

User Driven Programming

Contents

| | |
|---|----|
| User Driven Programming..... | 1 |
| Contents..... | 1 |
| Abstract..... | 4 |
| Introduction..... | 5 |
| Chapter 1 - Explanation of the Problem to be addressed..... | 8 |
| Research Aim..... | 8 |
| Research Approach..... | 10 |
| End-user Programming and Engineering Modelling..... | 10 |
| Spreadsheet Use..... | 11 |
| Main Categories of User..... | 11 |
| Model Builders..... | 12 |
| Model Users..... | 12 |
| Research Approach..... | 12 |
| User Driven Modelling/Programming..... | 13 |
| Evaluation of Visualisation, Interaction, and Translation techniques..... | 15 |
| Chapter 2 - History of End-user Programming..... | 19 |
| 1960s..... | 19 |
| 1970s..... | 20 |
| 1980s..... | 21 |
| 1990s..... | 21 |
| 2000s..... | 22 |
| Conclusions from Research Timeline..... | 23 |
| Chapter 3 - ACCS spreadsheet an illustration of the problems to be addressed..... | 25 |
| Implementation Example - Airbus ACCS (Aerospace Composite Costing System)..... | 25 |
| Project Aim..... | 25 |
| Implementation..... | 26 |
| Problems with this approach..... | 27 |
| An Alternative, Collaborative End-User Modelling..... | 28 |
| Chapter 4 - Outlining a different approach of End-User Programming..... | 30 |
| End-User Programming..... | 30 |
| UML (Unified Modeling Language) and End-User Programming..... | 32 |
| Information Sharing using Web Technologies..... | 33 |
| User Driven Modelling..... | 34 |
| Modelling and Decision Support using Web Technologies..... | 35 |
| Management of Complexity..... | 36 |
| Chapter 5 - Structured representation of information..... | 39 |
| Structure and Representation of Information..... | 39 |
| Visualisation and Representation of Problems..... | 40 |
| Structured Languages and Translation - application to decision support..... | 40 |
| The Need for Ontologies to aid Modelling..... | 43 |
| Taxonomies, Ontologies and Structuring of Information..... | 44 |
| Top Down and Bottom up Ontology development..... | 45 |
| Top Down Example Rolls Royce CAPPe..... | 46 |
| Bottom up Example - Faculty Information System..... | 47 |
| Combined user Tagging and Ontology Development..... | 47 |
| Interoperability for User Driven Modelling and Programming..... | 47 |
| Structured Language, Layered Architecture, and Translation..... | 48 |
| Ontology Editors and Modelling Tools..... | 49 |
| Domain Representation and Ontology development tools..... | 49 |
| Ontology and Semantic Tools and Translation for Modelling/Programming..... | 50 |
| Chapter 6 - Meta-Languages and their usefulness for User Driven Programming..... | 51 |
| Semantic Web and Semantic Grid Research - Application to Meta-Programming..... | 52 |

| | |
|--|-----|
| Semantic and XML Standards | 52 |
| Generic Standards | 52 |
| Engineering Domain Specific Standards | 56 |
| XML for Visualisation and Interaction | 58 |
| SVG and VRML for Visualisation..... | 58 |
| Representation of Equations and Rules..... | 59 |
| Methodology..... | 59 |
| Chapter 7 - Early Examples - attempts to create re-usable modelling software..... | 60 |
| User Driven Programming Early Research..... | 60 |
| Ontology User Interface improvements..... | 67 |
| Chapter 8 - Use of Semantic Web techniques for Model Translation..... | 68 |
| Translation for De-abstraction | 68 |
| Translation for the Modelling Process..... | 68 |
| Model-Driven Programming..... | 69 |
| Translation Steps..... | 70 |
| Dynamic and Responsive Software | 73 |
| Chapter 9 - Enabling User Driven Model Development..... | 78 |
| Criteria necessary for User Driven Model Development..... | 78 |
| Application of Semantic Web and Meta-Programming to User Driven Modelling | 79 |
| Capturing Information | 80 |
| Final Research - Putting it all together..... | 82 |
| Chapter 10 - User Driven Modelling Techniques implemented with a simple example | 86 |
| Summary..... | 95 |
| Chapter 11 - User Driven Modelling Techniques implemented with a complex example | 96 |
| User Driven Programming Examples | 96 |
| Implementation | 96 |
| Translation Process | 96 |
| Implementation Example – Spar Hand Lay-Up Process..... | 98 |
| Web Output..... | 104 |
| Semantic Search..... | 110 |
| Chapter 12 - Further Research | 114 |
| Objectives for future development of Ontologies..... | 114 |
| Ontology Visualisation and Interaction | 115 |
| Modelling and Simulation Objectives..... | 115 |
| Visualisation and Interactivity | 118 |
| User Driven Modelling Solutions | 119 |
| Method..... | 119 |
| Research Connectivity | 120 |
| Usability Testing..... | 121 |
| Testing and Metrics | 122 |
| Chapter 13 - Findings and Conclusion..... | 123 |
| Findings from using the alternative approach..... | 123 |
| Conclusion | 124 |
| Web References | 126 |
| References..... | 134 |
| Appendix..... | 149 |
| Appendix..... | 149 |
| Open Standards..... | 149 |
| RDF..... | 149 |
| DAML/OIL..... | 150 |
| Process Specification Language - PSL | 152 |

Abstract

The problem examined in this thesis is that of allowing domain experts to create decision support software. The main emphasis is on engineers, who experience problems in creating and sharing their software. 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 cost in time and person effort of developing decision support systems, often involving requests for help from IT service providers can be too high. Instead ad hoc decisions are made with insufficient justification or records for use as lessons learned. The problem and possible solutions are illustrated using the example of modelling the cost of wing components. The objectives of the research are to provide the foundation for a generic solution to these and other end-user programming problems.

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. The aim 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 more generic way. The possibilities for a generic user-driven programming 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. All that is necessary is to link the information visually via equations, and store these 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 programming languages would assist greatly with interoperability as these languages are standardised for use in a wide range of computer systems.

The metaphor behind the provision of this end-user programming environment is that of visual representation of interlinked information snippets. These snippets will be visualised as nodes or translated to other views. The nodes can be linked via equations. An example of this is an engineering component, which can be viewed as interconnected nodes of information or as a diagram. The same information can be viewed and translated both ways, from nodes to diagram or diagram to nodes. The information can be further translated into computer languages to make use of compilers and interpreters that can run models that perform calculations. This research also examines an approach of collaborative end-user programming by domain experts. The end-user programmers will be enabled to use a visual interface where the visualisation of the software matches the structure of the software itself, making translation between user and computer, and vice versa, much more practical. This makes it possible to model problems by creating items and equations that link them.

