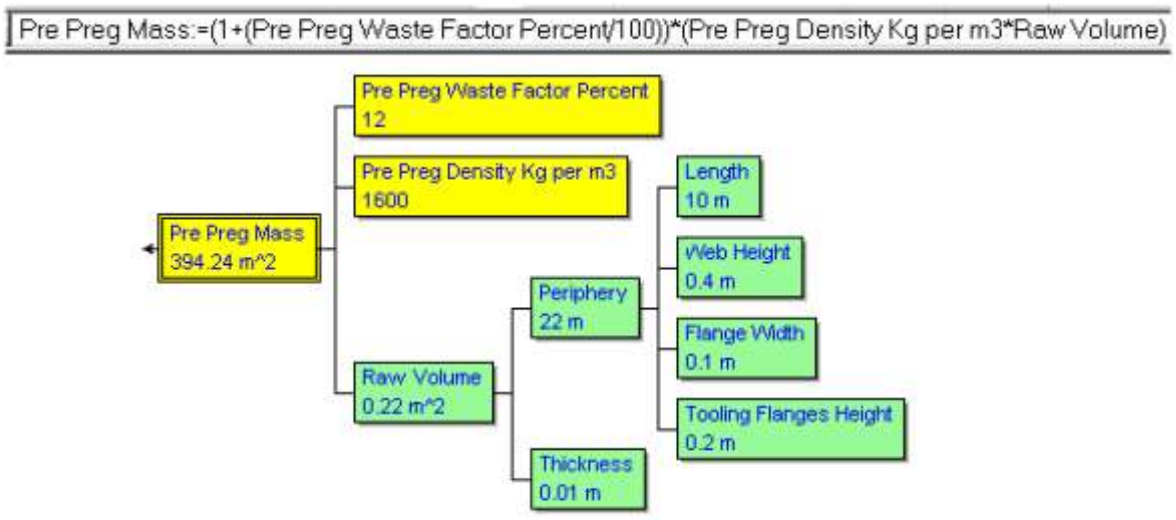

Pre Preg Mass:=(1+(Pre Preg Waste Factor Percent/100))*(Pre Preg Density Kg per m3*Raw Volume)

Figure 42. Pre Preg Mass Calculation

Figure 43 shows a branch produced in response to the user’s choice of the ‘Hand Lay Up’ manufacturing process, and sub choices for how this manufacturing process is carried out. This illustrates the link of process information to part and resource rate information. The process nodes are linked to the resource rate nodes ‘Cleaning Equipment Hourly Charge Rate’, and to the Part attribute ‘Area’ as these values are all required by process equations. Processes are represented in red, resource rates in violet, and part definition in green. A multiple (faceted) classification structure is used in the decision support system, so a child can have multiple parents, this means the tree is also a graph/network representation. The child is shown fully under its first parent but the child may appear again when it has other parents. An upward pointing arrow on the node’s right hand side designates that information is used in many locations in the tree, and its value is used under each of its parents.

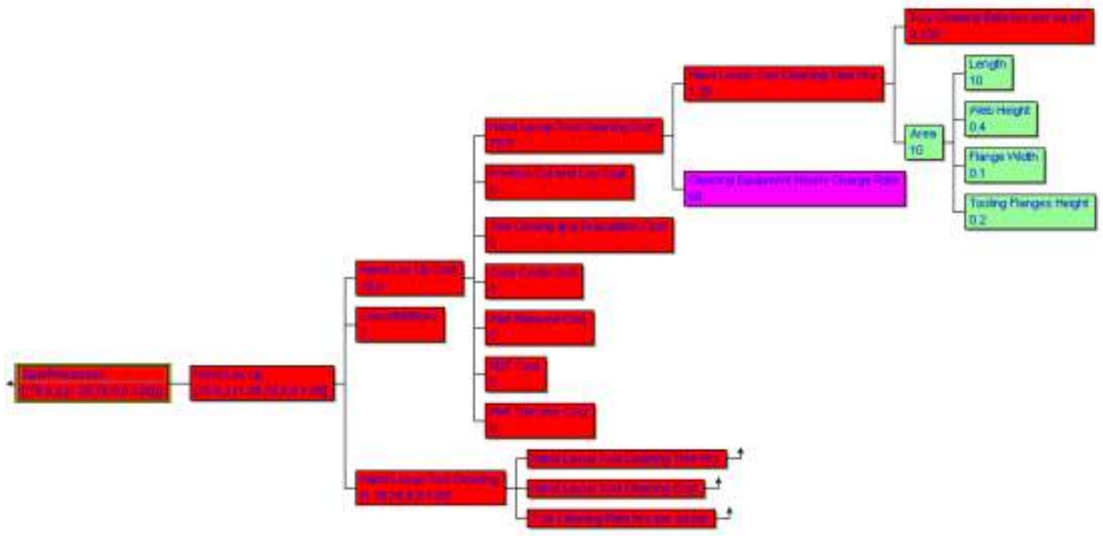


Figure 43. Hand Layup Calculation

When the user has finished making selections the cost break down of the spar will be displayed as a colour coded branch. Figure 44 illustrates how different branches in the tree allow a user to drill down to more detailed views of the cost, consumables are shown in light blue.

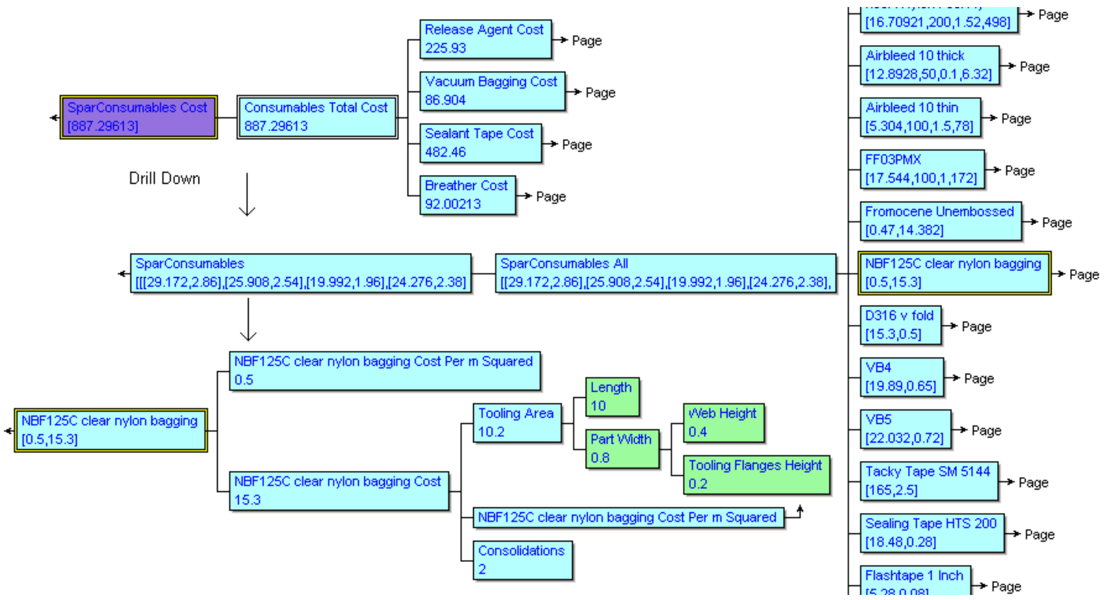


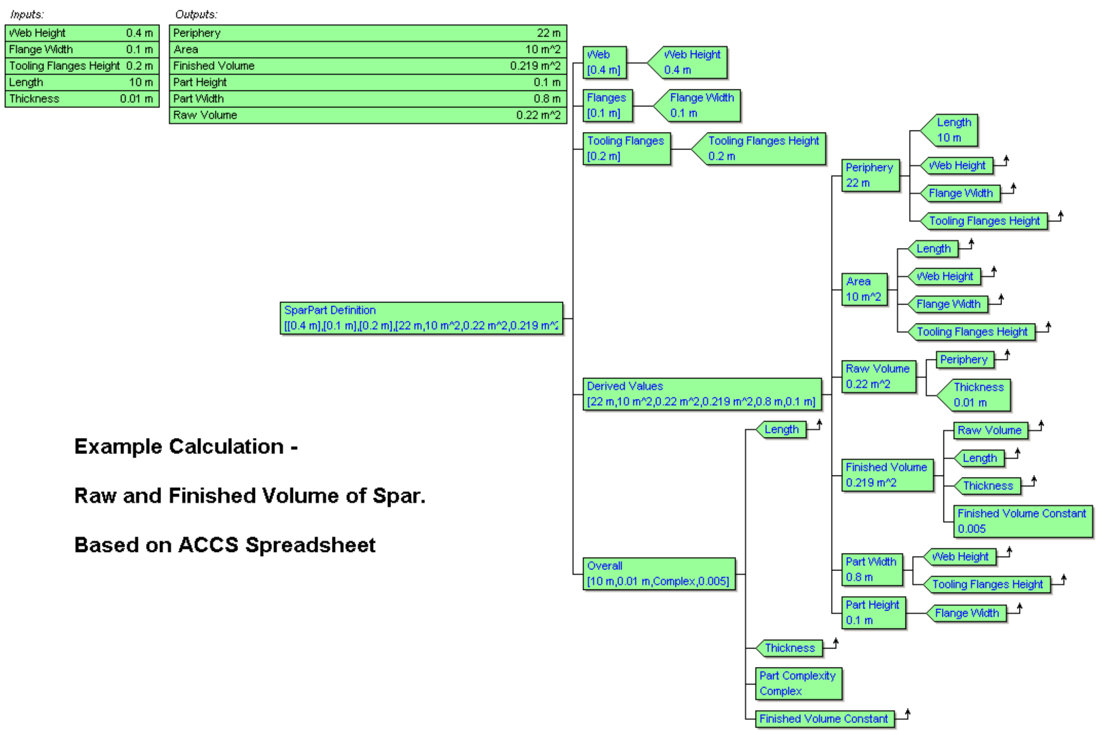
Figure 44. Using the tree view for cost drill down

6.5.3 Step 3 - Interactive Visualisation

Step 3 involves producing interactive visualisations that make use of different ways of visualising the same information. This was made possible by use of the ontology in Step 1 and interpretation/translation of the ontology in step 2. This also enabled use of colour coding to show different types of information.

Figure 45[33] shows the spar volume calculation uploaded to the web. This illustrates that an ontology defined in Step 1, has had calculation and translation performed in Step 2 and can now be visualised in Step 3.

[33] Hale, P. (2009) *Spar Model* [online]. Available from: http://wiki.vanguardsw.com/bin/browse.dsb?det/Engineering/Aerospace/Wing%20Spar%20Translated%20from%20Protege%20Taxonomy [Accessed 25 July 2011].



Inputs:

Web Height	0.4 m
Flange Width	0.1 m
Tooling Flanges Height	0.2 m
Length	10 m
Thickness	0.01 m

Outputs:

Periphery	22 m
Area	10 m^2
Finished Volume	0.219 m^2
Part Height	0.1 m
Part Width	0.8 m
Raw Volume	0.22 m^2

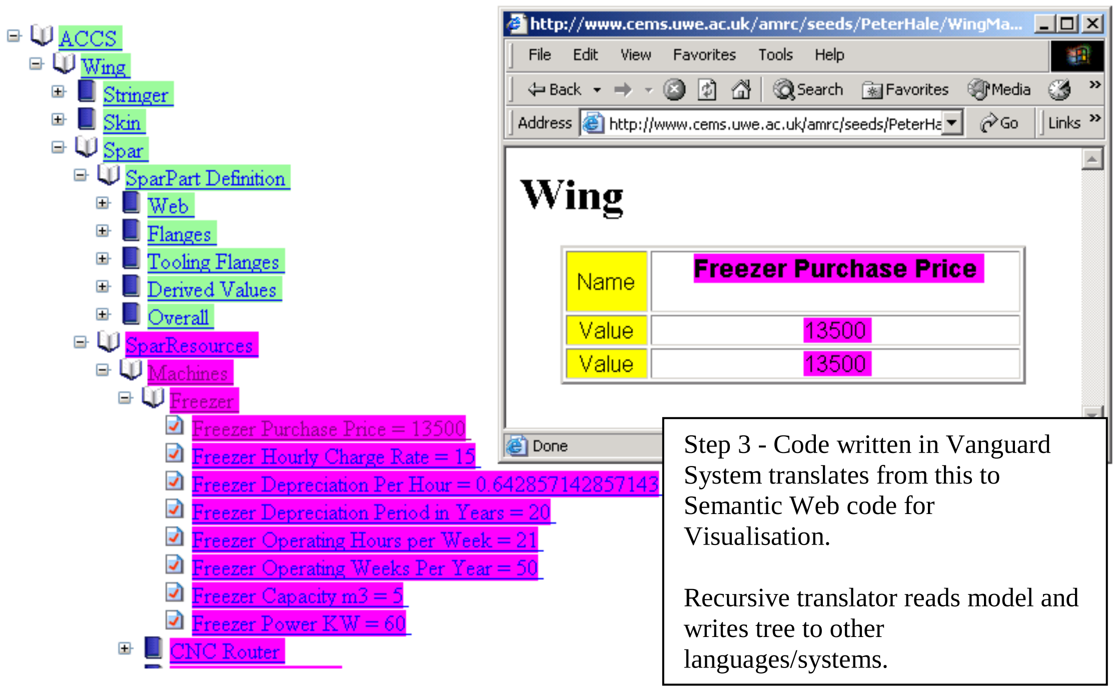
Example Calculation -

Raw and Finished Volume of Spar.

Based on ACCS Spreadsheet

Figure 45. Spar Volume calculation model running on the web

This approach to colour coding, of illustrating type with colour, is related to this work on using colour to indicate cost, time, and uncertainty (Bru et al., 2002), (Bru et al., 2003), (Bru et al., 2004). The need for sophisticated visualisation, and tools such as developed in this thesis (and that of Bru) became apparent when models such as the wingbox spreadsheet delivered volumes of information, which could not be easily absorbed and analysed in their textual form. The research in this thesis focuses on colour coding for indication of knowledge/information type, such as distinguishing between parts, processes, materials cost rates etc and illustrating which of these items are used in each calculation. Brus' work concentrates on using colour, size, and shape to illustrate cost, time, and uncertainty. Both approaches were used in combination within the DATUM project (Scanlan, 2006). The information visualisation for this thesis is compatible with a wide range of standard formats, thus allowing it to accept input from numerous tools and ontologies. The information is then further translated to various formats including XML and SVG for web visualisation and information exchange. This is shown in Figure 46 where XML is used to exchange information with a web page that is then able to colour code this information, display it and let the user interact with it.


http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/WingMa...
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.cems.uwe.ac.uk/amrc/seeds/PeterHa Go Links
Done

- ACCS
  - Wing
    - Stringer
    - Skin
    - Spar
      - SparPart Definition
        - Web
        - Flanges
        - Tooling Flanges
        - Derived Values
        - Overall
      - SparResources
        - Machines
          - Freezer
            - Freezer Purchase Price = 13500
            - Freezer Hourly Charge Rate = 15
            - Freezer Depreciation Per Hour = 0.642857142857143
            - Freezer Depreciation Period in Years = 20
            - Freezer Operating Hours per Week = 21
            - Freezer Operating Weeks Per Year = 50
            - Freezer Capacity m3 = 5
            - Freezer Power KW = 60
          - CNC Router

Wing

Name	Freezer Purchase Price
Value	13500
Value	13500

Step 3 - Code written in Vanguard System translates from this to Semantic Web code for Visualisation.

Recursive translator reads model and writes tree to other languages/systems.

Figure 46. Web page showing translated XML displayed as interactive web application

Figure 47 demonstrates the ontology translated via Step 2 into XML for Step 3 visualisation in Flash[34]. This creates a tree with a three dimensional look, colour and shading, and interactive repositioning of nodes to make it intuitive and assist in navigation. When a node is chosen, this is moved to the centre of the display and all the other nodes are moved or rotated to position themselves in relation to it.

[34] Rhodes, G., Macdonald, J., Jokol, K., Prudence, P., Aylward, P., Shepherd, R., Yard, T., 2002. A Flash Family Tree, *In: Flash MX Application and Interface Design Flash MX Application and Interface Design*. ISBN:1590591585. [online]. Available from: http://www.friendsofed.com/book.html?isbn=1590591585 [Accessed 22 June 2011].

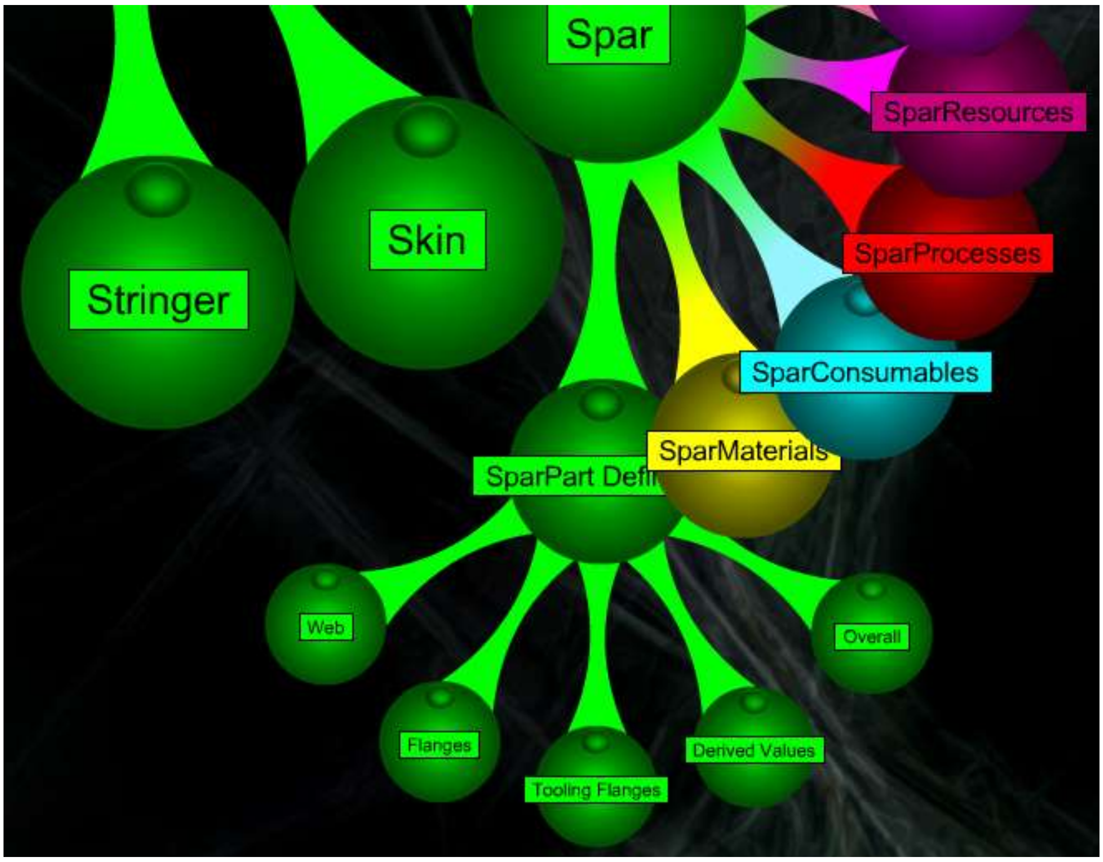
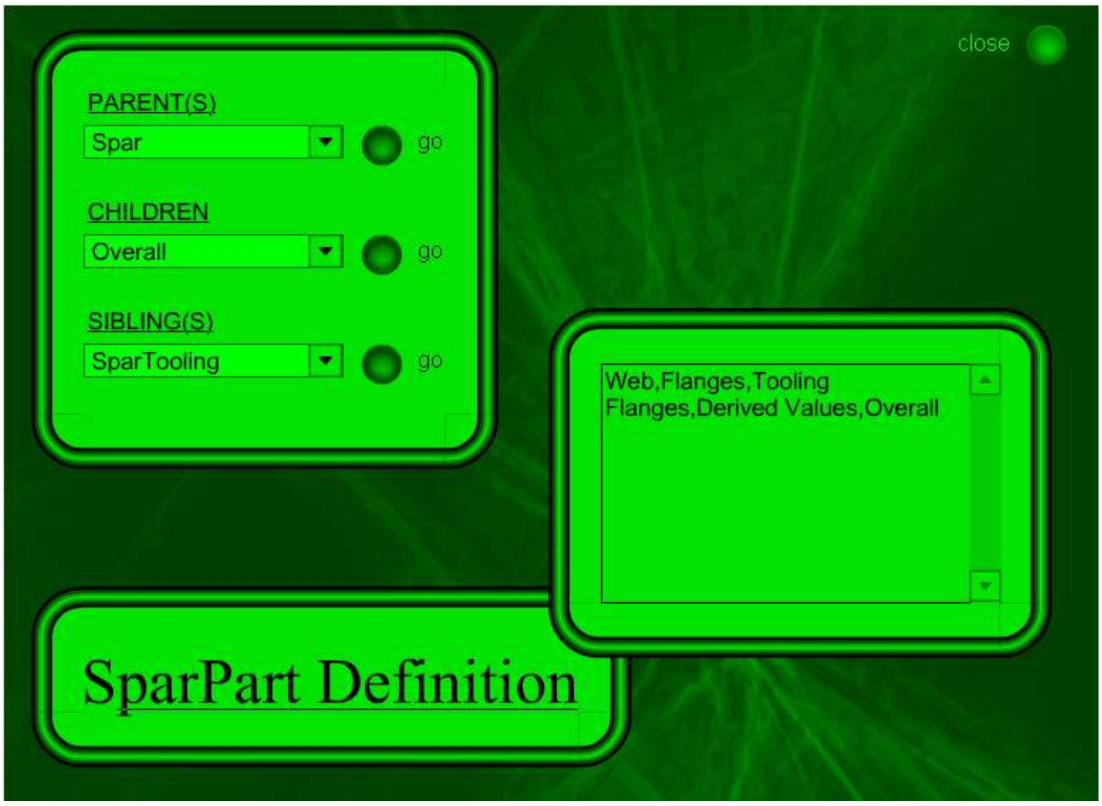


Figure 47. Flash interface for navigating exported XML tree

Figure 48 shows the view resulting from choosing the ‘SparPart Definition’. This shows the parents, children, siblings, and contents of that node. It also allows navigation to any of the related nodes.



Web,Flanges,Tooling
Flanges,Derived Values,Overall

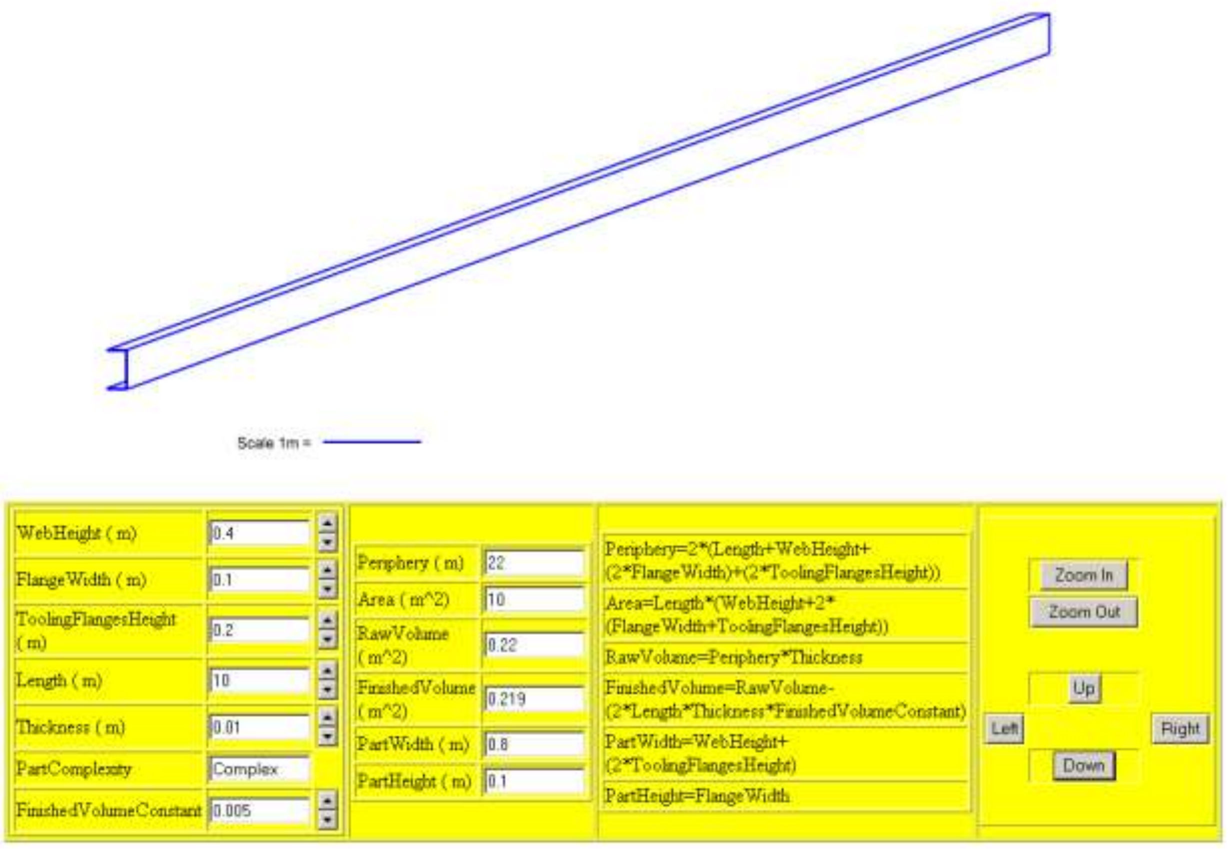
Figure 48. Flash viewing of Spar Part Definition node

6.5.3.1 Step 3 - CAD Style Interactive Visualisation

If attributes are provided in Step 1 or 2 it is possible to translate the tree representation into a dynamic CAD type representation for Step 3. This information allows the system to decide where to draw the lines to represent the attributes of the component. These attributes can then be changed by the user in an interactive CAD type representation that allows for visualisation of the component and its attributes. This capability can be used for enabling better understanding of a component's attributes/properties, and for modelling and calculation.

Figure 49[35] below is produced via an automated conversion from a tree representation of the spar component. The interface demonstrates modelling of information within a browser; 'Periphery', 'Area', 'Raw Volume', 'Finished Volume', 'Part Width' and 'Part Height' are all calculated dynamically. This calculation is in response to changes the user makes to the attributes on the left; as these changes are made the diagram changes in response. It would also be possible to reverse the translation by taking this interface and converting it to a tree or graph representation of the component.

[35] Hale, P. (2009) *Interactive SVG Examples* [online]. Available from: http://www.cems.uwe.ac.uk/~phale/InteractiveSVGExamples.htm [Accessed 25July2011].



WebHeight (m)	0.4
FlangeWidth (m)	0.1
ToolingFlangesHeight (m)	0.2
Length (m)	10
Thickness (m)	0.01
PartComplexity	Complex
FinishedVolumeConstant	0.005

Periphery (m)	22
Area (m^2)	10
RawVolume (m^2)	0.22
FinishedVolume (m^2)	0.219
PartWidth (m)	0.8
PartHeight (m)	0.1

```
Periphery=2*(Length+WebHeight+(2*FlangeWidth)+(2*ToolingFlangesHeight))
Area=Length*(WebHeight+2*(FlangeWidth+ToolingFlangesHeight))
RawVolume=Periphery*Thickness
FinishedVolume=RawVolume-(2*Length*Thickness*FinishedVolumeConstant)
PartWidth=WebHeight+(2*ToolingFlangesHeight)
PartHeight=FlangeWidth
```

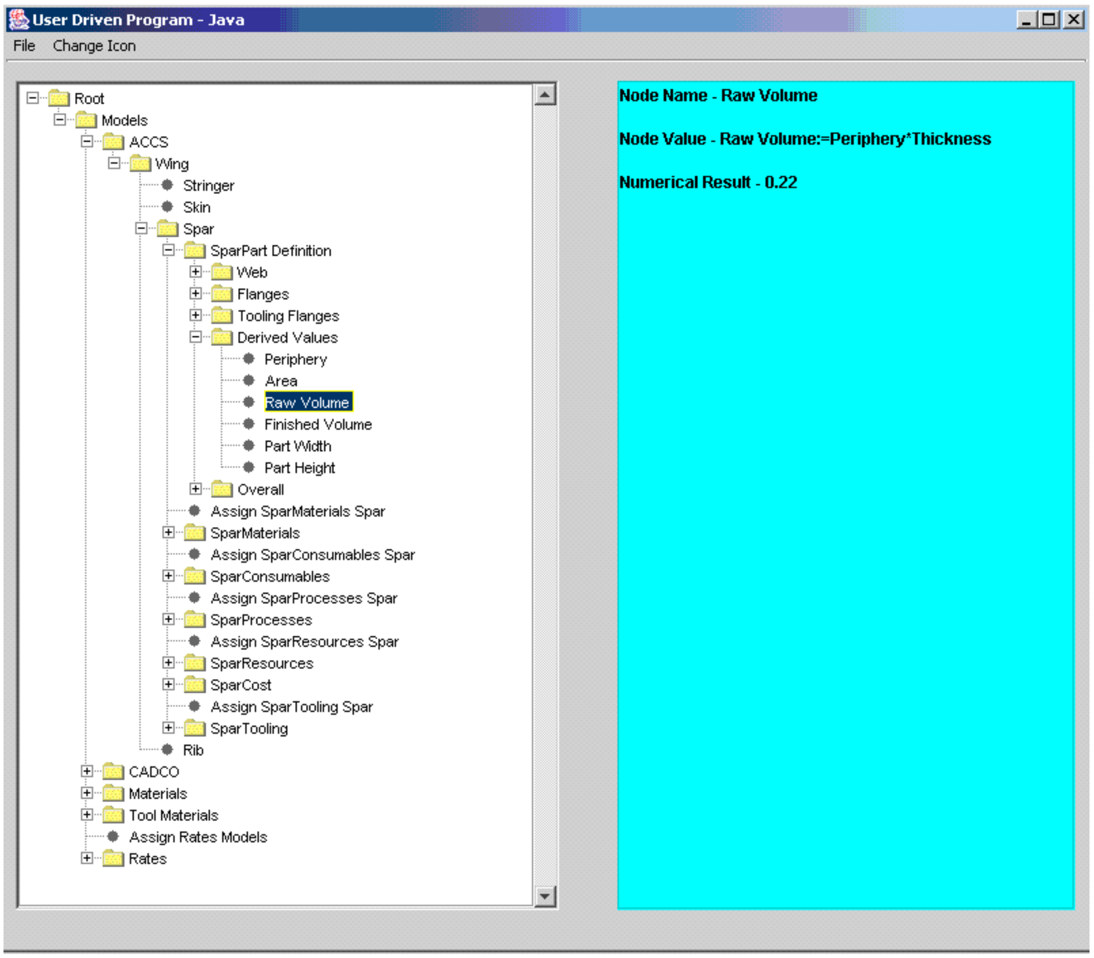
Figure 49. Interactive Spar Diagram (SVG)

Figure 49 was created from a tree based ‘analogical’ (Guibert et al., 2004) representation, translated to a view that used a CAD based analogy, and could also be translated back to the tree representation, and to a ‘fregean’ representation that is required for computer code.

6.5.3.2 Translation to Program/Meta-Program Code

This stepped translation methodology can also be used for translation from ontologies to program and meta-program code in various languages.

Figure 50 shows the Vanguard System tree translated into Java and visualised :-



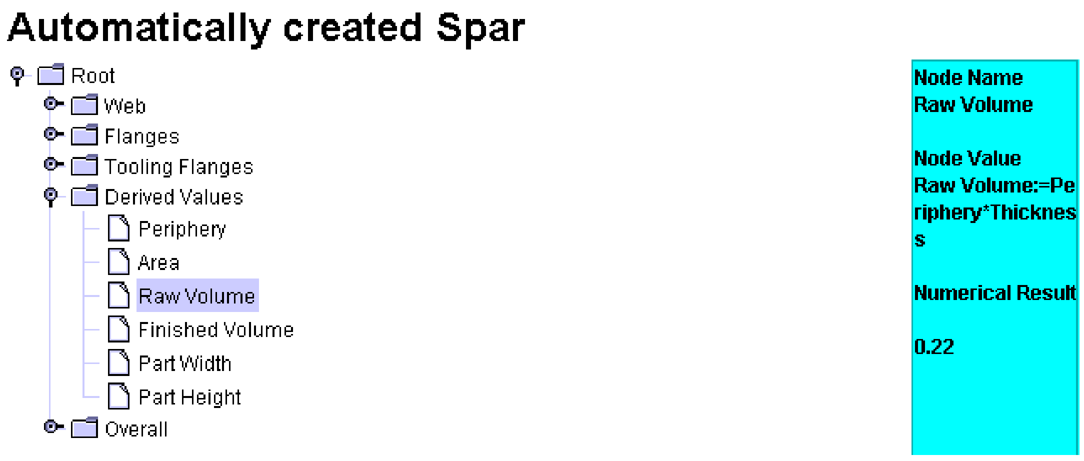
Node Name - Raw Volume

Node Value - Raw Volume:=Periphery*Thickness

Numerical Result - 0.22

Figure 50. Translation from decision tree into Java

This can also be created and visualised as a Java web applet; Figure 51 shows this :-



Node Name
Raw Volume

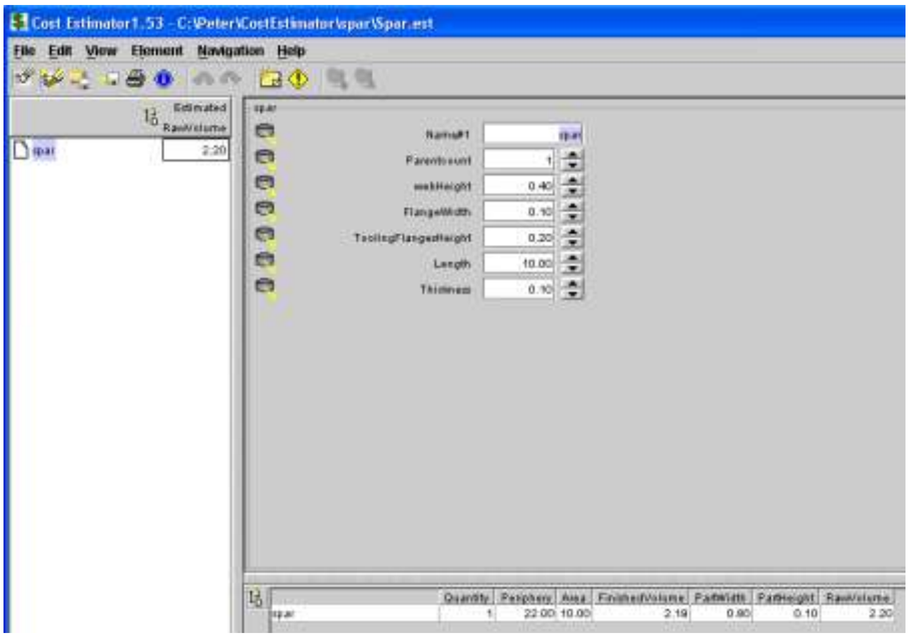
Node Value
Raw Volume:=Periphery*Thickness

Numerical Result

0.22

Figure 51. Translation from decision tree into Java web Applet

A translation into the Java based Cost Estimator System[36] was also created. This is an example of translating to an external application; Figure 52 illustrates this :-



	Estimated RawVolume
spar	2.20

spar

Name#1	spar
ParentCount	1
webHeight	0.40
FlangeWidth	0.10
ToolingFlangedHeight	0.20
Length	10.00
Thickness	0.10

	Quantity	Periphery	Area	FinishedVolume	PartWidth	PartHeight	RawVolume
spar	1	22.00	10.00	2.19	0.80	0.10	2.20

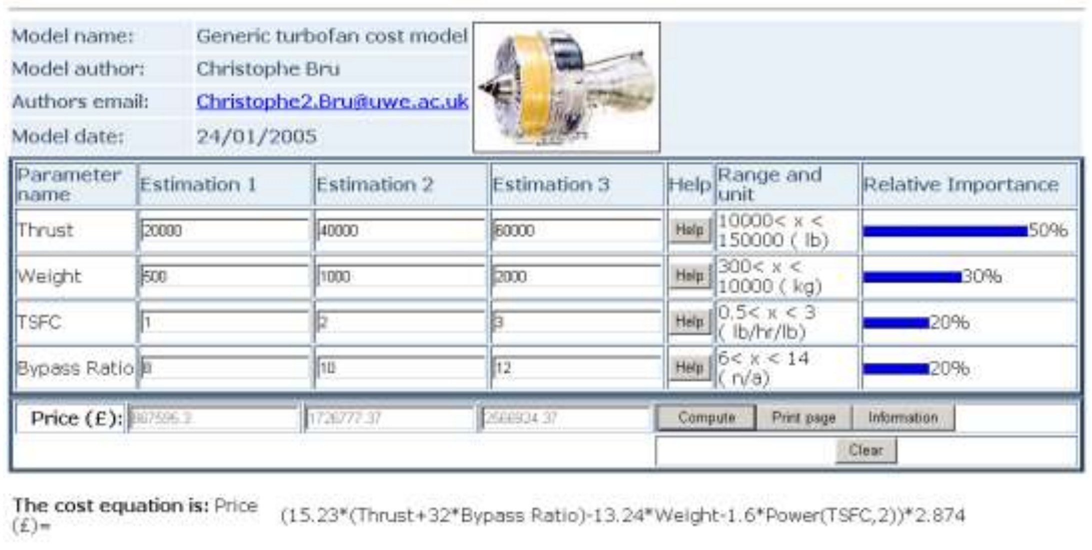
Figure 52. Translation from decision tree to Cost Estimator

Translations were also made into Meta-Programming languages, but further research is needed to automatically create software from these Meta-Programs; this is explained in 8.2.4.

6.6 Wider Engineering Translation

This approach to modelling, translation, and visualisation can be applied to other types of engineering problem. The example below is not of a wing component, as it was created for Rolls-Royce who manufacture aircraft engines. Creation of web applications for Rolls-Royce (part of the DATUM project) led to the view within the DATUM project and for this thesis that web applications could be a popular method of delivery for information and models. The example in Figure 53 is of combined work of the author and Bru for the DATUM project, the DATUM project is described by Scanlan et al. (2006). The work to enable the example model shown in Figure 53 tested use of web standards to provide a multi-user, freely accessible method for conveying information and models. The model is included in the thesis as an example of enabling free editing by all users of a simple model at the very earliest stages, when more complex information such as production processes and materials are not yet known. Therefore this is a good stage to involve anyone who may have an input into the requirements and potential costs of the components. Figure 53 shows an implementation of this simple ontology based parametric model.

[36] Koonce, D., Judd, R., Keyser, T., Bailey, M. A., 2001. A Cost Estimation Tool Integrated into FIPER, in: IFIP Conference Proceedings; Fourth International Conference on the Design of Information Infrastructure, vol 191 2001 pp. 396-402.



Model name:	Generic turbofan cost model
Model author:	Christophe Bru
Authors email:	Christophe2.Bru@uwe.ac.uk
Model date:	24/01/2005

Parameter name	Estimation 1	Estimation 2	Estimation 3	Help	Range and unit	Relative Importance
Thrust	20000	40000	60000	Help	10000< x < 150000 (lb)	50%
Weight	500	1000	2000	Help	300< x < 10000 (kg)	30%
TSFC	1	2	3	Help	0,5< x < 3 (lb/hr/lb)	20%
Bypass Ratio	8	10	12	Help	6< x < 14 (n/a)	20%
Price (£):	867596.3	1726777.37	2566934.37			

The cost equation is: Price (£)= (15.23*(Thrust+32*Bypass Ratio)-13.24*Weight-1.6*Power(TSFC,2))*2.874

Figure 53. Parametric Cost Estimation

This parametric modelling tool was provided to Rolls-Royce with a simple XML based ontology to enable immediate implementation.