Introduction

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. Sternemann and Zelm (1999) explained that it has become necessary to research collaborative modelling and visualisation tools, because of the business trend towards global markets and decentralised organisation structures. Work on this is explained by Green et al (2007). To achieve this, Semantic Web tools will be used that represent the information to be shared in an open standard way. Semantic Web/Web 2.0 approaches are becoming increasingly important in providing collaboration and interactivity, A Joint Information Systems Committee (JISC) report by Anderson (JISC, 2007) talks of 'harnessing collective intelligence' by means of interactive collaborative software, the author calls this 'distributed human intelligence'. Cheung et al (2007) explain the necessity for collaboration tools to support early stage product development within networked enterprises.

The aim is to ensure ease of development and use of a software system (explained in chapters 10 and 11) by using applications that operate at one or more levels in a conceptual hierarchy, while still being able to communicate with the layers above and below in the hierarchy, and with other applications. McGuinness (2003) writes about how ease of use via conceptual modelling support and graphical browsing tools is essential if systems such as that created in this PhD research are to be usable in the mainstream. To facilitate this, open standard tools will be used to provide communication within the system. A communication mechanism should be invisible to the end-user who cannot be expected to consider such matters. This communication would involve large amounts of related information being translated and passed on in its entirety rather than just individual objects or messages. The intention for this main prototype is to facilitate communication between software applications, and between people and the applications and so make it easier for engineers and others to collaborate and co-ordinate their product design and manufacture.

This thesis outlines the technique of User Driven Modelling (UDM). The idea behind this is that software users (in this case engineers), can create models that perform and visualise calculations (cost of manufacture and the reasons behind this cost). The advantage of this is that the engineers can share and adjust models without needing to call upon a software developer to create the model. The time saved can give engineers the chance to cost designs early. This could allow the design to be changed before most of the future costs are incurred. The thesis explains how the above aims can be achieved, in order to enable decision support during product development, whilst minimising dependence on specialist software and detailed programming effort. The basis of this is an ontology that can be visualised and edited in tree form. This involves making it easier for those who are not professional programmers to instruct computers. Visualisation of information, and allowing people to interact with

this visualisation directly is particularly important. This can allow the person to instruct the computer to perform calculations, and see the result without needing to write code.

This thesis is based on research undertaken to establish a way of representing information relating to the design and production of a wing box (the main structure of a wing), and providing a cost modelling capability. The argument presented is that it is possible to use an approach where such a system could be created as generic, and thus re-usable for other modelling problems, and could be built and maintained much more efficiently than current techniques allow. Models can be created by users, who can also be termed model builders, without the need for them to write code. Model builders can adapt the models for their individual modelling purpose. Varian (1997) explains how people can create economic models; such models could be created using the system provided in this thesis. Miller and Baramidze (2005) 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.

The approach of developing decision support models for design and costing using a spreadsheet or standalone program is compared and contrasted with the alternative approach of using open standards ontologies and software. It is argued that the second approach makes User Driven Modelling (UDM) possible and that this can enable more effective development of models. Users can create their own programs using this technique.

Chapters 1 to 4 illustrate the problem to be tackled. Chapter 1 explains the problems in software development that lead to the conclusion that a different approach is necessary. Chapter 2 covers the history of end-user programming, explaining research of others from the 1960s to present day. This historical view is necessary in order to explain what problems need to be solved to enable end-user programmers such as engineers to create software for the problems they are investigating. Chapter 3 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. Chapter 4 explains why a new approach is needed, and begins to define possible solutions to end-user modelling problems.

Chapters 5 to 7 outline the research of others and combines this with explanations of early attempts to apply this to the thesis. These chapters examine the alternative approach and justify this as solving the modelling problem more effectively. Chapter 7 outlines the first attempts at applying this research that eventually led to the implementations explained in chapters 10 and 11. The approach used involves creating structured taxonomies that are part of an overall ontology of information relating to aerospace, and the automated generation of software to access and process this information. This approach could also be used for other types of modelling problem.

Chapters 8-11 explain how the research was implemented in the final example systems. Chapters 8 to 9 explain the alternative approach of User Driven Modelling (UDM), and explain how this approach would be used and tested. User Driven Modelling, which can also be called User Driven Programming (UDP), is an end-user development technique. In chapters 8 and 9 this field of research is explored, and the theory behind the later research, and implementation of examples to demonstrate this approach, is

explained. Chapter 10 introduces the methodology used with a simple example to explain how this research is applied. This is in order to make the explanation of the research that led to this implementation more understandable. It is explained that User Driven Modelling, which is part of the User Driven Programming (UDP) approach, is thus a useful development of previous end-user development techniques.

The final part explains how the implementations can solve the problem of end-user programming specified at the beginning of the thesis. Chapter 11 explains the methodology with a real and complex example. Chapters 12 and 13 explain how this alternative approach could be applied to a wide range of other problems, and how the methodology outlined of User Driven Programming could be expanded to more general programming.

Chapter 1 - Explanation of the Problem to be addressed

Research Aim

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. 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, so the shaded area is examined.



Figure 1 - Research Area

Within this thesis the terms 'user', and 'domain expert' are used interchangeably. 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. Clarke (2007) examines the characteristics of end-user developers in order to assist with meeting their needs. The users/domain experts may well be computer literate and able to model certain problems using a software tool such as a spreadsheet. Crapo et al (2002) cite (Savage, 1996) in stating that "Every one of the perhaps 30 million users of spreadsheet software can be considered a potential modeler". 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". This means understandable visualisation is essential to the tasks of modellers, otherwise they will lose track of information. The reasons that spreadsheets are inadequate for representing complex models are connected with difficulties in maintaining, extending, and reusing spreadsheet models. So to be able to model a complex problem, the users/domain experts currently must specify their requirements to other software experts, who may not have domain knowledge themselves. Shim et al (2006) explain how modern decision support systems need to support teams "DSS once supported individual decision-makers, but later DSS technologies were applied to workgroups or teams, especially virtual teams. The advent of the Web has enabled inter-organizational decision support systems, and has given rise to numerous new applications of existing technology as well as many new decision support technologies themselves." Sometimes these teams can span more than one company. It is difficult to find and employ those who have sufficient expertise in both the software and the domain. Someone without the domain knowledge may not understand the requirements. Project teams will find it hard to model a problem such as a new design unless all team members can access relevant software, to understand the problem to be modelled and solved (Rodgers et al, 2001). Organizations are often geographically dispersed and act as virtual teams, (Fensel et al, 2001) and (Camarinha-Matos et al, 2001) make this point. An additional problem 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" (Willemain and Powell, 2006). So it is important to make technologies accessible for modellers to use, and to enable models to be shared and navigated, so that expert modellers can assist novices to create better models.