6.7 End-user Programming/Modelling Evaluation

Regular contact with the British Computer Society SocioTech group helped inform the research and assisted with development of a way to evaluate use of the prototypes. User feedback was obtained from regular meetings with Airbus and Rolls-Royce for their projects. These were requirements of the projects and were useful for developing ideas, testing them, and for developing and testing the implementation and prototypes (many of which were installed at these organisations). The planning and organisation of these meetings was systematic and matched with project goals. Also as it became more possible to create quite advanced demonstrations online (due to Semantic Web and Web 2.0 advances) this made it possible to get valuable feedback from the online examples and surveys of these.

6.7.1 End-user Survey

A web based survey was conducted early in the research (Figure 54) to assess likelihood of success for providing end-user programming, and to establish the viability for end-user programming necessary for widening of participation. From 234 participants 53% answered that this was possible and 47% that it was not. This indicates that research further into this problem is necessary as just over half of respondents indicated it was possible but this was not a convincing majority. The smallness of the majority of those who believed it was possible over those that believed it was not, was the first indication of a need to focus on a more achievable intermediate aim of applying this to tree/graph based modelling.

The participants can be assumed to be computer literate as they found and filled in the survey, some may be experienced programmers, while others could be the kind of computer literate end-users who might benefit from wider programming participation.

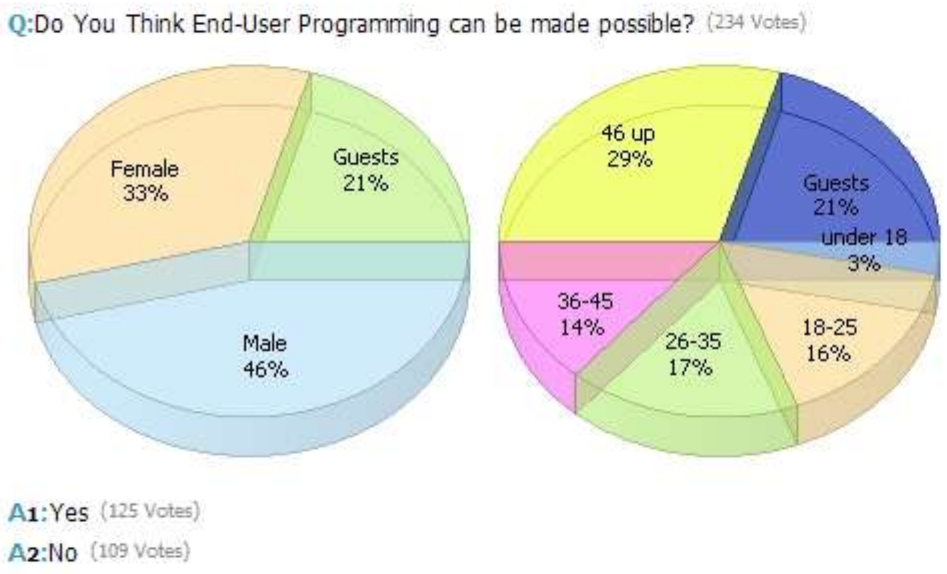


Figure 54. End-user programming survey

Survey available at http://www.toluna.com/polls/37921/Do_You_Think_End-User_Programming_can_be_made_possible?

The above screenshot (Figure 54) from the web survey shows the results and breakdown of participants. Guest are those who were not logged on and so the web survey could not identify their age or gender. From this, it can be seen that responses were across the age range and both genders with male responses probably outnumbering female.

6.7.2 Usability Evaluation

A usability evaluation of 61 people was also conducted, that was more specific and focused on the prototype implementations. This supplemented the feedback obtained from Airbus and Rolls-Royce. This evaluation was conducted in order to determine what kind of diagrammatic interface might be most appropriate. Extensive consultation with engineers at Airbus and Rolls-Royce helped establish the possible interfaces that needed evaluating, as these interfaces were useful to and understood by these engineers. These possible interfaces were taxonomy, CAD style diagrammatic, flow chart (or activity diagram), UML style (entity relationship). Which was preferred depended partly on the skills and type of work of the engineers, and others surveyed, and on the type of problem to be modelled/programmed.

Which would you consider yourself to be?

answered question 87
skipped question 4

An End-User (anyone else, familiar with computers but not in categories below) - 27.6% - 24
An Experienced Developer - 28.7% - 25
A Researcher - 33.3% - 29
Other (please specify) - 10.3% - 9

What type of visualisation is best for enabling end-user programming?

answered question 72
skipped question 19

Taxonomy (tree view) - 29.2% - 21
Diagrammatic - 38.9% - 28
Flow Chart - 34.7% - 25
Unified Modeling Language (UML) - 13.9% - 10
Other (please specify) - 12.5% - 9

Which example Program have you used?

answered question 75
skipped question 16

Taxonomy viewer - http://www.cems.uwe.ac.uk/~phale/ELearning/ELearningDemonstration1.htm - 18.7% - 14
Diagrammatic Viewer - http://www.cems.uwe.ac.uk/~phale/ELearning/ELearningDemonstration2.htm - 18.7% - 14
Neither – 40.0% - 30
Both - 22.7% - 17

Which example interface do you prefer?

answered question 70
skipped question 21

Taxonomy (tree) viewer - http://www.cems.uwe.ac.uk/~phale/ELearning/ELearningDemonstration1.htm - 28.6% - 20
Diagrammatic Viewer - http://www.cems.uwe.ac.uk/~phale/ELearning/ELearningDemonstration2.htm - 18.6% - 13
Neither – 10.0% - 7
Both - 12.9% - 9
No opinion - 30.0% - 21

More detailed results are available at - http://www.surveymonkey.com/ - login details petervincenthale – toffee, or summarised at - http://sites.google.com/site/userdrivenmodellingprogramming/Home/usability-survey-results.

The evaluation results indicate a need also to enable users to switch easily between different visualisations, as they require. Respondents have given extensive feedback, with detail as to the reasons for their responses, other questions that might have been asked, and well informed opinions and advice about this research. This provided extra qualitative results and feedback as well as the quantitative results. Both the quantitative and qualitative results were used to guide the research.

6.8 Analysis and Reflection

The prototypes in Chapter 6 are more effective in addressing the problems of maintenance, extensibility, ease of use, and sharing of information, than those in Chapter 5, and so represent progress. They do satisfy the aims of improving software development through the interaction with diagrams without requiring people to learn computer languages. However, a further aim would then be to research how this could assist with collaboration in problem solving. To achieve that a new project would be required, and is proposed as future work, to amalgamate and unify the different technologies used, and develop them with others who have created high level software, Web 2.0, and Semantic Web tools. This would enable creation of a single user interface for the ontology, modelling, and web parts of the approach to simplify use of the system, and the maintenance, re-use, and extensibility of it, and enable sharing of information amongst a wider user base.

Chapter 7 - Discussion

Chapter 7 discusses what has been achieved and what remains to be done. The discussion also examines ways to further unify this research with others, amalgamate, unify the system to created, and widen the user base by applying the research to fields and problems not yet addressed. This chapter examines what has been learned from and gained from the research and implementation and so allows moving on to examine the main uses to which the research can be applied.

Figure 55 shows the research direction this thesis moved in over time.

- End-user programming for modelling. Research began with examining programming needs of users (mainly engineers at Airbus and later Rolls-Royce aerospace companies).
- Modelling - Next, models were created to cater for the engineers needs for prediction and decision support; so the concentration was mainly on modelling.
- Semantic Web for modelling. Once the models became more complex and numerous, the collaboration necessary required a more systematic infrastructure, so Semantic Web and ontology research assisted with this.
- Semantic Web for end-user programming. Once an infrastructure was prototyped it was possible to research how this could enable construction of programs based on the Semantic Web/ontology infrastructure.
- Programs (also program/model creation systems) created using the Semantic Web infrastructure and visual diagrammatic programming were tested for modelling capability.

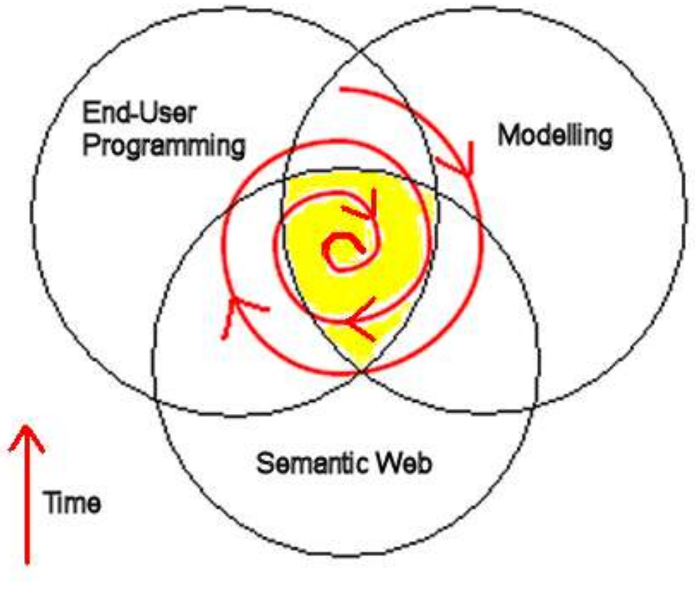


Figure 55. Research Direction Spiral

By the end of this research it was perceived that this particular combination as indicated by the midpoint of the shaded/yellow section was currently under researched by the wider community. For the literature review in chapter 2 it was found that researchers were concentrating on at most two out of the three subjects shown in Figure 55. Examination of literature throughout the PhD research also indicated the need to concentrate on investigation of End-User Programming, Modelling, and Semantic Web combined. This is why the research spiralled in on this combination in that way. For example Gruber, and Horrocks, specialised in Ontologies. Berners-Lee, and McGuiness combined Semantic Web and Ontology research. Miller and Baramidze combined research into Modelling and Semantic Web. Johnson, Huhns, Paternò, Crapo, Lieberman, and Resnick examined interaction and End-User Programming for Modelling. Only by examination of the research of all these people, and using this in combination, learning from their developments in each research area and combination, could all three research areas be combined and developed.

After researching the areas shown in Figure 55 it was necessary to implement this combination in a prototype using current technology. Examination of this research and the implementation led to the recommendation of an approach for the kind of tree based modelling/programming examined in this thesis to be based on these options :-

Option 1 - To put all the data in Semantic Web languages e.g. XML (eXtensible Markup Language), SVG (Scalable Vector Graphics), RDF/XML (Resource Description Framework), OWL (Web Ontology Language), and then display them using a programming language such as Flash, or Java (applets).

Option 2 - To use the above languages as meta languages for actual programming, including the display interface.

Option 1 concentrates the main effort on the Semantic Web/Ontology representation, and the modelling of this, and then adds on a user interface to match this and enable End-User Programming. This option is easier to implement than option 2, but only involves the System Creator, (introduced in section 1.5.1 and illustrated in Table 1 and Table 3). Option 2 would concentrate at all stages of development on all three research areas, potentially creating a more integrated solution with greater user involvement, extending involvement of development of the modelling system to Model Builders. This would aid knowledge transfer and collaboration within the early stages of the model system development as well as after the modelling system is created and when it is being used.

It was also possible to use aspects of both approaches, such as to program mainly in Semantic Web languages, and then add some extra interactive capabilities using JavaScript iteratively. It is becoming increasingly practical to program completely in the Semantic Web languages (option 2). These languages enable declarative programming, and a translation is performed either using languages such as JavaScript or Java, or translating into JavaScript or Java. This is different from the AJAX (Asynchronous JavaScript and XML) approach, which is more like option 1. At present both approaches are used in this thesis, because though option 2 is potentially more effective there are still practical technological problems to overcome. These problems result from the need to interpret/compile Semantic Web representations into program code. This should become more practical as research and implementations progress in this direction.

The advantage of this (option 2) form of declarative programming is that it is possible to use a language that is at a much higher level of abstraction, closer to the way people think. It was possible to create these programs by editing them in Protégé (ontology editor) and using a translator to convert them to whatever code was needed. This made it possible to perform visual programming in a meta language (OWL) (option 2), without needing to be concerned about how it was implemented. The possibilities for this are that it becomes sufficiently intuitive, so that people can create their own software for a wide variety of tasks, in a point and click way and using similar tools to web editors. This would enable anyone who is computer literate to program the computer themselves to do their tasks, and if this is of interest to others, they could release their solution over the web. Technologies such as XForms, XQuery, and SPARQL make it possible to provide this sort of collaborative interactivity.

7.1 Enabling Visual Semantic Web programming

This implementation demonstrated 2 main types of change that need to be enabled to provide for the editing necessary for visual modelling/programming. These are 'User Generated', and 'Model Generated'.

User Generated

Figure 56 shows a user initiating a change, which is to delete a node from the bottom left and attach a new node to a branch in the top tight. The tree is translated to structured text, and this is further translated to code.

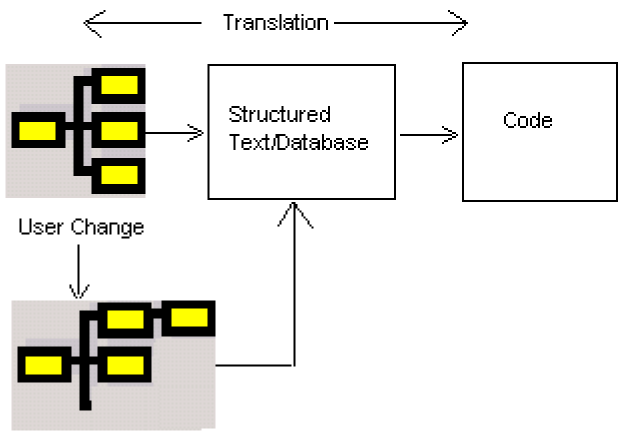


Figure 56. User Generated Change

For the second user generated change shown in Figure 57 an object represented by a tree is visualised as a diagram. The user can amend either the diagram or the tree; in either case the change is filtered to the alternative representation and translated to the structured text and code.

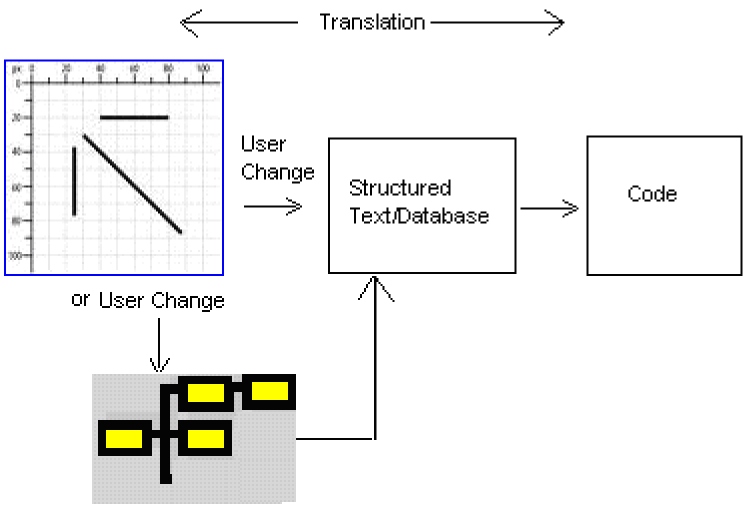


Figure 57. User Generated Change, Alternative Interfaces

Such an interface has been provided but editing the CAD style diagram outside a CAD system is quite an advanced undertaking, so is to be future research (section 7.3).

Model Generated

A model generated change is initiated by the model itself, which changes the code and the structured text in response to a calculation (that may have been requested by the user). The model passes a translated result tree (or graph or web) to the user interface to let the user know that the recalculations have been finished, and give the user the results using a suitable visualisation. This is shown in Figure 58 :-

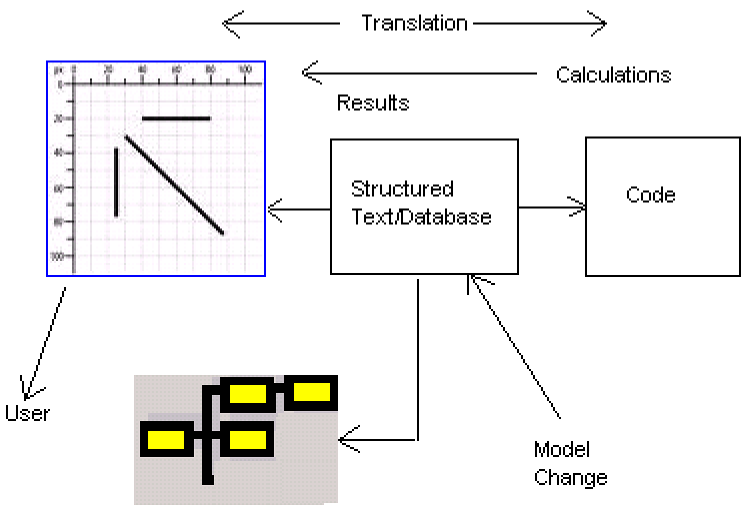


Figure 58. Model Generated Change

7.2 Widening user programming participation: Generic Modelling

Figure 59 shows the main areas of application for User Driven Modelling. This research has concentrated mainly on the central area of Decision Support, but it could be widened to Knowledge Management, and to Simulation.

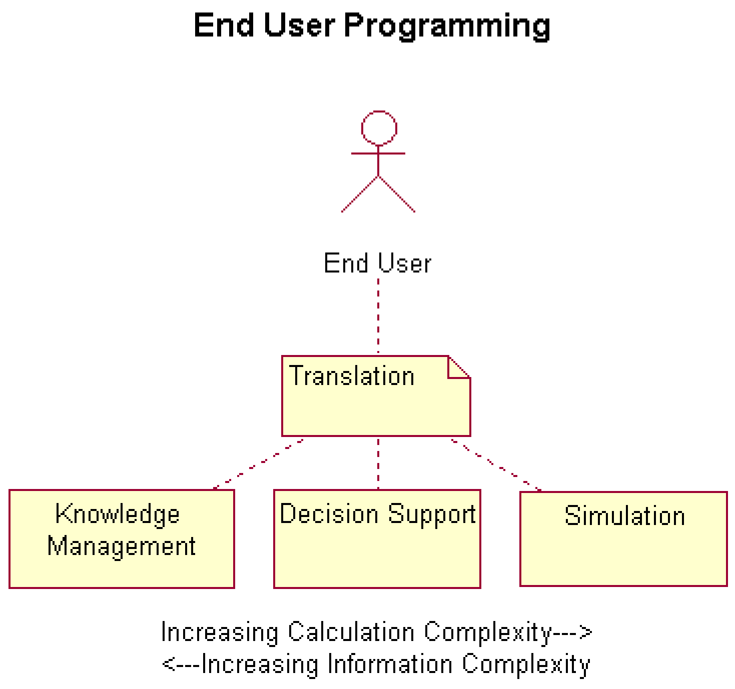


Figure 59. Generic Modelling

For generic modelling it is necessary to deal with complexity. This complexity is as a result of the need for representation of complex knowledge in a way that is correct and understandable to a wide range of people, and then of complex calculations that may draw on many parts of the ontology representation. For the types of modelling shown in Figure 59 pure Knowledge Management may not require any calculation, however this representation can be complex as the aim of the Knowledge Management and how wide ranging this needs to be is not known, as there is not the target of a Decision Support model to be constructed from the knowledge. This thesis concentrated on using Knowledge Management for Decision Support, which required complex and recursive calculations. For simulation the calculations may be required in real time. Therefore complexity of calculation increases in the direction from Knowledge Management to Decision Support to Simulation, and complexity of information representation increases in the other direction from Simulation to Decision Support to Knowledge Management.