Software development is time consuming and error prone, partly because of the need to concentrate so much on the implementation rather than the problem to be modelled. If people could instruct a computer without this requirement they could concentrate their effort on the problem to be solved. 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. This means they could 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. This personal programming alternative is called User Driven Programming (UDP) within this thesis, and for the examples demonstrated the term User Driven Modelling (UDM) is used to explain the application of User Driven Programming to model development.

Although other researchers have prototyped Semantic Web languages such as RDF (Resource Description Framework) (World Wide Web Consortium, 2006f) and XML (eXtensible Markup 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 can provide an interactive programming environment for computer literate non-programmers to manipulate information and construct models.

Objectives

Research Approach

For this thesis, 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 is explained in chapter 8). The Semantic Web was defined by Berners-Lee (2001) as "a web of data that can be processed directly or indirectly by machines". The thesis research has examined ways of structuring information, processing and searching this information to provide a modelling capability. Research by Aziz et al (2005) was investigated; this examines how open standards software can assist in an organisation's collaborative product development. Wang et al (2005) outlines an approach for integrating distributed relational database systems and Tao et al (2004) developed a model for sharing information between enterprises using XML. The automated production of software containing recursive Structured Query Language (SQL) queries enables this, which is illustrated in chapter 11. The approach is a type of high level Meta-programming. Meta-programming and structured language is explained by Dmitriev (2006) and Mens et al (2002), and in chapter 6. The approach is intended to solve the problems of cost and time over-run, and failure to achieve objectives, that are the common malaise of software development projects, and many projects that use software solutions. Grant and Ngwenyama (2003) explain these problems, which are common organizational and technical issues. The problems examined in this thesis are lack of shared understanding, lack of communication and co-ordination among departments, inconsistencies in documentation, and duplication of effort. The creation of a web-based visual representation of the information will allow people to examine and agree on information structures, a use for taxonomies and ontologies explained by McGuinness (2003).

This alternative approach aims to develop an integrated decision support system where maintenance and extension of information can be undertaken by model builder(s) via visual editing of library taxonomies. To facilitate this, information is visualised clearly in a structured understandable way, and which can be shared via web-based visualisation. This can be achieved by use of open standard languages for distribution of information over networks and between applications.

End-user Programming and Engineering Modelling

Many large companies, such as Rolls-Royce and Airbus, 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). 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. Chapter 3 illustrates the problem with an example. Information was used from a project to develop a large spreadsheet for Airbus. The rest of this thesis illustrates the alternative approach of User Driven Modelling; though the spreadsheet is still used, it is not

maintainable in the long run. An important part of work for the DATUM (Design Analysis Tool for Unit-cost Modelling) project at Rolls-Royce was solving this problem of allowing people to create, maintain and reuse software more easily.

Spreadsheet Use

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), Paine, (2003), and Scanlan et al (2006) examine this problem. Also, the spreadsheet may add to the problem, by hiding the detail behind an elaborate and visually attractive 'front end', which does not clearly relate to the spreadsheet content. If the author of such an application leaves the organization, it is commonly abandoned as colleagues are reluctant to master its complexity, and often refuse to take ownership of it. Paine states 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. In 1997 Panko reported 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). "Given the billions of spreadsheets in use, this leaves the worlds of business and finance horribly vulnerable to programming mistakes" (Scanlan et al, 2006). 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. Aragonés 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". 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, he 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". 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. 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.

Main Categories of User

It is important to distinguish between two main types of users who are the target beneficiaries of the research as they would work on different parts of any overall system created.

Model Builders

Model builders 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. Expert modellers aim to build models, whereas novice modellers are more inclined to search for a specific numerical answer (Willemain and Powell, 2006).

Model Users

Model users make decisions based on their domain knowledge. This type of user manipulates the tree 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'.

Research Approach

The research aim is to create software that enables people to program using visual metaphors. Users enter information in a diagram, which for these examples is tree based. Tree based visualisation is often a good way of representing information structures and/or program code structures as trees allow for navigation along branches, can be converted to a graph representation, and translate well to data structures that are useful in programming. Ing-Xiang et al (2005) demonstrate this with their ISWIVE (Integrated Semantic Web Interactive Visualization Environment) visualisation environment for RDF (Resource Description Framework), RDF is used and explained throughout this thesis. Sometimes other types of network based visualisations are used, and conversion of these visualisations into other diagrammatic forms, including CAD (Computer Aided Design) style representations and charts, is also examined. A tree representation and graph representation can be used as required, and both are used in Semantic Web research, which is a major influence on this thesis. This relating of information is explained as a 'web' in (Berners-Lee and Fischetti, 1999), which also explains 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. The software developed as part of this research translates this human readable representation into computer languages. The visualisation also shows the flow of information. This is useful when at first only sketchy information is available, but later more details can be modelled. This is important in costing of new engineering components where at first parametric information only is available, but information can be added to the model as the design becomes more detailed.

A further benefit of encouraging end-user programming is saving the cost of delivery, distribution and support of software, by allowing end-user programming using web-based services. Courtot explains this in an article by Ward (2007) "You cannot keep on developing software the old ways. The costs of distribution and support are higher and higher and the customers are less and less satisfied. Instead of buying a licence for a program and developing applications themselves, companies will move in great numbers towards firms offering software as a service via the web browser. It's going to be much more visible than it has at any other time."

User Driven Modelling/Programming

A model-driven approach (more detail in chapter 8) is important for enabling of user-driven programming. User-Driven Model-Driven Programming is the way to make it possible for a much wider range of people to do their own programming. This combines the approaches of End-User Programming, the Semantic Web and Modelling. The mechanism for this is program transformation, translating from user view to model to code. Figure 2 shows this approach.

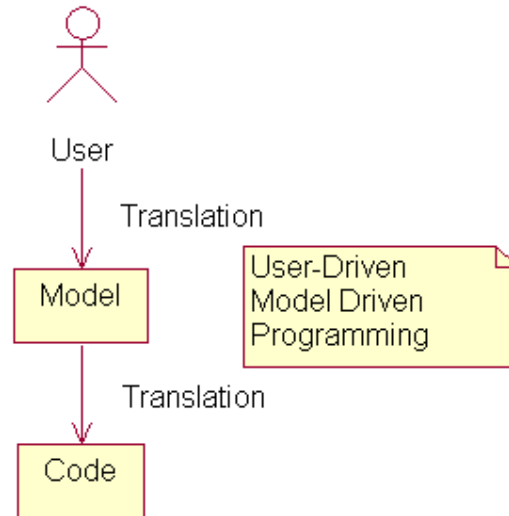


Figure 2 - Adding User Layer to Model-Driven Programming

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).
3. Visual Programming Extensions to spreadsheet type formulae based modelling, an example is (Vanguard System, 2006) enabled using approach 1.

This research links with the approach of enabling User Generated Content and providing a Visual Programming System. While the thesis concentrates mainly on 1 and 3, 2 is just as important and is referenced.

So there is considerable overlap between these three approaches and they must be integrated within interdisciplinary research to enable user driven programming. One approach to this is a Semantic User Interface; this is explained in Vernazza (2007). This can enable Drag and Drop programming that combines the benefits of all three research approaches. The 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. The next step is to make possible collaborative user-driven programming, by sharing the visualisation of models across computer networks and between collaborators. In order to make user driven modelling and programming possible, it is essential that a

communication mechanism is established, which allows users to generate changes and receive changes generated by the modelling system.

Types of Change

There are two types of change that can be applied to the model driven programming system, User Generated, and Model Generated.

User Generated

Figure 3 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 right. The tree is translated to structured text, and this is further translated to Code.

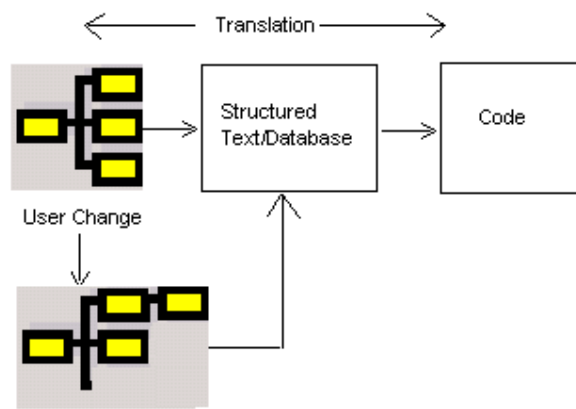


Figure 3 - User Generated Change

For the second user generated change shown in Figure 4, 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.

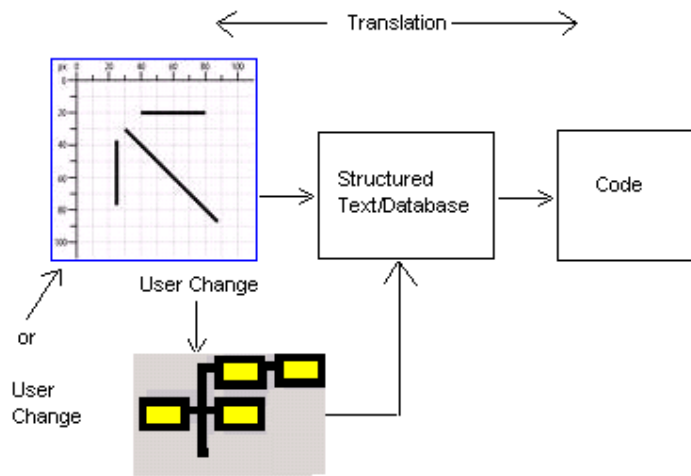


Figure 4 - User Generated Change, Alternative Interfaces

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 5.

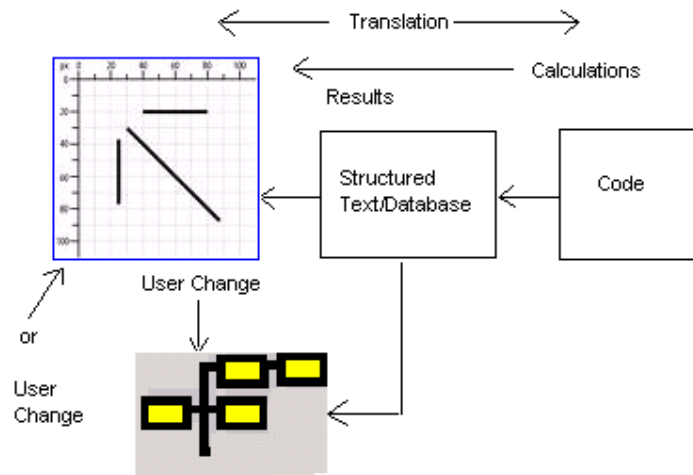


Figure 5 - Model Generated Change

Evaluation of Visualisation, Interaction, and Translation techniques

The intention is to enable non-programmers to create software from a user interface that allows them to model a particular problem or scenario. To achieve this it is important to research visualisation, and visualisation techniques to create a human computer interface that allows non experts to create software. This involves users entering information visually in the form of a diagram.

Begel (2007) explains that text based computer languages are often too obscure for end-user programmers, though Segal makes the exception of professionals who are used to programming. Fischer (2007) explains that it is the mismatches between end-users needs and software support that enables new understandings. Fischer also 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. He also argues for an approach to enabling end-user programming that makes it interesting to end-users. He explains that often the problem is that end-users find programming boring rather finding it hard.

The approach for this thesis is to develop ways of automatically translating information into program code in a variety of computer languages. The research is illustrated with frequent examples, most of these are visualised using a web interface, and provide a mechanism for using the technologies

discussed in the thesis to solve the problems raised. This includes examples which partially solved the problems but also failed in certain ways to tackle the issue needing to be solved. The experience from these attempts allowed for trying better solutions as the research continued.

This research is on translating from an abstract model of a problem expressed by a user, into software to solve the problem, and visualise the solution. This is very important and useful for many people who have insufficient time to learn programming languages. Scaffidi et al (2005) explain how much programming is undertaken by those who are not professional programmers. The open source Protégé ontology editor developed from a project of Stanford University is used in this thesis. The research links with a page on the Protégé Community Wiki (2006) to describe this.

The research demonstrates how a taxonomy can be used as the information source, from which it is possible to automatically produce software. Taxonomies are explained by Sampson (2000), who applies them to the problem of natural language processing, which is investigated by (Bernstein et al, 2006) and (Hudak et al, 2007). This technique is most suitable at present to modelling, visualisation, and searching for information. The thesis explains the technique of User Driven Model (UDM) development that could be part of a wider approach of User Driven Programming (UDP). This approach involves the creation of visual environments for software development, where modelling programs can be created without the requirement of the model developer to learn programming languages. The theory behind this approach is explained, and also the main practical work in creation of this system. The basis of this approach is modelling of the software to be produced in ontology management systems such as Jena (2006) and Carrol (2004), and Protégé (Stanford University, 2006b). It also has the potential to be computer language and system independent as one representation could be translated into many computer languages or Meta languages (Dmitriev, 2006), (explained in chapter 6).