Whatever is being modelled, the main issue is enabling management of complexity through appropriate structuring, visualisation, and interaction. Naeve (2005) examines Semantic Isolation, Semantic Coexistence, and Semantic Collaboration. In section 3.6 Ontologies and Semantic Web and their role in Modelling' of this thesis the implications of the need for appropriate and systematic use and combination of tools and techniques in order to enable Semantic Collaboration were examined. When the appropriate structure is created with these tools and techniques, modelling of things other than cost for engineering components uses the same categories and just requires different equations, so this methodology broadens to engineering modelling of properties other than cost e.g. design and/or manufacture (process plan). All the information necessary for creating a CAD (Computer Aided Design) type representation can be made available in the various categories and is indeed necessary for the production of the most accurate cost estimates. This made it possible to translate between a tree/graph-based representation and a CAD style representation. A further advantage of this methodology is that it allows estimates to be made at top level in the tree in early stage design, but this early estimate can be developed at a later stage when more is known of the design and production processes. This makes it possible to add further detail to improve the accuracy of cost modelling and/or other engineering modelling. Representing things other than engineering components merely requires different categories, so this methodology could be re-used for other modelling problems such as scientific modelling and economic modelling. This broadens to any problem that can be represented by linked equations, which can be visualised, especially if combined with Natural Language Programming (NLP) research, e.g. (Mihalcea, 2006). Visualisation and management of the information can aid in constructionist understanding of problems by modelling them (Papert, 1991) (Resnick, 1996). This methodology and constructionist approach ensures greater user involvement in model production and management.

User involvement is important in development of software but a domain expert does not necessarily possess expertise in software development, and a software developer cannot have expertise in every domain to which software might apply. So it is important to make it possible for software to be created, in ways that are as close as possible to those which the domain expert normally uses. This research examined the need for and provision of the User Driven Modelling/Programming methodology to enable computer literate engineers to model/program. The proportion of domain experts in a particular domain (aerospace engineering for example) who can develop their own programs and have access to such facilities is fairly low, but the proportion that are computer literate in the everyday use of computers is much higher (Scaffidi et al., 2006). Harnessing of computer literacy to allow domain experts to develop and share models increases productivity for software development and reduces the proportion of misunderstandings between domain experts and developers. Domain experts can then explore a problem they are trying to solve and produce code to solve it. The role of software developers would then become more that of a system creator, mentor and enabler rather than someone who has to translate all the ideas of domain experts into code themselves. Other software developers may work at providing better translation software for the domain experts (e.g. engineers). Creation of a systematic environment for model building can be assisted by advancing research in current tools and techniques, and standardising the representation and navigation of information. This enables complex problems to be represented by networks of models and modellers that share information.

As end-users are beginning to creating simulations and other software it is important to address this need and attempt to establish a dependable way for them to do this. To achieve the above aims it was necessary to research the interface between Meta-Programming, Modelling and Simulation, and Semantic Web Model Creation, shaded in Figure 60. This could allow end-users to develop their own Semantic Web based simulation and modelling tools using a graphical visual interface.

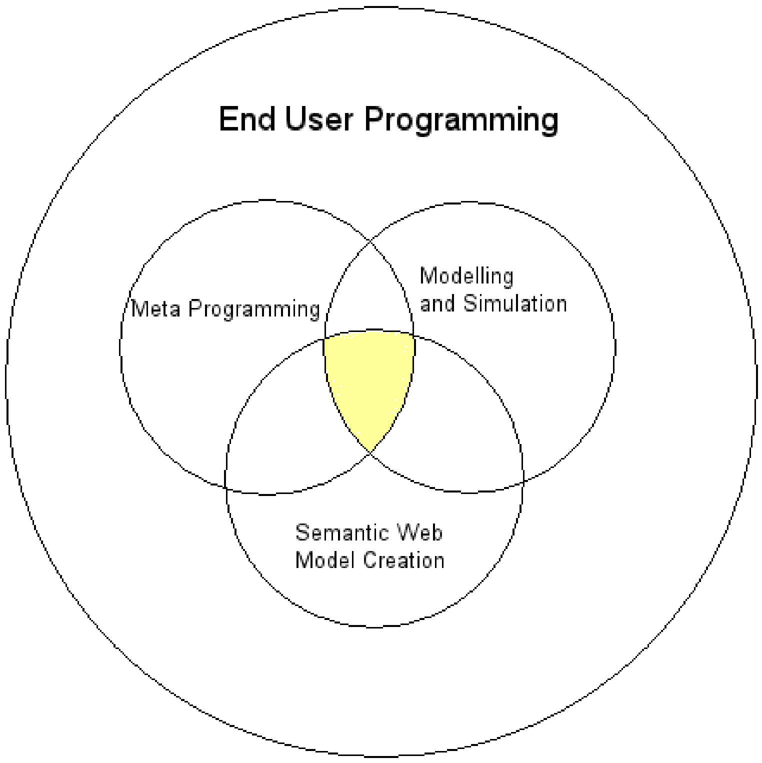


Figure 60. End-User Programming, Modelling and Meta-Programming with Semantic Web

To achieve the aim of widening user programming to involvement in more generic programming/modelling, it would be necessary to amalgamate and combine these research areas within an overall umbrella system of enabling End User Programming. Creation of a single user interface could enable generic programming, including for modelling/programming that is not tree based, then enabling a wider user base. This user interface can be created in a simple way even through a spreadsheet, or database data entry forms. Chelsom et al. (2011) illustrate this approach.

7.3 Widening Participation: Online Editable Systems

During this research some work was contributed to the University of West of England's Students Online[37] system, this investigated enabling staff to edit their profile using an XForms based system. This contrasts with the approach in the Rolls-Royce CAPPe system of provision of the information by a small group of experts. The extent to which information is defined top down or bottom up (discussed in sections 3.6.2, and 4.1) depends on the type of information that is being entered. More technical information such as engineering information is more likely to require the input of specialists in that area of work, and with that particular expertise. Also the structure of the organisation is important, a more hierarchical organisation is likely to require a more top down approach, and to use an information architecture that supports this.

Future work should involve construction of an online user interface for model creation software that makes use of XForms and/or other XML and RDF programming standards. Examples of online ontology creation were developed through 5.4 and 5.5 and shown in 6.4.4, and survey test results were discussed in 6.7. These examples used a combination of Semantic Web and Web 2.0 techniques. The research for the Students Online project showed it was possible to represent information and actions via declarative XML code and search and process it using declarative queries (XQuery). This provides an infrastructure for automation of editing using visual structures such as taxonomy based editors. This could also enable ontology tagging by users as discussed in section 3.6.

7.4 User Driven Modelling Solutions

Figure 61 illustrates the solution needed for User Driven Programming, established by the research, now the research implementation for this solution has been prototyped. This combines the thesis research, and that from the projects of others in end-user programming, the Semantic Web, and modelling referenced in this thesis.

7.4.1 Method

- The Translator (Figure 62) automatically creates the Schema and Stylesheets from the Ontology, and needs to update this only when the Ontology is changed.

- The Editor and Viewing and Tagging Interface can be created automatically from the Schema, Information, and Stylesheets. The Stylesheets act as a visualiser of the translated information.

[37]Students online (2011) [online]. Available from: http://www.cems.uwe.ac.uk/exist/index.xql [Accessed 22 June 2011].

• In Response to the Actions of Users that trigger a request for information, the Translator :-

- Makes the request to the Ontology.
- Receives information back.
- Forwards this to the information holder.
- The Editor and Viewing and Tagging interfaces are updated automatically.

• The purpose of the Viewing and Tagging is to visualise the information and allow tagging of items to explain the meaning, and add nodes to give further understanding of the item. This additional information could be fed back to the ontology to assist improving it.

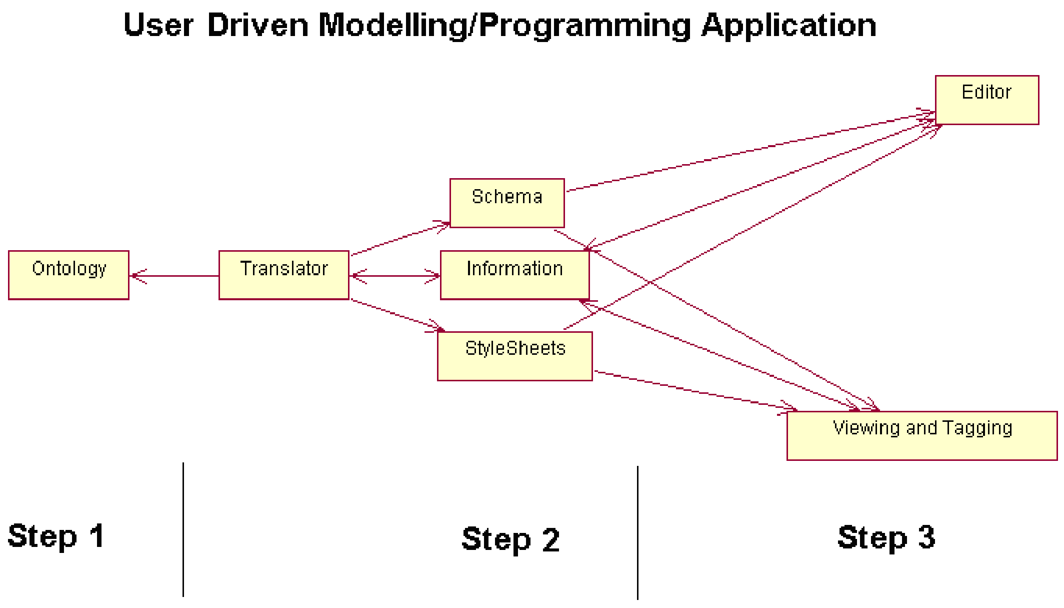


Figure 61. Solution for User Driven Programming

For Figure 61 above Step 1 is the ontology used for creating systems/models. Step 2 involves translating the system/models and so making them available for use and development. Step 3 is the editor and viewing and tagging interface, for visualisation and interaction with modellers and end-users

This solution involves an ontology and a translator that communicates with the user through Semantic Web languages that are visualised via the web, and the translator communicates with the ontology. This translator therefore provides two way interaction between the editor and the ontology. So far research has been undertaken and demonstrated for the thesis with Protégé and Vanguard System for this translation purpose and linked to an XForms user interface 5.5.2 and 5.5.3, as well as illustrated using web-based Semantic Web/Web 2.0 visualisations 6.4.4. An important point is that for these technologies to be extended into use for end-user programming ways must be found to visualise the information structure rather than expect users to understand this in text form. So a translator was created to convert diagrammatic visualisations to text. This enabled programming with Semantic Web languages rather than just using them for information representation. This makes the translation easier and more reliable, and improves maintainability of the whole system. The use of Semantic Web languages as programming languages assists greatly with interoperability as these languages are standardised for use in a wide range of computer systems (2.3.2 - Semantic Web and Ontologies). Although other researchers have prototyped Semantic Web language based search tools, this has not yet been combined into a comprehensive application that is usable for end-user programming of a large range of modelling problems. A flexible interface built with Semantic Web languages provides an interactive programming environment for computer literate non-programmers to manipulate information and construct their own models. Further work could enable user generated user interfaces to be customised as required by particular users; this could be achieved by a kind of Semantic Template based programming.

7.5 Business Benefits

Sternemann and Zelm (1999) explain that it has become necessary to research collaborative modelling and visualisation tools, because of the business trend towards global markets and decentralised organisation structures; Green et al. (2007) also explain this. Section 3.4 Ontology Based Modelling Solutions' examined the approach of Cheung et al. (2007) to this problem. This thesis demonstrated a system that could be used to solve this problem by means of accessible, interoperable collaborative software to enable visual modelling/programming. This can then be integrated with software already used in industry, as was achieved for the Rolls-Royce DATUM project described in sections 3.6 and 4.8. This enabled visualisation and collaborative modelling using information formally only available in text based reports.

With visual diagrammatic modelling it was possible to include one model within another as a software component, and demarcate responsibility for building, maintenance, and updating of each model. This was difficult using spreadsheets, and possible with non-visual programming though the link between individual responsibilities and code produced was not as clearly identified, because non-programmers cannot participate in code production, and this was not so clearly visualised. As an example of visualisation and interaction for cost modelling of an aircraft wingbox, different experts might build software models for wing spars, wing skins etc, and another expert might be responsible for the overall wing cost model. The wing spar and skins model can then be inserted into the overall wingbox cost model. Such a system makes generic modelling more achievable.

The techniques demonstrated in this thesis can aid progress towards accessing of data held using Semantic Web standards, and also other information that might be locked into particular systems such as databases, spreadsheets and enterprise systems. The translation and de-abstraction approach assists with enabling high level diagrammatic visualisations to be used and translated to computer queries. Programming using Semantic Web technologies can :-

- Assist with translating non-Semantic Web information into Semantic Web information.

- Assist in production of Semantic Web information by end-users.
- Assist end-users to query non-Semantic Web information.

7.5.1 Visualising Reports

Many organisations produce text based reports with their IT systems. But text based reports do not always show information well enough for good decision making. Automated conversion of these reports into Semantic Web languages and then visualising them could assist greatly with this. So a translation process is required and can be implemented as part of an overall User Driven Modelling/Programming approach. Once reports are converted to a standardised representation, hierarchical information can be represented as clickable trees/graphs and numerical representation as charts. This makes it possible to customise outputs from existing IT systems and so allows an improvement in readability of information without major changes to the way it’s produced. This could provide a large gain at little cost.

7.6 User Driven Modelling - Intermediate Benefits

Although User Driven Modelling/Programming is a difficult problem and only partially solved, there are numerous intermediate benefits from the search for this approach. These include better modelling and visualisation of problems, improved interaction with end-users, Semantic Web modelling search and visualisation methods, collaboration to improve modelling, and ways to agree ontology and Semantic Web representations. It was necessary to provide such intermediate benefits as the industrial collaborators had shorter term goals and so required deliverables.

The techniques used helped with progress towards improved interoperability that can aid in all the above areas. These uses and improved interoperability to support them needed to be developed together in an iterative way.

Experienced programmers/software engineers may have many of the problems of end-user programmers whenever they need to use a language/system they are unfamiliar with, or when the language/system they use is updated to a new version. So this means the techniques and approach developed can aid experienced software developers in such circumstances, as well as end-user programmers.

Ch 8 - Conclusion and Further Research

8.1 Summary and Evaluation

The thesis covered the following areas :-

- Enabling people to create software visually.
- Creating design abstractions familiar to domain experts e.g. diagrams for engineers.
- Ensuring interoperability using open standards.
- Automating the user to computer translation process.

This chapter explains how the alternative approach of User Driven Modelling/Programming used for this thesis to develop models and modelling capabilities compared to that of spreadsheet development, used within an Airbus project (Chapter 5). The alternative approach was outlined in this thesis, of using open standards ontologies/taxonomies and a web interface for developing decision support models for design and costing.

The approach of developing decision support models for design and costing using a spreadsheet was compared to the alternative approach of using open standards taxonomies and a web interface for this purpose. The conclusion is that although use of spreadsheets allows for the creation of models relatively quickly, they are beset by problems. These relate to Maintenance, Extensibility, Ease of Use, and Sharing of Information. The spreadsheet example and the explanation in chapter 5 section 5.1.2 represents problems currently experienced throughout software and computer use.

The research involved enabling computer literate people to create software using a combination of modelling via use of formulae (equations) and visualisation of the way these formulae interact. It was possible to construct modelling software without requiring code, but there is a need to use software technologies that are being developed now, unify and simplify the system. Crapo et al. (2002) explained the need for this approach, and this was applied in this thesis as a User Driven Modelling/Programming approach. This approach was applied to engineering modelling problems, and the test implementation proved it was usable for large models. Some work has already been completed on applying this to other problems. The approach involved a 3 step translation process, and this eased problems of maintenance, extensibility, ease of use, and sharing of information. Each step produces a diagrammatic representation for humans, and computer code.

The stepped translation approach designed and implemented in this thesis enabled structured modelling, and visualisation using interactive technologies. The steps for this were as below :-

- Step 1 - Ontology

- Step 2 - Modelling Tool and System Translator
- Step 3 - Interactive Visualisation

Table 3 shows the roles and skills of people who would make use of this translation.

Table 3. Roles, Skills, and Translation

Step	Person Role	Skills	Tool Type
Step 1	System Creator	Programmer	Ontology
Step 2	Model Builder	End-User Programmer	Modelling Tool and System Translator
Step 3	Model User	End-User	Interactive Visualisation

This stepped translation solved problems of the spreadsheet approach as indicated in Table 4 below, and then in more detail in the following sections :-

Table 4. Stepped Translation and Modelling

Improvement	Achieved By
Maintenance	Structuring and Translation
Extensibility	Structuring and Visualisation
Ease of Use	Visualisation, Interaction, and Translation
Sharing of Information	Shared Ontology and Interoperability

The 3 step translation process created ensures translation of domain level modelling into open standard representation and software and vice versa. The approach takes a representation of a problem and breaks this down into subsets to repeatedly simplify the problem until a result subset is produced that represents a reasonable solution to the user. This translation process was described in section 3.3.2. Logic languages might enable getting to this subset quicker but would involve much more research and work to design and implement a solution, so these are reserved for future work. This does lead to a problem in that some other representations may be at a more structured level in the stack of possible ontologies (shown in Figure 3 section 2.3.2), and others higher, so this requires translation between levels in order to integrate with other ontology systems.

The alternative approach for this thesis of User Driven Modelling/Programming involved the development of a system, where a model builder, via visual editing of library taxonomies can undertake maintenance and extension of information. Dealing with this proof of concept has indicated that it is easier to maintain, search, and share information using this approach than it was using spreadsheets. This approach could also enable more of the maintenance task to be left to end-users, who then could also customise the system. Creating the infrastructure has taken more time than it did for the spreadsheet system, but having done so it is quicker and easier to create further models. This indicates that the extra research and development time taken, though far exceeding what would have been required for a spreadsheet modelling project, is well worth it in the long term. Also the use of a centralised information source makes these models more reliable than the standalone spreadsheet, standalone decision support models created individually are prone to out of date information. In addition, since a well constructed ontology implies that a piece of information can only belong to a unique location, the problems arising from duplicate pieces of information can be eliminated, especially if top level ontologies, standards, and namespaces are used. It is also easier to create models once the infrastructure is in place; this can enable users to develop models. The ability to visualise, search, and share information using structured languages and web applications is an advantage for creation of dynamic structured views and decision support models over the web, to aid collaboration.

This research was a test case for an approach, of collaborative end-user programming by domain experts. The end-user programmers could use a visual interface where the visualisation of the software exactly matches the structure of the software itself, making translation between user and computer, and vice versa, much more practical. For this reason, highly structured visualisations were preferred over web spreadsheets. Semantic Web languages are ideal for representing graphs and trees in an open standard way. The spatial, and tree/graph visualisations used both have the same underlying semantics, and therefore can both be translated to computer languages. In fact it would be better in the long run to use the Semantic Web languages as standardised programming languages for such problems as this would avoid the need to further translate into other programming languages, and systems. The advantage to this is that of using Semantic Web languages for representation of information, meta-programming, and translation to a visual display for users. The use of Semantic Web languages as a connectivity environment for connecting information, and for connecting users to the information held in Semantic Web data sources could enable an environment that could be made easier to use, install and maintain.

More generally a new approach is required to software creation. This approach should involve developers creating software systems that enable users to perform high level programming, and model the problem for which they are the experts, and involve collaboration to share results. This is an alternative to the provision by developers of modelling solutions that try to provide an out of the box solution that just needs ‘tweaking’. Such an out of the box system is not practical for complex models/modelling considering both increases in complexity of manufactured products, and of software systems themselves. Feedback from publishing the research examples behind this thesis and working with industrial partners indicates that people like to work on their own solutions, providing they are computer literate and confident they have domain knowledge that the developers do not possess. This is true for software development in general, not just in the domain of engineering. Research cited in this thesis from others involved in end-user programming confirms this.