Visual Interfaces

The development of visual user interfaces has been a major step forward. The use of pictorial metaphors such as folders to represent a collection of files has greatly aided human computer interaction. Pictorial metaphors give visual feedback so the user knows what the software system is doing. Crapo et al (2002) explain that visual representations are often processed by people pre-attentively, and immediately, so this gives an advantage over text. This technique can be used more dynamically in simulations. Simulations represent the real world problem and provide constant feedback to the user on how the system is progressing. In this sense, visualisation is a type of simulation as visual interfaces are perceived in a similar way to the real world (Crapo et al, 2002). Pictorial metaphors are static, while a user's mental model is made up of images connected together by a set of rules, as explained by Sasse (1997). The user runs a mental model like a simulation; this is explained by Crapo et al (2000) and (2002). Static user interfaces rely on a user to string together images into a mental model which correctly represents what the system is doing. A user may generate a mental model in response to user interface metaphors that is inconsistent with the system model. Begel (2007) argues that end-users may lack a mindset to form mental models of how to make the computer do what they want. Shim et al (2002) explain "the mental models of stakeholders with

various perspectives lie at the heart of the decision process, from defining what is a problem, to analysis of the results of trying to solve the problem." Simulation can help to ensure that the designer's model, system model and user's model are all the same. Ernst et al (2003) and Auer et al (2006) also discuss mapping between user's 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 (2000) and (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. This is also explained by Smith (1993) and (Cypher, 1993), which explain how the Pygmalion language attempts to bridge the gap between the programmer's mental model of a subject and what the computer can accept. The author of this system (Smith, 1977) went on to develop office oriented icons as part of the Xerox Star computer project.

Application of Open Standards to User Driven Modelling

The research applies this User Driven technique to aerospace engineering but it should be applicable to any subject. The basis of the research is the need to provide better ways for people to specify what they require from computer software using techniques they understand, instead of needing to take the intermediate steps of either learning a computer language(s), or explaining their requirements to a software expert. These intermediate steps are expensive in terms of time, cost, and level of misunderstanding. If users can communicate intentions directly to the computer, they can receive quick feedback and be able to adapt their techniques in a quick and agile way in response to the feedback.

A consideration which influences the work is that organizations should avoid proprietary or closed standards for their information. Systems should be designed with the assumption that information, which represents the primary system asset, may eventually need to be migrated to another software tool or environment. Open use of information was the priority for the DATUM (Design Analysis Tool for Unit-cost Modelling) where techniques in this thesis were tested. This project is explained by (Scanlan et al, 2006). Therefore a requirement of this research is that open standard semantic languages are used to represent information, and these languages should be used both as input and output for the models. These languages are based on eXtensible Markup Language (XML) (explained in chapters 5 and 6). XML can be used as a way to manage workflow within and between organizations (Ferreira and Ferreira, 2001), and for exchanging information between enterprises (Tao et al, 2004). Liu et al (2001) describe an XML enabled wrapper construction system. The wrappers convert HTML documents into XML, thus bringing it within reach of sophisticated query tools, and within a defined taxonomy. These same open standard languages can be used for developing the program code of models. The use of open standards is evaluated by Kelly et al (2007). For this thesis it is proposed that software and information represented by the software is separated but represented in the same open standard searchable way. Software and the information it manipulates are just information that has different uses, there is no reason why software must be represented differently from other information. So XML can be used both as the information input and output by the application, and for the definition of the model itself. The model can read or write information it represents, and the information can read from or write to the model. This recursion makes 'meta-programming' possible. Meta-programming is

covered in chapter 6. The methodology proposed is to provide a cascading series of layers that translate a relatively easy to use visual representation of a problem to be modelled into code that can be run by compilers and interpreters. This is to make it easier for computer literate non-programmers to specify instructions to a computer, without learning and writing code in computer languages. To achieve this, any layer of software or information must be able to read the code or the information represented in any other. Code and information are only separated out as a matter of design choice to aid human comprehension; they can be represented in the same way using the same kinds of open standard languages.

Meta-programming can also be used as a tool to produce software for linking ontologies, using Semantic Web, modelling, and visualisation techniques. The work involves enabling non-programmers to model complex problems visually and without having to use programming languages. Information is created in a visual tree using an ontology editor, the information is then transformed, and all calculations performed. Pipelines are important for translation and Meta Programming techniques used in this research as they apply one program to the results of another. XML Pipelines (XPL) (W3C, 2005) have allowed for repeated steps in transformation of a document, this can enable XML based meta programming without it being necessary to use lower level languages. Further transformations can be performed into other visual representations, any programming language or open standard information representation language, and this can be displayed on the web. This approach can be described as 'pipelining', which is explained by Gropp (2003) using the example of a project to convert Geography Markup Language (GML) to Scalable Vector Graphics (SVG). SVG is explained in chapters 5 and 6, and by McKeown and Grimson (2000). Pipelining is also core to XML (eXtensible Markup Language) and XForms technologies (Bruchez, 2006), which are explained in this thesis, particularly chapter 9. Pipelines are important for translation and Meta Programming techniques used in this research as they apply one program to the results of another. Also, transformations can be performed using SVG, between a tree/graph representation and other styles of representation e.g. an interactive CAD style representation. A major theme of the research is that of prototyping solutions to the problems raised using web and other software technologies. These are then referenced from the thesis document to illustrate the solutions discussed. The methods used for this representation and translation will be explained in the rest of this document.

Chapter 2 - History of End-user Programming

In this chapter a summary is given of end-user computing technologies from the 1960s to the present day, which influenced the work in this thesis on enabling end-user programming. This begins with the early interactive computing environments that put users rather than specialists in the 'driving seat' of computer software. The early work involved forerunners of PCs (personal computers) that gave users the opportunity to use computers for business, education, and at home. The chapter examines how each technology has assisted in moving towards a goal of making it possible for users to generate their own content. However there has been a fragmentation of research, gradually as computing has become more complex researchers and developers have become more specialised, leading to the need for research in the area shaded in Figure 1 (chapter 1). This thesis combines research into end-user programming, the Semantic Web and Modelling, and ensures that these research areas are enabled by a visual interface with the end-user.

The 1960s

In the 1960s Dartmouth BASIC (Beginner's All-purpose Symbolic Instruction Code) programming language Wikipedia (2007b) was designed and implemented at Dartmouth College by John Kemeny and Thomas Kurtz. Over time BASIC became a popular language for home and business use. It introduced many people to programming as a hobby or career. Recently, many variations of BASIC have appeared as programming, or macro, languages within applications. For example, Microsoft Word and Excel both come with a version of BASIC with which users can write programs to customize and automate these applications. Most versions of BASIC now include proprietary extensions; Microsoft Visual Basic adds object-based interaction to the standard BASIC.