8.1.1 Maintenance

The use of a centralised information source makes these models more reliable than the standalone spreadsheet. This centralised structure was easier to manage than updating multiple instances of the spreadsheets used by different people and ensuring they all contain the same information. So the first task was to build a system for collaborative model building. As a piece of information can then only belong to a unique location, the problems arising from duplicate pieces of information are eliminated. In a wider implementation use of namespaces and integration with other ontologies could ease the maintenance problem further. The models have only the functionality that is added by the model builder so there are not other side effects to keep track of, as there are with generic functionality within spreadsheets. Enabling people to create software visually makes it easier for model builders and model users to keep track of any information they are responsible for. Translation from the ontology to models and visualisations ensures one change will affect all stages, so this makes maintenance easier.

8.1.2 Extensibility

Creating the infrastructure for the collaborative model building system took more time than it did for the spreadsheet system, but having done so it is quicker and easier to create further models, and this makes it more practical to integrate with other Semantic Web and modelling systems. This benefit results from the facilities provided for model builders and end-users to customise the software in any one of the three step translation process. This means progress has been made in making it possible for non-programmers to build models. It also indicates that the extra research and development time taken was worth it in the long term, most of this time involved productive research and this can be used in future projects. The use of open standards in this thesis for information and models ensures there should be a development path, to help cope with changes there may be in external software systems. This use of open standards also ensures that the system can link with most environments used by others. The translation and visualisation approach ensures that new models can be added using the existing ontology, and that design changes in the ontology and translation can enable modelling of different problems. So if there is a new problem to be modelled there are two ways to achieve this, add new models to the ontology, or change the ontology.

8.1.3 Ease of Use

Many people now are familiar with web pages and at least the basics of how to navigate them, and by creating such an environment, and standardising the navigation to those ways commonly used over the web, it is possible to ease usability. Comparison of the approach of spreadsheet modelling as tested with the ACCS spreadsheet model (section 5.1) demonstrated the advantage of including only the functionality needed. The use of spreadsheets which had generic functionality that was not required for models led to users' confusion. So an advantage of creating models using the alternative approach of developing a system for this research outside of spreadsheets (section 5.5, and chapter 6) is that the models contain only the functionality that is added by the system creator and model builders. New Web

2.0 interaction technologies have allowed production of a rich user interface for web programs in a similar way to single computer applications. This means information held in an ontology and translated through modelling tools can be made available as interactive applications for many users. Translation allowed the same user interface to be provided in multiple tools and computer languages. Also this research showed that it was possible to provide user interfaces and visualisation differently as appropriate according to the type of user, the situation, and/or the kind of information to be shown.

8.1.4 Sharing of Information

The use of open standards languages for representing information makes it easier to represent information in a way that makes it accessible both to people and software. Use of namespaces and integration with standardised upper level ontologies, and standardised semantic languages such as PSL (Process Specification Language) (covered in section 3.5.1), could ease sharing of information. Ontology based modelling tools use these open standards and so ensure dependable translation, interoperability, and sharing of information. Web browsers make it possible to share information with many users at once, and so this enables collaboration. Structuring of the information using standardised languages makes it easier to search and visualise the information. This ensuring of interoperability is important for long term use of the overall modelling system.

8.2 Conclusion

The research question answered in this thesis is - ‘To what extent was it possible to improve user-driven software development through interaction with diagrams and without requiring users to learn particular computer languages?’The solution advocated and tested for this is based on representation of software as a tree and translation from a source tree to a result tree, and visualisation of both. The result tree shows a structured representation of the model with a full visualisation of all parts of the model that led to the result.

It is concluded that this direct representation of the structure of the code based on an ontology that provides a tree-based representation of the software enables an understanding of the program as an ontology and model that is then visualised. This results in a shared understanding by all users. The main limitation is that the research was constrained to problems/models that lend themselves to representation as tree structures. The translation used does enable other types of visualisation, such as methods of CAD style, provided there is an underlying tree structure to the information. The usability evaluation in Chapter 6 tested the extent to which visualisation and navigation of the structure aided the understanding of the structure.

The wider significance of this solution is that it applies to tree based structures, and tree based structures are generic and common to many programs and models. The solution is based on linked nodes that are visualised and linked by equations in a model. The advantage is that it is simpler then to develop a structure based on standardised representation of these equations in a generic way. This enables model builders more easily to develop a model based on mathematical representations with high level syntax and semantics, thereby avoiding having to learn and use one or more ‘lower level’ programming languages. Further, such a representation is independent of programming languages so is stable, whilst programming languages wax and wane in terms of popularity and extent of use. An example of a use of this tree structure in order to chain equations through the tree to the result figure at the root is process and product modelling that can be used for decision support and costing. Widening of applicability is possible because when calculation is not necessary and instead a hierarchy is to be represented for knowledge management, the tree can be created via a syntax and semantics for linking nodes, which is simpler than that for linking nodes with equations. An example of wider uses of the solution without the extra facility of chained equation calculation is phylogenetic trees for representation of biological evolutionary histories and structures. Additionally, knowledge management within industry, and business process modelling are other fields for which this method could be used.

The structure and visualisation/representation of this structure diagrammatically ensures visibility for editing, and thus brings with it advantages for ease of editing of this structure and thus improvements for maintenance, extensibility, ease of use, and sharing of information.

The answer then to the question - ‘To what extent was it possible to improve user-driven software development through interaction with diagrams and without requiring users to learn particular computer languages?’ is partially, at least for this thesis. The solution used in this thesis provides for effective diagrammatic modelling/programming for tree-based problems. This solution has wide applicability, but other solutions need to be found for diagrammatic representation of the many problems that do not fit in with this structure.

Thus we have demonstrated that this method of User Driven Modelling/Programming solves problems of modelling and programming using tree based structures that can be represented to users in order that they can perform tasks such as calculation. The solution enables representing of results back to users to interact with, thus aiding their understanding of the model and of the problems users investigate.

8.3 Further Research

This section outlines future research on extending the areas of application for User Driven Modelling/Programming both for other types of problem within engineering and for other types of modelling. One possible application is the use of these techniques for scientific visualisation and modelling based on taxonomies.

8.3.1 Ontology Development

It is important not to stay limited on one ontology development environment but instead explore how ontologies can be developed using a range of tools and translated between each where necessary; (Garcia-Castro and Gomez-Perez, 2006) are testing this process. This would involve using and translating between tools such as those discussed in section 3.4 - ‘Ontology Based Modelling Solutions’, and others that may yet be developed.

Ciocoiu et al. (2000) and Horrocks (2002) consider the advantages of moving towards a more formal ontology, this was investigated in section 2.3.1 - ‘Ontologies for Modelling and Simulation’, and section 3.6 - ‘Ontologies and Semantic Web and their role in Modelling’. This would enable translation between all the layers in the ‘Layer Cake’ discussed at the end of section 2.3. More expressive semantic descriptions are possible through the use of the standard OWL dialects. These more expressive descriptions require sophisticated visualisation tools. Making use of a more formal ontology is the next major aim for the research behind this thesis. Creation of a formal ontology, while at the same time creating applications that model problems such as early stage design and cost, and interactive modelling environments for students, will widen the applicability of the research. This would enable further testing on ways ontologies can be used to solve problems, and how they are meaningful to people as well as being searchable by computer software. The intention is to enable online tagging of this ontology/ies and editing of it by users, in order to allow users and domain experts to be involved in the ontology construction.

So far the ontologies/taxonomies used in this thesis include traditional object oriented relationships such as child, parent, sibling, attribute, and instance. Though for this research instance means re-use of a class within an application rather than its object oriented meaning. There are other types of relationship that would need to be modelled in order to maximise the capabilities of software that would use the ontologies/taxonomies. Key relationships used within the object oriented programming domain between classes/objects have been modelled already. These key relationships depict families and aggregations of classes/objects that may share attributes and methods through inheritance. When physical items are represented, this can be translated to geometric diagrams. Semantic descriptions with more relationship types than the ones modelled so far would allow a more expressive depiction of a problem domain, and can aid some forms of search within a model.

8.3.2 Ontology Visualisation and Interaction

A major aspect of future work will be to develop a reversal of the 3 step process and ensure the translation can be performed from visualisation to ontology as well as ontology to visualisation. So this would involve Step 3 - ‘Visualisation and Interaction’ - Step 2 ‘Translation’ - Step 3 ‘Ontology’. This reverse translation has been examined but not prototyped in so much detail as the Step 1 to 3 process. It could be possible to provide high level facilities for end-users to edit ontologies, using this stepped translation methodology, and thus complete the circle of iterative communication between human and computer. Each side of the iterative diagram illustrated in Figure 15, would then have a double headed arrow. This is illustrated in Figure 62.

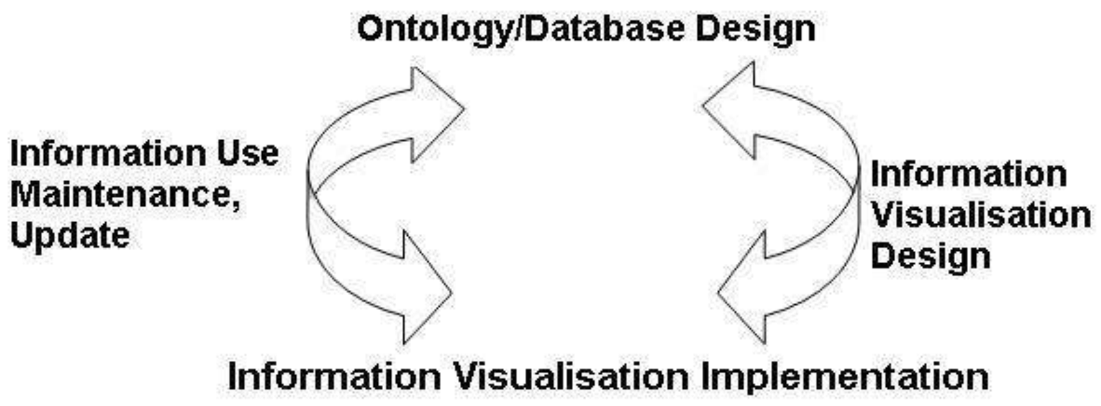

Ontology/Database Design
Information Use Maintenance, Update
Information Visualisation Design
Information Visualisation Implementation

Figure 62. Two Way Translation between users and computers

To encourage easier end-user interaction with Step 1 ontology creation it is important to make ontology editing easier. As mentioned in section 3.3 Protégé has OWL plug-ins available that provide extra capabilities for representing and visualising information e.g. Jambalaya (Ernst et al., 2003) for visualisation of knowledge and relationships. Enabling of web-based editing of ontologies would make ontology editing more accessible, and Leaver (2008) created an online ontology editor at the University of the West of England (section 6.4.4). Richer semantics could also make the editing process simpler by reducing complexity in the user interface.

Modelling of if-then choices formally using OWL and ontology editors should be provided. Elenius (2005) explains how OWL can be used for process modelling including for if-then choices:

“Process Modeling A powerful feature of OWL-S is the ability to model composite processes. A composite process is constructed from subprocesses that can in turn be composite, atomic, or simple. The control flow of a composite process is defined using control constructs, such as If-Then-Else, Sequence, and Repeat-Until. These constructs can be nested to an arbitrary depth.”

Elenius illustrates this with a screenshot of a composite process, its tree structure, and its graph representation.

SWRL (Semantic Web Rule Language) combining OWL and RuleML, and its use in modelling will also be investigated. This could be used for formally specifying the construction of equations and rules in a model and the relationships and constraints between items represented in an equation. Miller and Baramidze (2005) explain the SWRL language. An editing facility to model these equations and constraints, so that errors could be prevented, would improve the usability of future visual modelling systems created. Support for SWRL in Protégé and other ontology based systems will assist with the construction of a modelling system with sophisticated editing of rules (Miller and Baramidze, 2005). In addition, these rules could assist with provision of alternative interfaces such as an editable CAD type view, and with translating between these interfaces on the fly (e.g. from CAD type to tree/graph type).

8.3.3 Modelling and Simulation

Future work would build further on research in Semantic Web enabled modelling and simulation outlined in section 3.6. Miller and Baramidze (2005) examine efforts to develop mathematical semantic representations above the syntactical representations of MathML (referred to in 4.4). SWRL also has standardised arithmetic and comparison operators (Zhao and Liu, 2008). These languages should enable standardisation of the representation of mathematical expressions that relate nodes, and their values and expressions; this would seem to be a difficult problem as it needs a user interface that enables complex mathematical structures to be conveyed by language and/or diagrammatic visualisation. The next stage in the research after this thesis will be provision of constraints to prevent invalid mathematical expressions. Miller and Baramidze's DEMO system uses OWL to define a simulation and modelling class hierarchy. The first steps were taken towards creating an example simulation to demonstrate with a practical model how a web-based simulation can be provided based on an ontology, and how this can enable people to use interactive simulations on the web. This could be extended as described below.

8.3.4 Meta-Programming and Rule Based Programming

Meta-Programming and Rule-based languages (Wallace, 2003) could be used to develop an interface to an end-user programming environment. So far the automated output of code in such languages has been provided and automated output of machine independent code such as in XML, RDF, and SVG format. Research is needed into combining meta-language, rule-based and web and interoperability standard code to enable creation of modelling systems from this code automatically. This would be achieved in a similar way to that demonstrated in chapter 6, but in a more flexible and machine independent way. Chapter 6 demonstrated that the translation process can be used to create meta-code necessary for this, (in Step 1 and Step 2). So it remains to make use of this meta-code within Step 3 by sending it to appropriate visualisation and interaction tools (e.g. Simkin (Whiteside, 2008)). This could involve a Step 4 translation from meta-code, interoperable or web standard code, and/or rule based code to an automatically created model/program. A further possibility is to provide a meta-code version of the translation itself, so this can be machine independent.

8.3.5 Visualisation and Taxonomy Management

Through this thesis, ontologies and taxonomies has been used to represent computing structures and engineering information. Taxonomies have not yet been used in this research for managing biological information, though this was the purpose for which Linnaeus developed taxonomy representation. So, future research could provide an interactive visual taxonomy management system that uses the translation and interaction techniques developed in this thesis. The taxonomy will be used to structure, manage, and enable understanding of complex scientific information to enable scientists to collaborate using a systems approach. A circular layout would make it possible to visualise and navigate large trees more easily. The main subject would be editing and display of phylogenetic knowledge, this could make possible new insights. It is intended for this to act as a resource to link the research of biologists and environmental scientists.

8.3.6 Research Connectivity

It is intended that this research will also be applied to E-Learning and Collaboration, and to enabling wider participation in online communities. Work is beginning on a project to link companies and individuals mainly in the West of England aerospace/space industry and also to involve amateurs in this industry. This is a practical application for a web-based distributed constructionist approach (Resnick, 1996) to modelling and collaboration, and could also be used in teaching. As explained in 2.3.2 - ‘Semantic Web and Ontologies’, Berners-Lee and Fischetti (1999) also argue for collaborative interactivity, which they call ‘Intercreativity’. Research in e-learning would involve end-user programming enabled with Semantic Web technologies and with a visualisation and interactivity layer, as shown in Figure 63.

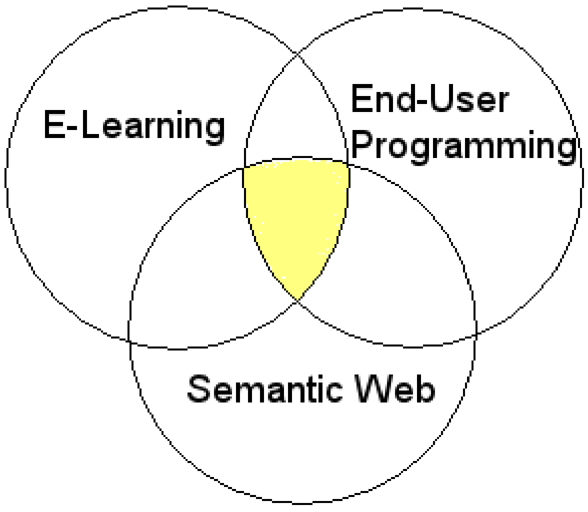


Figure 63. E-Learning, End-User Programming, and the Semantic Web

8.3.7 User/System Builder Management

Future work would be required to provide more sophisticated definition, and implementation of user types and rights. Through this thesis people were represented in 3 categories - ‘System Builders’, ‘Model Builders’, and ‘Model Users’. However, these categories would need to be further divided into sub-categories, and these be managed visually and interactively. Management of use, based on an ontology of people constructing or using systems would help enable management and collaboration on larger scales.

8.4 Final Statement

The question examined was ‘to what extent was it possible to improve user-driven collaborative software development through interaction with diagrams and without requiring users to learn particular computer languages?’

Within this thesis it was argued that there is a need for software developers to create programs that enable users to solve problems themselves. In effect this involves production of a system to create systems. This approach can widen programming participation by including computer literate non-programmers. This is a reaction to the increased complexity of real world problems and software systems, which makes development of some types of software solutions impractical without greater involvement from end-users. It is difficult for developers to foresee every need of users and use of the software produced, so it makes sense to enable more end-user customisation, especially for modelling of complex engineering, science, and business problems. It is also argued that the research for this thesis has been a step towards making end-user programming possible particularly for modelling. The research ideas cover a wide range but the core research is all about simplifying software development.

There are numerous obstacles such as current technologies and languages providing insufficient support for User Driven Modelling/Programming systems. However, research on this can improve the development and use of such technologies. Accessible model creation systems are needed that enable intuitive construction and sharing of models.

For proving the hypothesis that it is possible to create an end-user programming environment, usable by non programmers, it has been found that structuring and relating of information is all important in this solution. To achieve this, it was only necessary to link the information visually via equations, and store these results for reuse and collaboration. Semantic Web standardisation could enable the range of model building/use and collaboration possible, making the solution more generic. If users can understand and navigate relationships, and add new relationships they could model most problems. It was important to design a visual interface that is intuitive to use, and allows for proper interpretation of the results. Feedback has indicated that users can navigate this structure and manipulate it. This is preferable to ‘black box’ solutions that hide information. There are no dead ends or permanent blocks to expanding and improving this approach. To make the system easier to use it is necessary to trial continually better interfaces, and to assist by providing guidance to the user. There was not sufficient time and resources to expand this research much to areas outside engineering modelling, but there is scope for researchers to improve end-user programming for engineering modelling systems, and to expand the research into other areas. Any problem that requires structured visualisation and/or equation based calculation would benefit from this approach.

References

Abraham, R., Erwig, M., 2007. Exploiting Domain-Specific Structures For End-User Programming Support Tools. *In: End-User Software Engineering Dagstuhl Seminar February 2007*.

Al-Khalifa, H. S., Davis, H. C., 2006. Harnessing the Wisdom of Crowds: How To Semantically Annotate Web Resource Using Folksonomies. *In: Proceedings of IADIS Web Applications and Research* (WAR2006).

Ambler, S. W., (2003) *The Object-Relational Impedance Mismatch* [online]. Available from: http://www.agiledata.org/essays/impedanceMismatch.html [Accessed 25 July 2011].

Anderson, P. Technology and Standards Watch (2007). JISC (Joint Information Systems Committee) *What is Web 2.0? Ideas, technologies and implications for education* [online]. Available from: http://www.jisc.ac.uk/media/documents/techwatch/tsw0701b.pdf [Accessed 25 July 2011].

Aragones, A., Bruno, J., Crapo, A., Garbiras M., 2006. An Ontology-Based Architecture for Adaptive Work-Centered User Interface Technology. *In*: *Jena User Conference, 2006, Bristol, UK*.

Baclawski, K., Mieczyslaw, K., Kogut, P., Hart, L., Smith, J., Holmes, W., Letkowski, J., Aronson, M., 2001. Extending UML to Support Ontology Engineering for the Semantic Web. *In*: *Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, pp 342-360.

Bechhofer, S., Carrol, J., 2004. Parsing owl dl: trees or triples?. *In: Proceedings of the 13th international conference on World Wide Web*, NY, USA, pp 266-275.

Begel, A., 2007. End-user Programming for Scientists: Modeling Complex Systems. *In: End-User Software Engineering Dagstuhl Seminar February 2007*.

Berners-Lee, T., (2000) *Semantic Web on XML* – Slide 10 [online]. Available from: http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html [Accessed 29 April 2011].

Berners-Lee, T., Fischetti, M., 1999. *Weaving the Web*. Harper San Francisco; Paperback: ISBN:006251587X

Berners-Lee, T., Hall, W., Hendler, J., Shadbolt, N., Weitzner, D. J., 2006. Creating a Science of the Web. *Science 11 August 2006*:Vol. 313. no. 5788, pp. 769 - 771.

Berners-Lee, T., Hendler, J., Lassila, O., 2001. The Semantic Web. *Scientific American.* May 17, 2001.

Bishop, J., 2006. Multi-platform user interface construction: a challenge for software engineering-in-the-small. *In: International Conference on Software Engineering, Proceedings of the 28th international conference on Software engineering* pp 751-760.

Bru, C., Scanlan, J., Hale, P., 2003. Generation and Cognitive Representation of Cost Information over a Network *In*: *Proceedings of the 9th International Conference on Concurrent Enterprising*, Espoo Finland pp 301-309

Bru, C., Scanlan, J., Hale, P., Dunkley, M., 2002. Visualisation of Cost Information. *In*: *Proceedings of the 9th ISPE International Conference on Concurrent Engineering Advances in Concurrent Engineering, Cranfield, UK,* pp 829-838.

Bru, C., Scanlan, J., Hale, P., 2004. Visualization of Cost Information, *International Journal of Agile Manufacturing*, 7(1), pp 53-59.

Burnett, M. M., Engels, G, Myers, B. A., Rothermel, G., 2007. End-User Software Engineering Executive Summary. *In: End-User Software Engineering Dagstuhl Seminar February 2007.*

Carroll J., Turner, D., (2008) *The Consistency of OWL Full,* HP Labs Technical Report, Bristol, UK, Computer Laboratory, Cambridge Univ [online]. Available. http://lists.w3.org/Archives/Public/www-archive/2008May/att-0053/58.pdf [Accessed 25 July 2011].

Cayzer, S., 2004. Semantic Blogging and Decentralized knowledge Management. *Communications of the ACM*. Vol. 47, No. 12, Dec 2004, pp. 47-52. ACM Press.

Chan, S. C. F., Dillon, T., Ng, V. T. Y., 2003. Exchanging STEP Data through XML-Based Mediators. *Concurrent Engineering,* 111, pp 55-64.

Chelsom, J. J., Summers, R., Pande, I., Gaywood, I., 2011. Ontology-driven Development of a Clinical Research Information System, *In: CBMS 2011, The 24th International Symposium on Computer-Based Medical Systems, June 27th-30th, University of the West of England, Bristol, UK.*

Cheung, W. M., Maropoulos, P. G., Gao, J. X., Aziz, H., 2005. Ontological Approach for Organisational Knowledge Re-use in Product Developing Environments. *In: 11th International Conference on Concurrent Enterprising - ICE 2005,* University BW Munich, Germany.

Cheung, W. M., Matthews, P. C., Gao, J. X., Maropoulos, P. G., 2007. Advanced product development integration architecture: an out-of-box solution to support distributed production networks. *International Journal of Production Research* March 2007.

Ciocoiu, M., Gruninger, M., Nau, D. S., 2000. Ontologies for Integrating Engineering Applications. *Journal of Computing and Information Science in Engineering*, 1(1) pp 12-22.

Corcho, O., Fernández-López, M., Gómez-Pérez, A., 2003. Methodologies, Tools and Languages For Building Ontologies. Where is their Meeting Point?. *Data and Knowledge Engineering*, 46, pp 41-64.

Corcho, O., Gómez-Pérez, A., 2000. A Roadmap to Ontology Specification Languages. *In*: *Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management, Chicago,* USA.

Coutaz, J., 2007. Meta-User Interfaces for Ambient Spaces: Can Model-Driven-Engineering Help?. *In: End-User Software Engineering Dagstuhl Seminar February 2007*.

Crapo, A. W., Waisel, L. B., Wallace, W. A., Willemain, T. R., 2002. Visualization and Modelling for Intelligent Systems. *In:* C. T. Leondes, ed. *Intelligent Systems: Technology and Applications*, Volume I Implementation Techniques, 2002 Chapter 3 pp 53-85.

Cypher, A., Halbert D. C., Kurlander D., Lieberman, H., Maulsby, D., Myers, B. A., Turransky A., 1993. *Watch What I Do Programming by Demonstration*. MIT Press, Chapter 1 [online]. Available from: http://www.acypher.com/wwid/Chapters/01Pygmalion.html [Accessed 25 July 2011] ISBN:0262032139.

Davies, J., Fensel, D., van Harmelen, F., 2002. *On-To-Knowledge: Semantic Web enabled Knowledge Management*, (John Wiley and Sons Ltd, ISBN: 0470848677).

De Souza, C., 2007. Designers Need End-User Software Engineering. *In: End-User Software Engineering Dagstuhl Seminar February 2007*.

Dmitriev, S., (2006) *Language Oriented Programming: The Next Programming Paradigm* [online]. Available from: http://www.onboard.jetbrains.com/is1/articles/04/10/lop/ [Accessed 25 July 2011].

Duverlie P., Castelain J. M., 1999, Cost Estimation During Design Step: Parametric Method versus Case Based Reasoning Method. *The International Journal of Advanced Manufacturing Technology*, Vol 15: pp 895-906.

Eaglesham, M., 1998. *A Decision Support System for Advanced Composites Manufacturing Cost Estimation*. Ph.D. thesis, Virginia Polytechnic Institute and State University.

Eklund P, Roberts N, Green S, 2002. OntoRama: Browsing RDF Ontologies using a Hyperbolic-style Browser, The First International Symposium on Cyber Worlds, CW02, Theory and Practices, IEEE Press. (2002) pp 405-411.

Elenius, D., 2005. The OWL-S Editor - A Domain-Specific Extension to Protégé. *In: 8th Intl. Protégé Conference* - July 18-21, 2005 - Madrid, Spain.

Elrad, T., Aksit, M., Kiczales, G., Lieberherr, K., Ossher, H., 2001. Discussing aspects of AOP. *Communications of the ACM*, 44(10) pp 33-38.

Eng, N., Salustri, F. A., 2006. “Rugplot” Visualization for Preliminary Design. *In: CDEN 2006 3rd CDEN/RCCI International Design Conference* University of Toronto, Ontario, Canada.

Engels, G., 2007. Model-Driven Development for End-Users, too!? . *In: End-User Software Engineering Dagstuhl Seminar February 2007*.

Enoksson, N. (2006) *Serverside Solution for Conceptual Browsing on the Semantic Web*. MSc. Dissertation, Stockholm University.

Erdmann, M., Studer, R. 1999. Ontologies as Conceptual Models for XML Documents. *In*: *Proceedings of the 12th Workshop on Knowledge Acquisition, Modelling and Management (KAW’99)*, Banff, Canada, October 1999.

Ernst, N. A., Storey, M., Allen, P., Musen, M., 2003. Addressing cognitive issues in knowledge engineering with Jambalaya. *In: Workshop on Visualization in Knowledge Engineering at KCA.*

Erwig, M., Abraham, R., Cooperstein, I., Kollmansberger S., 2006. Automatic Generation and Maintenance of Correct Spreadsheets?. *In: Proceedings of the 27th international conference on Software Engineering, St. Louis, MO, USA* pp 136-145.

Fensel, D., Van Harmelen, F., Horrocks, I., McGuinness, D., Patel-Schneider, P. F., 2001. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2), pp 38-45.

Fischer, G., 2007. Meta-Design: A Conceptual Framework for End-User Software Engineering. *In: End-User Software Engineering Dagstuhl Seminar February 2007.*

Fluit, C., Sabou, M., Harmelen, F. V., 2003. Supporting User Tasks through Visualisation of Light-weight Ontologies. *In:* S. Staab and R. Studer, ed. *Handbook on Ontologies in Information Systems,* Springer-Verlag pp 415-434.

Frankel, D., Hayes, P., Kendall, E., McGuinness, D., 2004. The Model Driven Semantic Web. *In*: *1st International Workshop on the Model-Driven Semantic Web (MDSW2004) Enabling Knowledge Representation and MDA® Technologies to Work Together.*

Garcia-Castro R., Gomez-Perez A., 2006. Interoperability of Protégé using RDF(S) as interchange language. *In*: *9th Intl. Protégé Conference, July 23-26, 2006* - Stanford, California.

Gray, J., Zhang, J., Lin, Y., Roychoudhury, S., Wu, H., Sudarsan, R., Gokhale, A., Neema, S., Shi, F., Bapty, T., 2004. Model-Driven Program Transformation of a Large Avionics Framework. *In*: *Third International Conference on Generative Programming and Component Engineering GPCE*, pp 361-378.

Green, S., Beeson, I., Kamm, R., 2007. Process architectures and process models: opportunities for reuse. *In: 8th Workshop on Business Process Modeling, Development, and Support BPMDS07 and CAiSE'07* 11-15 June 2007, Trondheim, Norway.

Gross, M. D., 2007. Designers Need End-User Software Engineering. *In: End-User Software Engineering Dagstuhl Seminar February 2007*.

Gruber, T. R., 1993. A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, Vol 5 pp 199-220.

Gruber, T. R., 1993. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *In:* N. Guarino and R. Poli, ed. *Formal Ontology in conceptual Analysis and Knowledge Representation.* Kluwer Academic Publishers.

Guibert, N., Girard, P., Guittet, L., 2004. Example-based Programming: a pertinent visual approach for learning to program. *In*: *Proceedings of the working conference on Advanced visual interfaces*. pp 358-361 - ISBN:1-58113-867-9.

Gutowski, T. G., Haffner, S. M., Pas J. W., 2001. Web Based Cost Estimation for Advanced Composites. *In*: *Proceedings of the 2002 NSF Design, Service and Manufacturing Grantees and Research Conference*. San Juan, Puerto Rico, January 2002.

Hale, P., Scanlan, J., Bru, C., 2003, Design and Prototyping of Knowledge Management Software for Aerospace Manufacturing. *In*: *10th ISPE International Conference on Concurrent Engineering.*

Hale, P., (2009) *Interactive Examples* [online]. Available from: http://www.cems.uwe.ac.uk/~phale/ [Accessed 25 July 2011].

Hanna, K., 2005. A document-centered environment for Haskell. *In*: *17th International Workshop on Implementation and Application of Functional Languages IFL 2005* Dublin, Ireland - September 19-21 2005.

Horrocks, I., 2002. DAML+OIL: a Reason-able Web Ontology Language. *In: proceedings of the Eighth Conference on Extending Database Technology (EDBT 2002)* March 24-28 2002, Prague.

Horrocks, I., Patel-Schneider, P. F., van Harmelen, F., 2003. *From SHIQ and RDF to OWL: The making of a web ontology language*. Journal of Web Semantics, Vol 1(1), pp 7-26.

Huber, G. P., 2001, Transfer of knowledge in knowledge management systems: unexplored issues and suggested studies. *European Journal of Information Systems*, Vol 10 pp 80-88.

Hudak, P., Hughes, J., Jones, S. P., Wadler, P., 2007. A History of Haskell: being lazy with class. *In: The Third ACM SIGPLAN History of Programming Languages Conference (HOPL-III)* San Diego, California, June 9-10, 2007.

Huhns, M., 2001. Interaction-Oriented Software Development. *International Journal of Software Engineering and Knowledge Engineering*, 11, pp 259-279.

Hunter, A., (2002) *Engineering Ontologies* [online]. Available from: http://www.cs.ucl.ac.uk/staff/a.hunter/tradepress/eng.html [Accessed 25 July 2011].

Jackiw, R. N., Finzer, W. F., 1993. The Geometer's Sketchpad:Programming by Geometry. *In:* A. Cypher, ed. *Watch What I Do: Programming by Demonstration*. MIT Press, Chapter 1 [online]. Available from: http://www.acypher.com/wwid/Chapters/13Sketchpad.html [Accessed 26 July 2011] ISBN:0262032139.

Jackson, D., 2006. Software Abstractions: Logic, Language, and Analysis. MIT Press. ISBN 978-0-262-10114-1.

Johnson, P., 2004. Interactions, collaborations and breakdowns. *In: ACM International Conference Proceeding Series; Proceedings of the 3rd annual conference on Task models and diagrams* Vol 86 Prague, Czech Republic.

Kim, T., Lee, T., Fishwick, P., 2002. A Two Stage Modeling and Simulation Process for Web-Based Modeling and Simulation. *ACM Transactions on Modeling and Computer Simulation*, 12(3), 230-248.

Ko, A. J., 2007. Barriers to Successful End-User Programming. *In: End-User Software Engineering Dagstuhl Seminar February 2007*.

Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., Smith, J., 2002. UML for Ontology Development. *The Knowledge Engineering Review* Vol 17(1) pp 61-64.

Kuljis, J., Paul, R. J., 2001. An appraisal of web-based simulation: whither we wander?. *Simulation Practice and Theory*, 9, pp 37-54.

Lau, H. C. W., Ning, A., Pun, K. F., Chin, K. S., Ip, W. H., 2005. A knowledge-based system to support procurement decision. *Journal of Knowledge Management*, 9(1), pp 87-100.

Leaver, N. (2008) *Using RDF as an Enabling Technology*. MSc. Dissertation, University of the West of England, Bristol.

Lemos, M., (2009) *MetaL: An XML based Meta-Programming language* [online]. Available from: http://www.meta-language.net [Accessed 25 July 2011].

Letondal, C., 2005. Participatory Programming: Developing programmable bioinformatics tools for end-users. In H. Lieberman, F. Paterno, & V. Wulf (Eds.), End-User Development. Springer/Kluwer Academic Publishers.

Lieberman, H., 2000. Your Wish is My Command: Giving Users the Power to Instruct their Software, Morgan Kaufmann.

Lieberman, H., 2007. End-User Software Engineering Position Paper. *In: End-User Software Engineering Dagstuhl Seminar February 2007*.

Macías, J. A., Castells, P., 2004. An EUD Approach for Making MBUI Practical. *In*: *Intelligent User Interfaces and Computer-Aided Design of User Interfaces Conference (IUI/CADU'2004). Funchal, Madeira Island, Portugal, 13-16 January*.

McGuinness, D. L., 2003. Ontologies Come of Age. *In:* Dieter Fensel, Jim Hendler, Henry Lieberman, and Wolfgang Wahlster, ed. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2003.

Mihalcea, R., Hugo, L., Lieberman, H., NLP (Natural Language Processing) for NLP (Natural Language Programming) *In: International Conference on Computational Linguistics and Intelligent Text Processing*, Mexico City, Springer Lecture Notes in Computer Science, February 2006.

Miller, J. A., Baramidze, G., 2005. Simulation and the Semantic Web. *In. Proceedings of the 2005 Winter Simulation Conference*.

Miller, J., Fishwick, P. A., Taylor, S. J. E., Benjamin, P., Szymanski, B., 2001. Research and commercial opportunities in Web-Based Simulation. *Simulation Practice and Theory*, 9, pp 55-72.

Murphy, G. C., Walker, R. J., Baniassad, E. L. A., Robillard, M. P., Lai, A., Kersten, M. A., 2001. Does aspect-oriented programming work?, *Communications of the ACM*, Vol 44(10) (October 2001) pp 75 - 77, ISSN:0001-0782.

Naeve, A., 2005, The Human Semantic Web – Shifting from Knowledge Push to Knowledge Pull. *International Journal of Semantic Web and Information Systems* (IJSWIS), Vol 1(3) (July-September 2005) pp 1-30.

Noy, N.F., McGuinness, D., 2004. Semantic Integration: A Survey Of Ontology-Based Approaches. *SIGMOD Record, Special Issue on Semantic Integration*, 33(4).

Nurminen, J. K., Karaonen, O., Hatonen, K., 2003. What makes expert systems survive over 10 years-empirical evaluation of several engineering applications. *Expert Systems with Applications* 24(2) pp 199-211.

Olsson, E., 2004. What active users and designers contribute in the design process. *Interacting with Computers* 16, pp 377-401.

Paine, J., 2003. Spreadsheet Structure Discovery with Logic Programming, *In*: *Proceedings of European Spreadsheet Risks Interest Group EuSpRIG* Greenwich, England.

Palanque, P., Bastide R., 2003. *UML for Interactive Systems: What is Missing INTERACT 2003 Closing the Gaps: Software Engineering and Human-Computer Interaction* Zürich, Switzerland.

Panko, R. P., 2000. Spreadsheet Errors: What We Know, What We Think We Can Do. *Proceedings of European Spreadsheet Risks Interest Group EuSpRIG, Greenwich, England*, pp 7–17.

Papert, S., Harel, I., 1991. Situating Constructionism An essay. *In:* book Constructionism (Ablex Publishing Corporation, 1991) [online]. Available from: http://www.papert.org/articles/SituatingConstructionism.html [Accessed 25 July 2011].

Paternò, F., 2005. Model-based tools for pervasive usability. *Interacting with Computers*, 17(3), pp 291-315.

Peirce, C.S. (1906) *Prolegomena to an Apology for Pragmaticism* [online]. Available from: http://www.existentialgraphs.com/peirceoneg/prolegomena.htm [Accessed 25 July 2011].

Rajalingham, K., Chadwick, D. R., Knight, B., 2001. Classification of Spreadsheet Errors. In: Symp. of the European Spreadsheet Risks Interest Group (EuSpRIG).

Reed, J. A., Follen, G. J., Afjeh, A. A., 2000. Improving the Aircraft Design Process Using Web-Based Modeling and Simulation. *ACM Transactions on Modeling and Computer Simulation*, 10(1), pp 58-83.

Repenning, A., 2007. End-User Design. *In: End-User Software Engineering Dagstuhl Seminar February 2007*.

Resnick, M., 1996. Distributed Constructionism. *In: Proceedings of the International Conference on the Learning Sciences Association for the Advancement of Computing in Education*, Northwestern University (accepted: March 1996; published: July 1996) [online]. Available from: http://llk.media.mit.edu/papers/Distrib-Construc.html [Accessed 25 July 2011].

Scaffidi, C., Shaw, M., Myers, B., 2005. Estimating the Numbers of End-users and End-user Programmers. *In: IEEE Symposium on Visual Languages and Human-Centric Computing, (VL/HC'05): 207-214 Dallas, Texas*.

Scanlan, J., Hill, T., Marsh, R., Bru, C., Dunkley, M., Cleevely, P., 2002. *Cost Modelling for Aircraft Design Optimization, Journal of Engineering Design*, 13(3), pp 261-269.

Scanlan, J., Rao, A., Bru, C., Hale, P., Marsh, R., 2006. DATUM Project: Cost Estimating Environment for Support of Aerospace Design Decision Making. *Journal of Aircraft,* 43(4).

Schmitz, P., 2006. Inducing ontology from Flickr tags. *In: WWW2006 Conference, Edinburgh, UK.* May 22-26, 2006.

Schrage, M., (1991) *Spreadsheets: Bulking Up On Data* [online]. Available from: http://www.systems-thinking.org/buod/buod.htm Los Angeles Times [Accessed 25 July 2011].

Shim, J.P., Warkentin, M., Courtney, J. F., Power, D J., 2002, Past, present, and future of decision support technology. *Decision Support Systems 33* pp 111-126.

Smith, D. C., 1977. *A Computer Program to Model and Stimulate Creative Thought*. Basel: Birkhauser.

Spahn, M., Scheidl, S., Stoitsev, T., 2007. End-User Development Techniques for Enterprise Resource Planning Software Systems. *In: End-User Software Engineering Dagstuhl Seminar February 2007*.

Sternemann, K. H., Zelm, M., 1999. Context sensitive provision and visualisation of enterprise information with a hypermedia based system, *Computers in Industry* Vol 40 (2) pp 173-184.

Storey, M., Lintern, R., Ernst, N., Perrin, D., 2004, Visualization and Protégé *In*: *7th International Protégé Conference* - July 2004 - Bethesda, Maryland.