Many of the modern concepts of computer graphics, dynamic objects and Object-Oriented programming were prototyped by Ivan Sutherland in 1963 in Sketchpad (Rauterberg, 2002). Sketchpad was the first program ever to utilize a complete graphical user interface. It helped change the way people interact with computers. Sketchpad is considered to be the ancestor of modern CAD (computer-aided drafting/design) programs as well as a major breakthrough in the development of computer graphics in general. Sutherland demonstrated with Sketchpad that computer graphics could be utilized for both artistic and technical purposes in addition to showing a novel method of human-computer interaction.

In the mid 1960s Seymour Papert, a mathematician who had been working with Jean Piaget in Geneva co-founded the MIT Artificial Intelligence Laboratory in the United States with Marvin Minsky. Papert worked with the team led by Wallace Feurzeig that created the first version of Logo (MIT Logo Foundation, 2006) in 1967. Logo is often used for teaching, and is an easier to use and understand dialect of Lisp (a language whose name derives from "List Processor"). The work of Papert is a major influence on research for this thesis because it demonstrates the approach of Constructionism (Papert and Harel, 1991) (Resnick, 1996). The Constructionism idea is based on the constructivist theories of Piaget. To this theory constructionism "adds the idea that people construct new knowledge with

particular effectiveness when they are engaged in constructing personally-meaningful products" (Resnick, 1996). 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." 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 the work of Kimberly (1995) where participants became part of the simulation they constructed in order to understand economic models. Constructionism theory can be applied to end-user programming and ontology modelling and building, which are explained in this thesis. To test these theories and demonstrate this approach interactive models were constructed, some of these are illustrated in this thesis. These examples add interactivity so that users can change the information visualised and see how the results change.

In the late sixties Alan Kay (Cypher, 1993), (Kay, 2003), (Kyoto Prize Laureates, 2004) used the term 'personal computer' and created a concept prototype, the FLEX Machine, he also envisaged a 'Dynabook' machine, the sketches for this look very similar to the laptop computers of recent years. Alan Kay and Seymour Papert envisioned in the 1960s the computer's role as a tool for the mind, an 'idea processor'. Kay and Papert have worked at bringing computers into this role for adults and children through the Logo language and environment by Papert and the open source Smalltalk language and environment, by Kay (Bergin and Gibson, 1996).

The Simula language (Wikipedia, 2007e) was developed by Ole-Johan Dahl and Kristen Nygaard and this included object-oriented concepts. Simula can therefore be considered the first object-oriented programming language, and a predecessor to Smalltalk, C++, Java, C#, and other object-oriented languages. Simula was designed for performing simulations, and the needs of that domain provided the framework for many of the features of object-oriented languages. Simula included objects, classes, garbage collection, and discrete event simulation.

Douglas Engelbert worked on a project to augment the human intellect. As part of the Augment (Stanford University, 2006a) project he demonstrated Hypertext and video conferencing. He is best known as the groundbreaking inventor of the mouse, windows, e-mail, and the word processor. Engelbart led one of the most important projects funded by ARPA (Advanced Research Projects Agency) in the 1960s: a networked environment designed to support collaborative interaction between people using computers. It was named the NLS (oNLine System). This prototype, developed at the Stanford Research Institute, influenced the development of the first personal computer and the graphical user interface at Xerox Palo Alto Research Center (PARC) in the early 1970s.

The 1970s

Alan Kay joined the Xerox PARC California in 1971 (Bellis, 2006), (Palo Alto, 2006), (Shim et al, 2002). Throughout the seventies the group at PARC led by Kay developed an integrated programming

language and environment called Smalltalk. Smalltalk was an early pioneer of object-oriented programming influenced by the first object-oriented programming language Simula. Alan Kay designed the system, which Dan Ingalls implemented. Through the seventies the group developed Smalltalk, which was one of the first systems to pioneer the WIMP (Windows, Icons, Menus and Pointers) interface. Squeak and Croquet have developed from the early work in Smalltalk, and provided a tool for end-user programming, collaboration, visualisation, and simulation. Open source and commercial Smalltalk programming environments have been developed, and Smalltalk is also used as a way of enabling constructionist learning and object-oriented programming.

In the early seventies the Alto personal computer was created at PARC. The Alto eventually featured the world's first What-You-See-Is-What-You-Get (WYSIWYG) editor, a commercial mouse for input, a graphical user interface (GUI), and bit-mapped display, offered menus and icons, and linked to a local area network. The Alto provided the foundation for Xerox's STAR 8010 Information System.

The development of the Altair computer in 1975, and the Apple II, helped create the personal computer revolution of the late seventies, and the eighties. There was still a need to find a common use for a personal computer that would increase the demand for it. In 1978, Harvard Business School student Daniel Bricklin had the idea for an interactive visible calculator. Bricklin and Bob Frankston then co-invented the software program VisiCalc (Power, 2006a), (Bricklin, 2006). VisiCalc was a spreadsheet, and the first 'killer' application for personal computers as this application provided a justification for using personal computers as a productive tool.

The 1980s

During the 1980s, ownership of personal computers became increasingly popular and many home users programmed using BASIC. There were many varieties of personal computers sold for home use. In the early eighties, IBM developed the first personal computer, built from off the shelf parts (called open architecture) (Bellis, 2006). This included a command line operating system written by Microsoft and their BASIC programming language.

Apple developed the Graphical User Interface (GUI) for the Lisa (Apple Lisa, 2006) that later became the Macintosh (Mac). The IBM style PC became most popular for business applications, while the Apple Mac was often used for Desktop publishing.

While at CERN (particle physics laboratory), Berners-Lee (2006) proposed a project based on the concept of hypertext, to facilitate sharing and updating of information among researchers. He then saw an opportunity to join hypertext with the Internet.

The 1990s

End-user Programming research has continued in techniques of Visual Programming (Dmoz, 2006) e.g. (Alice, 2006), Programming by Example (Cypher, 1993), (Guibert, 2004), (Smith, 1977) and (Smith, 1993), programming with automated assistance (Rich and Waters, 1990), and Natural Language

Programming (Bernstein et al, 2006). Squeak and Croquet (Croquet, 2006) have developed from the early work in Smalltalk.

Berners-Lee developed HyperText Markup Language (HTML), and has been involved with the World Wide Web Consortium (W3C), (2006b) in developing standards based languages for the Web. This has encouraged the growth of the Semantic Web (BBC News Technology, 2006) which allowed both humans and computers to search and interact with pages, and so encouraged the development of interactive web pages and communities.