Sutton, D. C., 2001. What is knowledge and can it be managed?. *European Journal of Information Systems*, Vol 10 pp 72-79.

Tufte, E., R., 1990. *Envisioning Information*. Graphics Press.

Uschold, M., 2003. Where are the semantics in the semantic web? *AI Magazine* Vol 24 (3) pp 25-36.

Uschold, M., 2006. Ontologies Ontologies Everywhere - but Who Knows What to Think? *In: 9th Intl. Protégé Conference* - July 23-26, 2006 - Stanford, California.

Uschold, M., Gruninger, M., 2004. Ontologies and Semantics for Seamless Connectivity. *In: Association for Computer Machinery - Special Interest Group on Management of Data - SIGMOD Record December*, 33(4).

Vernazza, L., 2007. Himalia: Model-Driven User Interfaces Using Hypermedia, Controls and Patterns *In:* IFAC/IFIP/IFORS IEA Symposium - Analysis, Design, and Evaluation of Human-Machine Systems Seoul, Korea - September 4-6th 2007 - International Federation of Automatic Control.

Volz, R., Oberle, D., Staab, S., Motik, B., 2003. KAON SERVER - A Semantic Web Management System. *In WWW (Alternate Paper Tracks).*

Wakeling, 2007. Spreadsheet functional programming, *Journal of Functional Programming*, 17(1)(January 2007) pp 131-143 - ISSN:0956-7968.

Wallace, C., 2003. Using Alloy in process modelling. *Information and Software Technology*, 45(15), pp 1031-1043.

Whiteside, S., (2009) *Simkin the embeddable scripting language* [online]. Available from: http://www.simkin.co.uk/ [Accessed 25 July 2011].

Wikipedia (2009) *Metaprogramming* [online]. Available from: http://en.wikipedia.org/wiki/Metaprogramming [Accessed 25 July 2011].

Willemain, T. R., Powell S. G., 2006, How novices formulate models. Part II: a quantitative description of behaviour, *Journal of the Operational Research Society*, pp 1-12.

Zhao, W., and Liu, J.K., 2008. OWL/SWRL representation methodology for EXPRESS-driven product information model Part I. Implementation methodology, *Computers in Industry* - Article in Press, Corrected Proof [online]. Available from: http://www.sciencedirect.com/science/article/B6V2D-4S7HWFC-1/2/8c720905e75881248d9df9fc64e6824e [Accessed 25 July 2011].

Chapter 9 Appendix

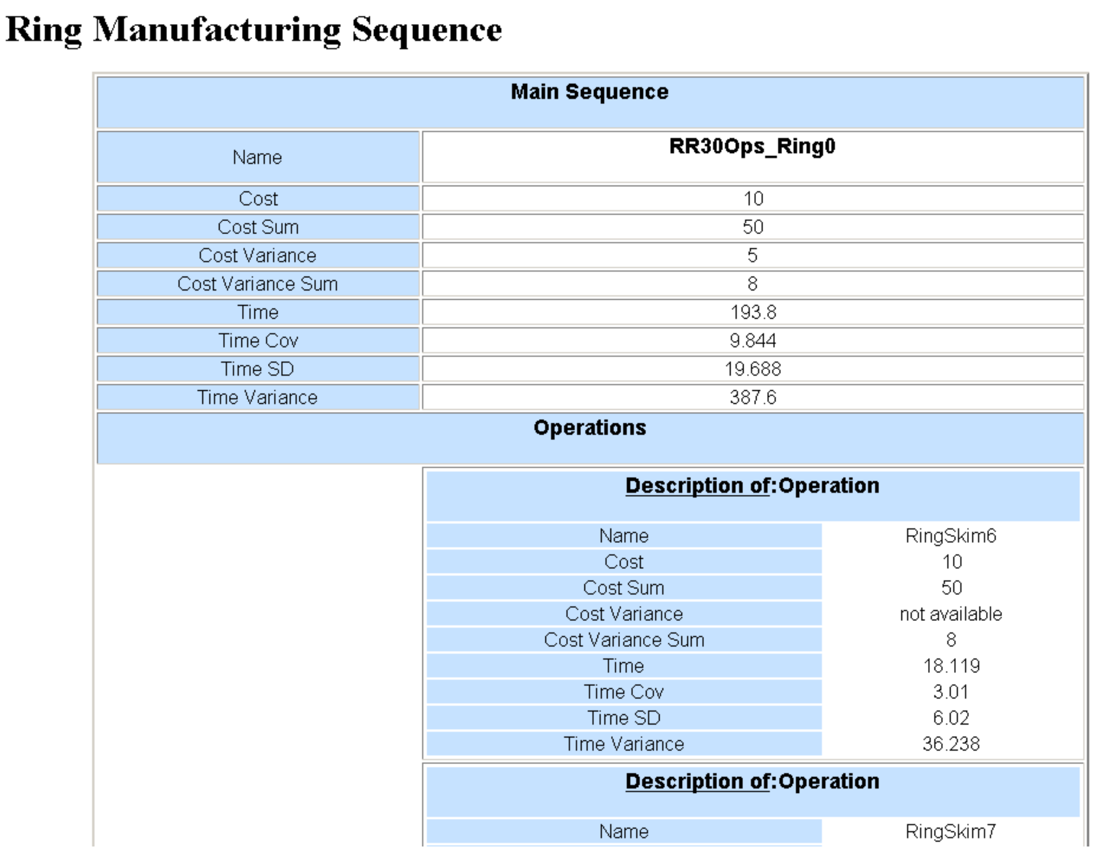
The following examples show how standardised ontologies can be represented and visualised, Stylesheets are used to provide this representation and visualisation. This makes it possible to share models with domain specific (e.g. engineering) agreed semantics above the level of generic agreed semantics. This can make it practical to share and re-use ontologies and models.

9.1 Process Modelling

9.1.1 DATUM Process Modelling Example

Varieties of XML that allow for additional semantics such as standardised representation of inheritance relationships, attributes and sequences or lists of items were investigated. RDF and DAML/OIL (DARPA Agent Markup Language/Ontology Inference Layer) are represented using XML and allow for this additional layer of semantics. These syntaxes allow for explicit representation of relationships such as class/subclass, attributes, and sequences. These information representation languages were used to represent Product Data Structures and Process, Materials, and Tooling libraries. An example[38] of a Process Sequence represented by the layered XML/RDF/RDF Schema standard and displayed using a stylesheet is shown in Figure 64. The semantics were agreed with Rolls-Royce in the DATUM project (Scanlan et al., 2006), and the representation was displayed using an XSL (eXtensible Stylesheet Language) stylesheet :-

[38] Hale, P. (2009) *Ring Manufacturing Sequence* [online]. Available from: http://www.cems.uwe.ac.uk/~phale/XMLDemonstrators/rdfring.xml [Accessed 25 July 2011].



Ring Manufacturing Sequence

Main Sequence	
Name	**RR30Ops_Ring0**
Cost	10
Cost Sum	50
Cost Variance	5
Cost Variance Sum	8
Time	193.8
Time Cov	9.844
Time SD	19.688
Time Variance	387.6

Operations

Description of:Operation

Name	RingSkim6
Cost	10
Cost Sum	50
Cost Variance	not available
Cost Variance Sum	8
Time	18.119
Time Cov	3.01
Time SD	6.02
Time Variance	36.238

Description of:Operation

Name	RingSkim7

Figure 64. Representation of Engine Ring Manufacturing Sequence

Only demonstration test values without units are shown in the above illustration. The above example could be extended to additional layered representation of DAML+OIL and/or OWL.

9.2 Process Specification Language - PSL

PSL was discussed in section 3.5.1 - ‘Engineering Domain Specific Standards’.

PSL can use XML, RDF (Resource Description Framework), and its own semantics to add a layer of engineering meaning to these Semantic Web languages for communication between process modelling tools, and for use in defining ontologies. These PSL examples make use of these languages as recommended in this publication[39] :-

“1. Use RDF Schema to represent the objects used in a process.

2. Represent timepoints as sequentially ordered groups of elements, with each timepoint element having a unique identifier. If the XML application uses a Document Type Definition (DTD), the unique

---

[39] Schlenoff, C., Gruninger, M., Tissot, F., Valois, J., Lubell, J., Lee, J., (1999) *The Process Specification Language (PSL) Overview and Version 1.0 Specification. NIST Internal Report (NISTIR) 6459, National Institute of Standards and Technology. Appendix C: Mapping PSL Concepts to the eXtensible Markup Language (XML) Representation* pp 76-79 [online]. Available from: http://www.mel.nist.gov/msidlibrary/doc/nistir6459.pdf#page=76 [Accessed 25 July 2011].

identifier should be represented using an ID attribute so that references to the timepoint can be made using IDREF.

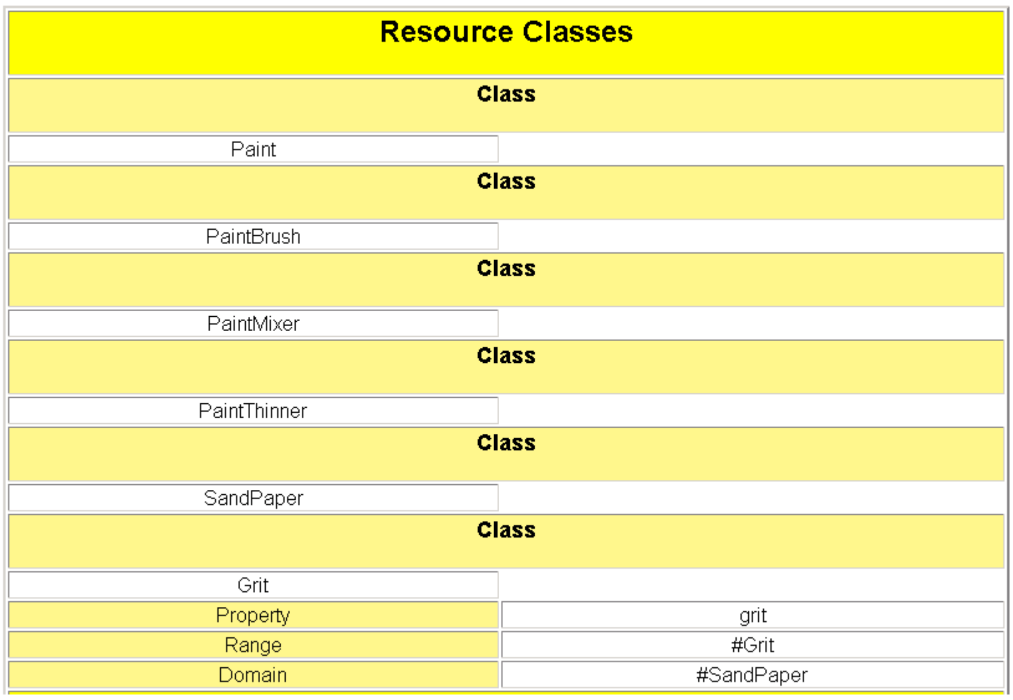
3. For each activity, specify a unique identifier (with an ID attribute if using a DTD) and an activity name. If the activity contains subactivities, specify these within a container element. If the activity has no subactivities, specify the resources used with references to the appropriate class defined in the RDF Schema.

4. Specify occurrences of activities in sequential order with sub-activities enclosed inside parent activities. Each activity occurrence should have a beginning and ending time point and, if it cannot be decomposed into sub-activities, a list of RDF-defined resource instances it uses."

The figures below[40] show a section from an example PSL process sequence based on the NIST example representation[39], and rendered using an XSL (eXtensible Stylesheet Language) stylesheet. This example is just to illustrate the standard, the example could be expanded to be a full ontology and allow visualisation, navigation, and interactivity.

The first step is to declare the resources; this is shown below, Figure 65 :-

[40] Hale, P. (2008) *Process Specification Language Example* [online]. Available from: http://www.cems.uwe.ac.uk/~phale/XMLDemonstrators/psl.xml [Accessed 25 July 2011].

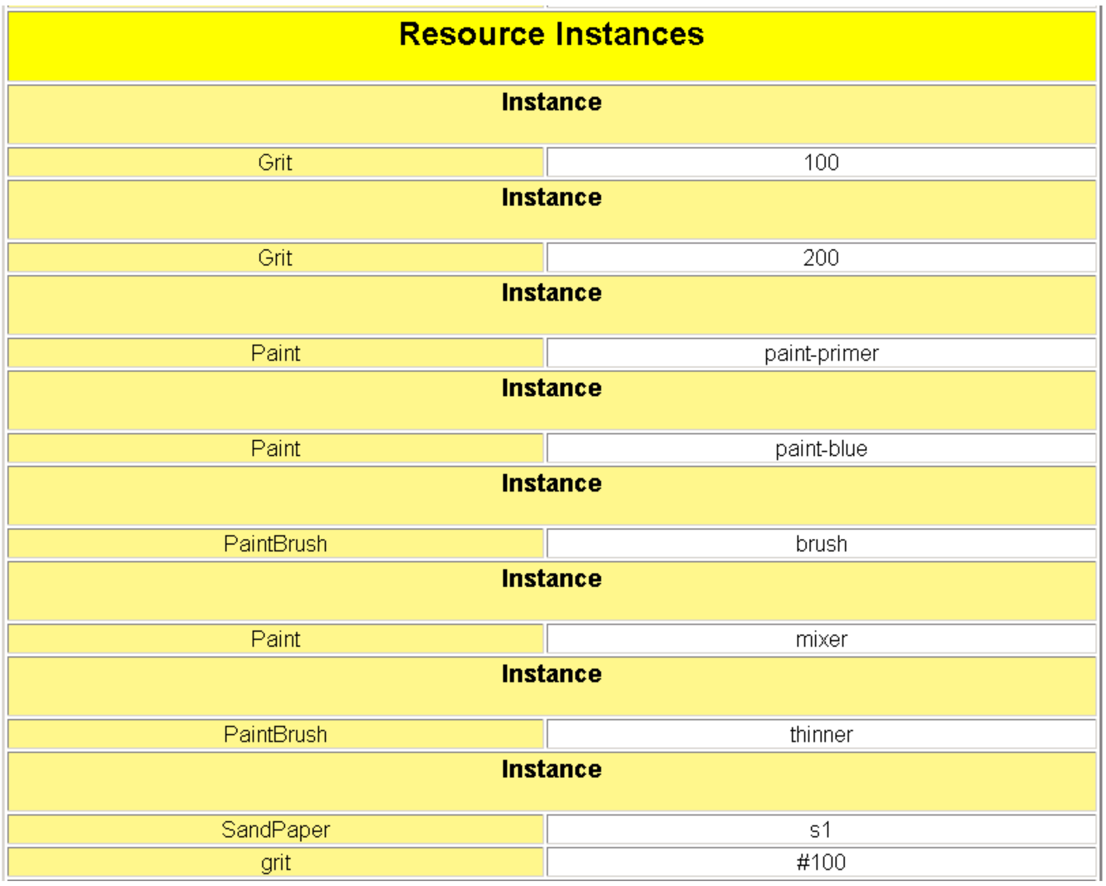


Process Specification Language Example

Resource Classes	
Class	
Paint	
Class	
PaintBrush	
Class	
PaintMixer	
Class	
PaintThinner	
Class	
SandPaper	
Class	
Grit	
Property	grit
Range	#Grit
Domain	#SandPaper

Figure 65. Section from PSL Process Taxonomy and sequence rendered with stylesheet - PSL Class Definition

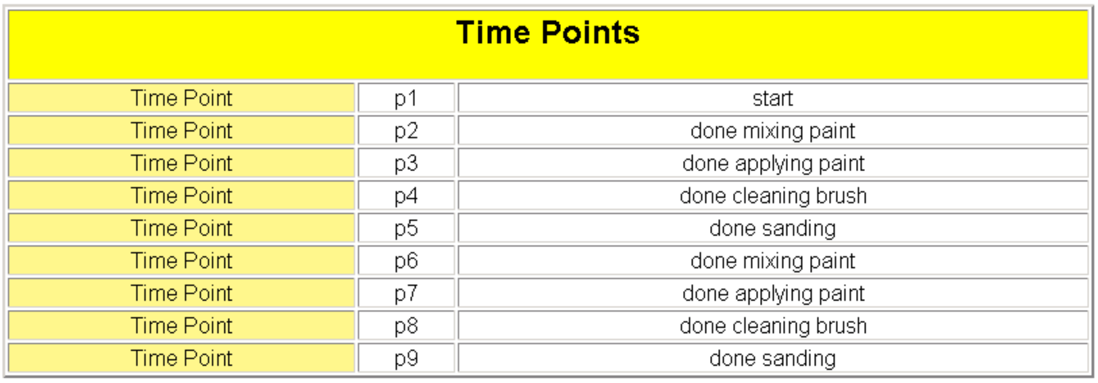
Instances of this class can then be declared; this is illustrated next, Figure 66 :-



Resource Instances	
Instance	
Grit	100
Instance	
Grit	200
Instance	
Paint	paint-primer
Instance	
Paint	paint-blue
Instance	
PaintBrush	brush
Instance	
Paint	mixer
Instance	
PaintBrush	thinner
Instance	
SandPaper	s1
grit	#100

Figure 66. PSL Instance Creation

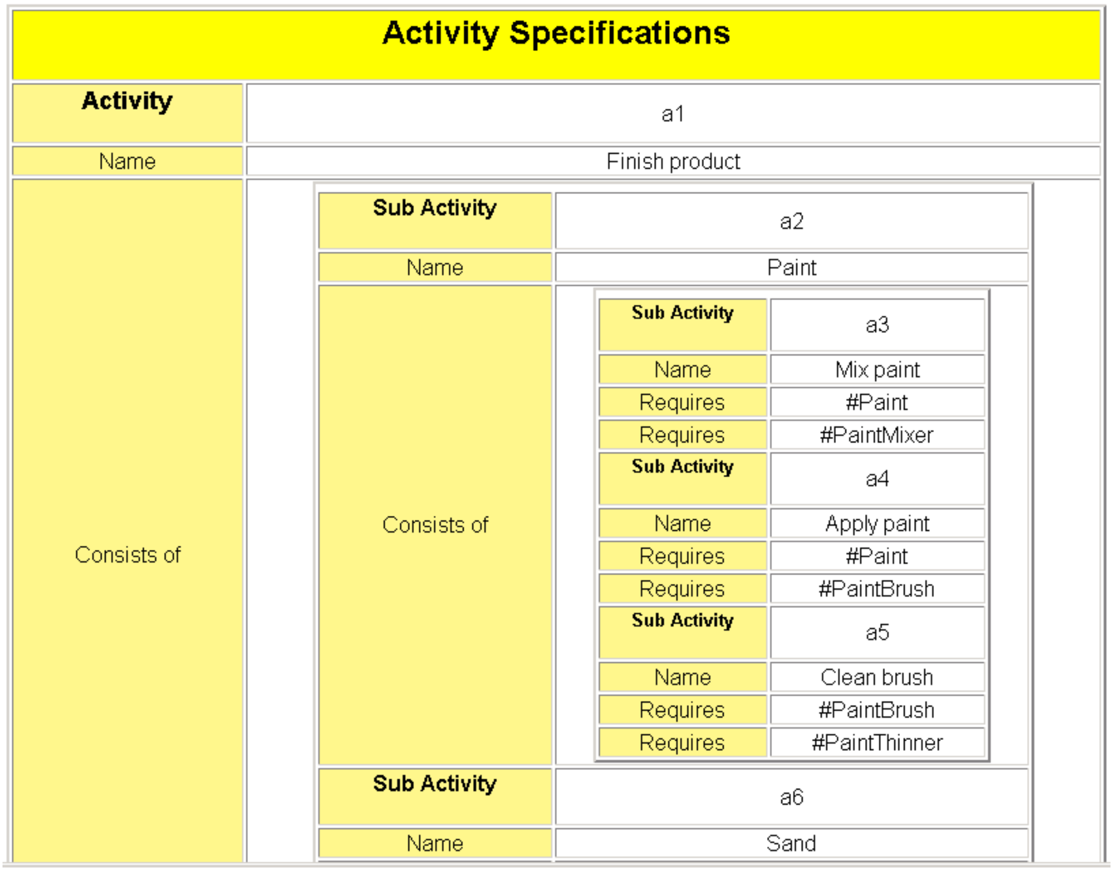
Time points are then created to make it possible to create a sequence of activities; this is shown below, Figure 67 :-