The 2000s

Henry Lieberman (2000) has been involved with End-User Programming Research since the 1960s. He is author of 'Your Wish is My Command: Giving Users the Power to Instruct their Software'. The subject of this is giving users the power to create and modify their own programs. 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. Macías and Castells (2004a) and (2004b) use a Programming by Example approach for creation of model based user interfaces. Software agents are explained by (Gruber, 1993a), (Sycara, 1998), (Grove, 2000), (Hendler, 2001), (Shim et al, 2002), and (Bloodsworth and Greenwood, 2005). Shim et al also examine the role of web-based systems for modelling and decision support. They explain that the web dramatically increases usability, reduces training needs and allows for personalisation of web-based modelling systems in order to be better suited to each individual user.

Recent, present and future research can enable the use of Semantic Web technologies, (developed from HTML by Berners-Lee and others (Berners-Lee and Fischetti, 1999), to enable End-user Programming. This fusion of research and technologies is illustrated by (Lieberman, 2000) that contains has explanations of both areas of research, Semantic Web, and End-user Programming. Examples of this fusion include Protégé (Stanford University, 2006), (Jena, 2006), (TopBraid Composer, 2006), and (OpenCyc, 2006) and (Masters and Güngördü, 2003). (Isenhour et al, 2001) undertook a project to enable user interaction over the web. A related development is that of Web 2.0, explained by Anderson in (JISC, 2007), this report explains how the technologies used are enabling user-centred web applications, and the use of the web as a development platform. It explains "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. A cabinet office report for the UK Government (Mayo and Steinberg, 2007) recommends the Government makes use of User generated content to enable provision of information to achieve public policy objectives. Visual development environments based on AJAX (Asynchronous JavaScript And XML), (Bruchez, 2006) and (Cagle, 2006) aim to reproduce on the web the functionality provided by office tools such as Excel (often used as an End-user Programming Environment). During this thesis software of this type was created for a project involving a technical publisher. Information about Ajax and Web 2.0 is available on an Ajax/Web2.0 page (Hale, 2007a). Ajax and Web 2.0 are explained in more detail in chapter 6 under XML for Visualisation and Interaction. Ajax is part of what is generally called Web 2.0

because of the intention to provide much greater interactivity than previously found in web pages. The implications of Web 2.0 technologies for education are outlined in (JISC, 2007).

The Semantic Web and Web 2.0 techniques can be combined with programming by example research, and visual programming developed from the research of Sutherland (1963), Papert (1999), Kay (2003), and many others. This should enable users to experience a much more meaningful interaction with computers. Anderson's JISC report (JISC, 2007) 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. The 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.'

A trend has been towards partnerships that can further end-user programming by means of collaboration and sharing ideas. These collaborations include the End-users Shaping Effective Software (EUSES) (2006) research collaboration, Network of Excellence on End-user Development in Europe (EUD.Net) (2006), and the Institute for End-user Computing, (IEUC) (2007). Fabio Paternò investigated this subject as part of EUD.net (Paternò, 2005). This kind of collaborative research is important because end-user programming requires interdisciplinary research that includes co-ordinated research of the Semantic Web, Visualisation, Human Computer Interaction, and Collaboration. Much of user-driven modelling requires sharing of information in a way understandable to non programmers. Creation of the multi-levelled translation process between the user's view of the information and the representation required for computers requires a range of skills and knowledge.

Conclusions from Research Timeline

Two important points can be made from this research history:-

- Research that created prototype systems for specialist users, school children, and other researchers and programmers, but which made headway 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.

Lieberman (2007) 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." He 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.

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. Lieberman (2007) also argues that HCI

experts have concentrated on ease of use and should examine ease of programming. This needs to involve interdisciplinary research to combine different research approaches. Blackwell (2007) explains the need for interdisciplinary research on the end-user programming problem to identify software engineering techniques that can assist with this problem.

Chapter 3 - ACCS spreadsheet an illustration of the problems to be addressed

Implementation Example - Airbus ACCS (Aerospace Composite Costing System)

This example was chosen to illustrate problems that result from building a complex costing system in a spreadsheet, and the need for provision of alternative ways to represent such systems. An alternative system is illustrated in chapter 11, which was created with a user driven visual programming approach, and an application built on an ontology. This spreadsheet development approach is compared and contrasted in chapter 13 with the alternative approach developed in this thesis.

This chapter is based on work undertaken for Airbus to establish a way of representing information relating to the design and production of a wing box. 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. Later in this thesis 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 spreadsheet; the project is explained in (Scanlan et al, 2002).

Project Aim

The ACCS spreadsheet estimation tool was created for a project involving Airbus, to provide Airbus with a comparative cost for the manufacture of the various wing box parts using carbon-fibre composite. This example 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 wing. 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 costing of composites. (Scanlan et al, 2002) investigate parametric cost estimating, and generative cost estimating. (Brundwick, 1995), (Duverlie et al, 1999), and (Department of Defense, 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. A similar project was undertaken by Eaglesham (1998); this costed composite manufacture in aerospace.

So this 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 wing box, 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.

Implementation

The composite wing spreadsheet is a decision support tool for designers and manufacturers to evaluate the options for design and manufacture of composite wing box 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 wing box 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 6 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.

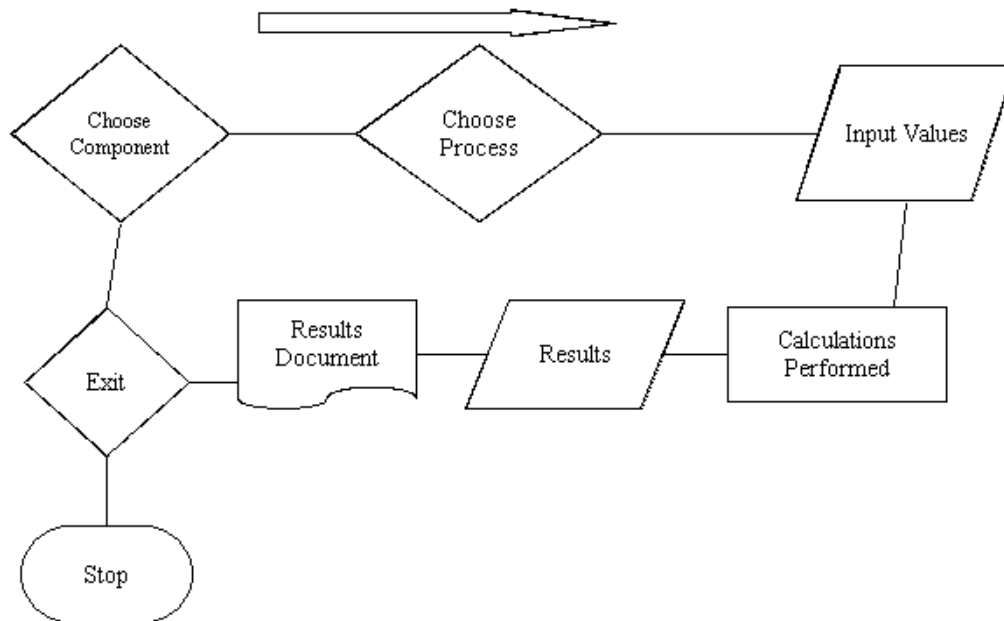


Figure 6 - ACCS Navigation

Visual Basic was used in the spreadsheet to provide navigation and constrain the navigation order to ensure decisions were made in the correct order. (Buehlman, et al, 2000) also used this approach for a decision support system, and its user interface. However problems were experienced within this wing box costing project, and with use of the Visual Basic interface that will emerge in the longer term. These problems relate to maintenance, extensibility, ease of use, and sharing of information. These

problems are explained in (Panko, 2000), (Rajalingham, 2001), (Paine, 2003), (Erwig et al, 2006), (Aragones et al, 2006), (Scanlan et al, 2006).