Time Points		
Time Point	p1	start
Time Point	p2	done mixing paint
Time Point	p3	done applying paint
Time Point	p4	done cleaning brush
Time Point	p5	done sanding
Time Point	p6	done mixing paint
Time Point	p7	done applying paint
Time Point	p8	done cleaning brush
Time Point	p9	done sanding

Figure 67. PSL Time Point Creation

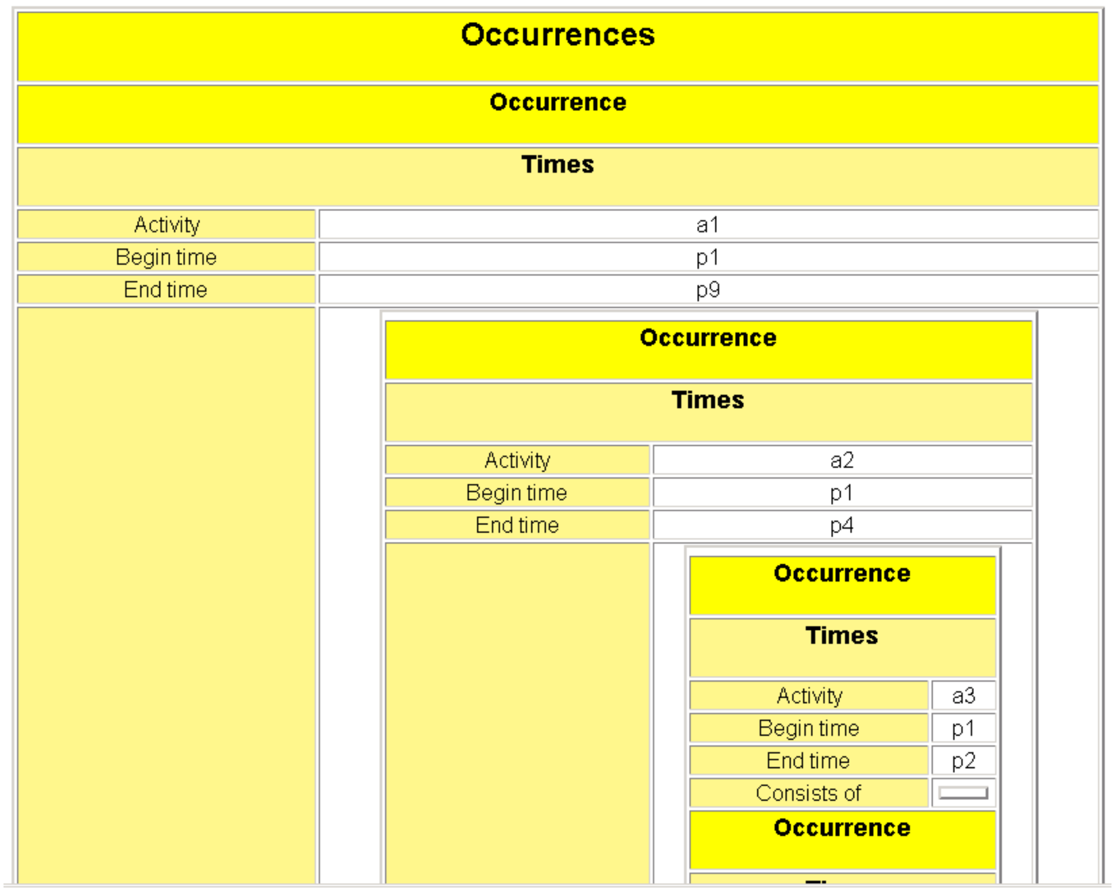
After time points are created, ‘activities’ are specified; this is shown below, Figure 68 :-



Activity Specifications				
Activity	a1			
Name	Finish product			
Consists of	**Sub Activity**	a2		
	Name	Paint		
	Consists of	**Sub Activity**	a3	
		Name	Mix paint	
		Requires	#Paint	
		Requires	#PaintMixer	
		Sub Activity	a4	
		Name	Apply paint	
		Requires	#Paint	
		Requires	#PaintBrush	
		Sub Activity	a5	
		Name	Clean brush	
		Requires	#PaintBrush	
		Requires	#PaintThinner	
	Sub Activity	a6		
	Name	Sand		

Figure 68. PSL Activity Specifications

The activities are then assigned occurrence times that allow each activity to be related in a sequence of sequences. Figure 69 demonstrates this :-



Occurrences

Occurrence

Times

Activity	a1
Begin time	p1
End time	p9

Occurrence

Times

Activity	a2
Begin time	p1
End time	p4

Occurrence

Times

Activity	a3
Begin time	p1
End time	p2
Consists of	

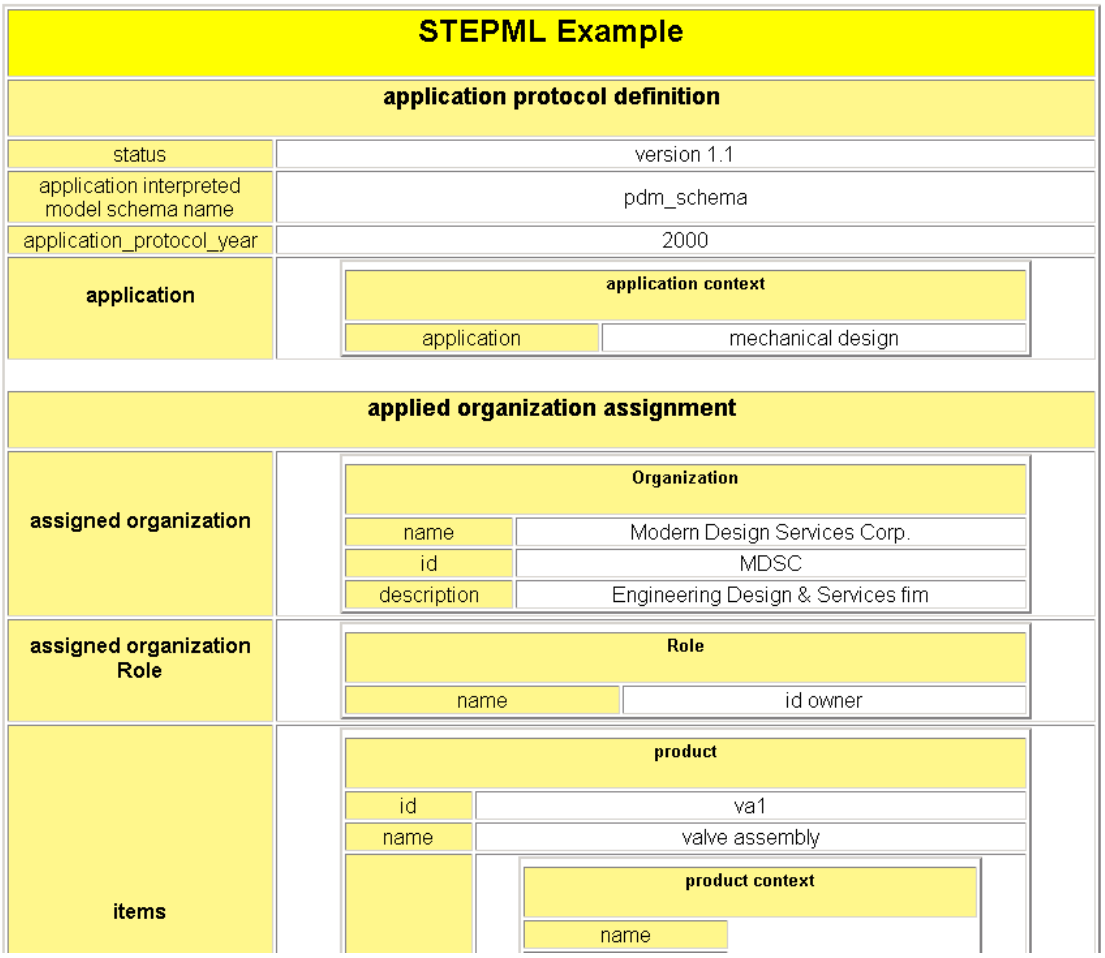
Occurrence

Figure 69. Activity Occurrence Times

9.3 STEPml

The next figure[41] shows a STEPml example, Figure 70 :-

[41] Hale, P. (2009) *STEPml* [online]. Available from: http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/STEPml.htm. [Accessed 25 July 2011].



STEPML Example

application protocol definition

status	version 1.1
application interpreted model schema name	pdm_schema
application_protocol_year	2000
application	**application context** application: mechanical design

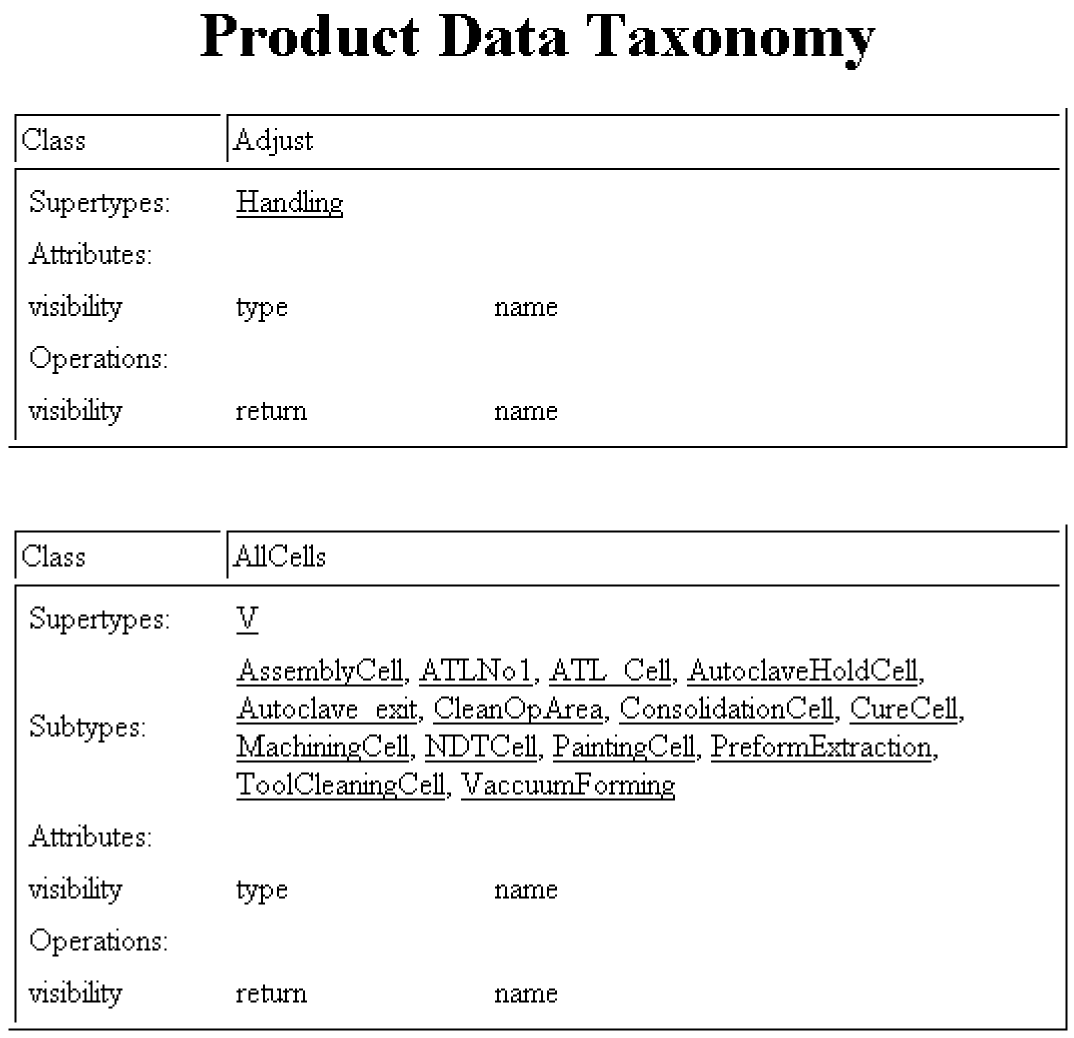
applied organization assignment

assigned organization	**Organization** name: Modern Design Services Corp. id: MDSC description: Engineering Design & Services fim
assigned organization Role	**Role** name: id owner
items	**product** id: va1 name: valve assembly **product context** name

Figure 70. STEPml example rendered with stylesheet

9.4 XMI (XML Metadata Interchange)

The XML Metadata Interchange specification provides a standard format for describing UML (Unified Modeling Language) model elements. Most UML tools contain functionality for exporting data in the XMI format. Figure 71 shows part of a UML design translated into XMI. A stylesheet is used to display the XMI on the Web in a readable form. A user can click on an associated term to view that term.



Product Data Taxonomy

Class	Adjust	
Supertypes:	Handling	
Attributes:		
visibility	type	name
Operations:		
visibility	return	name

Class	AllCells	
Supertypes:	V	
Subtypes:	AssemblyCell, ATLNo1, ATL_Cell, AutoclaveHoldCell, Autoclave_exit, CleanOpArea, ConsolidationCell, CureCell, MachiningCell, NDTCell, PaintingCell, PreformExtraction, ToolCleaningCell, VaccuumForming	
Attributes:		
visibility	type	name
Operations:		
visibility	return	name

Figure 71. XMI Taxonomy - Product Data Structure - Visualised with Stylesheet

9.5 Yahoo Pipes

Figure 72 below shows an example of a Yahoo Pipe created to test Yahoo Pipes as a diagrammatic end-user programming tool. This did prove to be a useful and flexible diagrammatic programming tool to use, with the advantage of making use of Semantic Web languages without requiring understanding of their syntax.

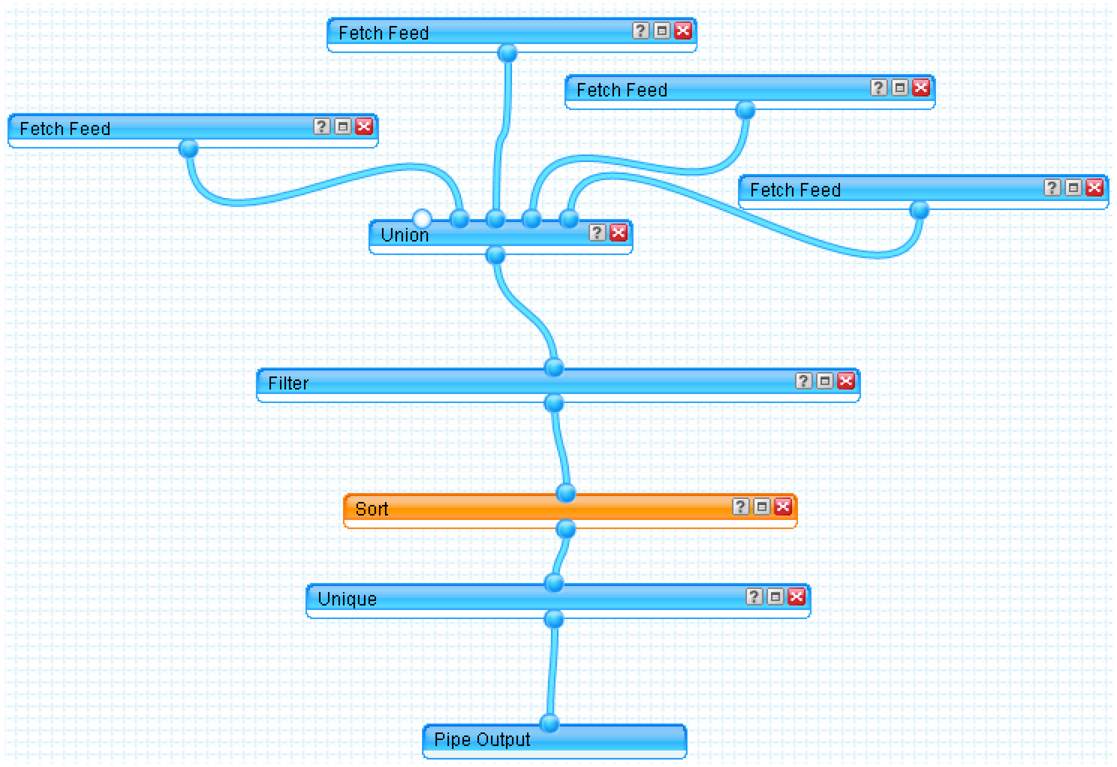


Figure 72. Example Yahoo Pipe

This example is available at
http://pipes.yahoo.com/pipes/pipe.info?_id=1717f42cc067a80511c255e93deca0dc :-
And all the examples created can be accessed at :-
http://pipes.yahoo.com/pipes/person.info?display=pipes&eyuid=nyyrkD0jrXf16Jis1uKBZs5Yf3u7AjvpKQ--.

9.6 Online Examples

The table shows the examples created and put online, either by translation through the ontology and modelling system created for this thesis, or created independently to test and demonstrate ideas, technologies, and/or standards. An online interactive version of this table and the Language and Tool Mapping table from chapter 3 is available at
http://www.cems.uwe.ac.uk/~phale/#LanguageToolMapping.

The categories of 'Designed/Implemented/Translated from Ontology', 'Coded/Adapted', and 'Used' indicate the authors experimentation with taking the role of 'System Developer', 'Model Builder' and 'Model User' respectively. The top headings indicate the type of system developed to aid End-User Modelling. It was necessary to combine research and implementation in all these areas to make progress in this thesis.

Table 5. Online Examples by Categories

	Taxonomy Visualisers	**Diagrammatic Editors**	**Models**	**Semantic Web/other standards Representations**
Designed/Implemented/Translated from Ontology	Taxonomy View of Wingbox Model http://www.cems.uwe.ac.uk/~phale/Flash/FlashHCI.htm Java Applet Spar taxonomy http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/JavaTree/AutomaticaOutputSpar/classes/TreeOutput.html XML Spar Taxonomy http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/SparMenu.xml	CAD Diagrammatic View of Wingbox Model http://www.cems.uwe.ac.uk/~phale/InteractiveSVGExamples.htm	Spar Model http://wiki.vanguardsw.com/bin/browse.dsb?det/Engineering/Aerospace/Wing%20Spar%20Translated%20from%20Protege%20Taxonomy	XML Spar Taxonomy http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/SparMenu.xml XML Colour Coded Wing Taxonomy http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/WingMap/Wing.xml XML Taxonomy View of Wingbox Model http://www.cems.uwe.ac.uk/~phale/Flash/FlashHCI.htm SVG - CAD Diagrammatic View of Wingbox Model

				http://www.cems.uwe.ac.uk/~phale/InteractiveSVGExamples.htm
Coded/Adapted	Drag and Drop http://www.cems.uwe.ac.uk/amrc/seeds/Ajax/components.html XML Colour Coded Wing Taxonomy http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/WingMap/Wing.xml	Interactive Map http://www.cems.uwe.ac.uk/amrc/ajaxGoogleMapsMashup.html	Parametric Cost Models http://www.cems.uwe.ac.uk/~phale/ParametricModelExamples/engine.xml - Created with Christophe Bru	SVG State Transition Example http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/StateTransition/statetrans.htm SVG Graph Example http://www.cems.uwe.ac.uk/~phale/SVGExamples/powerpricegraph.htm XML XForms http://www.cems.uwe.ac.uk/amrc/seeds/FormFaces/Examples/WingBox/index.html Process Specification Language http://www.cems.uwe.ac.uk/~phale/XMLDemonstrat

				ors/psl.xml RDF - Ring Manufacturing Sequence http://www.cems.uwe.ac.uk/~phale/XMLDemonstrators/rdfring.xml STEPml http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/ProductData/PDM_BOM_v203.xml
Used		Yahoo Pipes http://pipes.yahoo.com/pipes/		