Although the beginnings of source to result structure translation used later in the thesis is present in this example, the input and outputs are structured only for visual understanding (e.g. by positioning within a sheet). This makes it difficult for humans or software to search the structure, as there is no formal representation, and therefore it creates maintenance and reuse problems.

Problems with this approach

Maintenance

Maintenance is a major problem. Although the software is popular and is still being used, there are several pitfalls that will eventually end its usefulness. Firstly although colour coding is used to indicate which fields should be edited, people inevitably make the mistake of editing other cells. If for example a user overrides a formula with a value, future cost calculations will be incorrect. In order to prevent this, code was written that protects the cells which should not be edited, if the user switches off this protection and tries to edit such a cell, this triggers the cell protection to be switched back on, and prevent this editing. In the system as a whole there is more code to deal with such situations than there is to provide navigation and calculation of cost. This means there are many more lines of code than a future maintainer might expect. Even so it is impossible to envisage and prevent everything a user might do that could corrupt the calculations as there are a wide range of menus and options available in spreadsheet tools. Therefore it would have been easier to begin developing software from scratch and add facilities than to use those provided by Excel and attempt to prevent their accidental misuse.

In order to provide all the options for costing of components and manufacturing processes the spreadsheet needed to be very large. There are many sheets and thousands of fields and formulae. The information in the spreadsheet is structured by means of grouping it on the appropriate sheet in tables, but this structure is hard to maintain. To ensure a valid calculation it is important to audit the spreadsheet to make sure all default values and formulae are reasonable, and correctly labelled. This is a major task and if this task is not undertaken often, errors will creep in. If the component design or manufacture changes, major changes would be needed to the spreadsheet in response, and these would be time consuming and difficult.

Extensibility

The maintenance problems explained above also impact on the extensibility of the spreadsheet. If this spreadsheet was to be extended or re-used to cost different components or processes, it would be very difficult to find all relevant values and formulae to change. It would probably be easier to begin development of a new piece of software.

Ease of Use

Feedback indicates the system is reasonably easy to use because the interaction required is limited to editing values and pressing buttons to go forward or back. Many people are familiar with this form of navigation from using web toolbars. However users know they are using a spreadsheet not a web page

and try to use it in generic ways that were not intended for this software. When they are prevented from exploring the alternative forms of navigation such as scrolling around a sheet or clicking on sheet tabs they may become confused.

Sharing of Information

It might become necessary to export the information from the spreadsheet to a process-planning tool. The lack of structure in the information makes it difficult to export it to other software systems. Kelly et al (2007) criticise the use of Excel for holding information as it is not an open standard, this leads to difficulties in archiving and reuse. Because the information is a flat structure, exporting it in a tree based (World Wide Web Consortium (W3C)) (2006b) standard such as XML (eXtensible Markup Language) (World Wide Web Consortium, 2006a) or RDF (Resource Description Framework) (World Wide Web Consortium, 2006f) would entail the development of more customised software. There are similar problems in importing information from other systems. For example if a component is part of another system there is no standard way to ensure this is specified in a machine readable way within spreadsheets, so searches cannot find this relationship or translate it to other systems. 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. 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. 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 alternative representations and visualisations. There is an alternative solution of research to automatically generating correct spreadsheets, and solve errors of meaning (semantic errors) (Erwig et al, 2006). This thesis concentrates on visualisation in order to make the meaning clearer to the human modellers.

An Alternative, Collaborative End-User Modelling

Knowledge sharing is essential for collaboration, especially in the early stages of design, (Cheung et al, 2007) explain how greater use of collaboration tools can be achieved. Merlo and Girard (2004) explain the necessity for collaborative information systems for designers and the need for an object-oriented approach to this. Dittrich (2007) argues that more research is needed into the software lifecycle and methods and tools needed for end-user developers, especially when they are collaborating. End-users often need to adjust old software for new purposes. Costabile and Piccinno (2007) also explain that new methodologies and environments are required for supporting end-user collaborative development. These problems can be solved by using the meta-programming and transformation approach, explained in chapter 1, by Fischer (2007), and in more detail throughout the rest of this thesis. This will be demonstrated using a simple example in chapter 10, and a complex example in chapter 11. Eaglesham (1998) examines how a decision support system could search existing information in order to create a new costing. This is a major undertaking and further research is needed into how such a system can

structure and search for what the manufacturing decision software requires. The existing data may be stored in various formats and it is difficult for an automated system to deduce meaning from data. The use of SQL (Structured Query Language) to illustrate search strategies was a useful step but a sophisticated taxonomy is needed as well in order to make a search strategy work.

The rest of this thesis explains how the approach of user driven modelling can be used to solve these problems. This begins with background explanations and research.

Chapter 4 - Outlining a different approach of End-User Programming

Translating concepts into an implementation is difficult. The difficulty of trying to explain the subjects of interest in a call for papers, or proposals, illustrates this problem. Because of the ambiguity of words, there is always going to be a problem of interpretation between those who specify the requirements, and those who need to understand and interpret them.

For software development, a good way to reduce the level of misunderstandings is to go through the loop from concept to design to implementation quickly and efficiently so that feedback can be returned from the software model. Then mistakes can be seen and corrected quickly. It becomes much easier to achieve this high speed development if the interface for development is made sufficiently easy to understand, so that a domain expert can use it to create the software, or at least a simple prototype that a developer can then work with and improve. Even if the aim of users is to specify requirements rather than create programs, creating working programs conveys requirements much better than any other form of requirement specification.

It is possible to work in reverse from implementation to design, or design to conceptual model. El-Ghalayini et al (2005) explains how ontologies can be mapped to conceptual models. This process can be eased if the same open standard software representations, languages, and structures are used throughout this process. This would be useful for checking that software is designed well, or re-using software designs.

The research for this thesis allows translation from a model-based representation of software to the actual software. This could involve automatically producing software for a Semantic Website from visual representations of the problem. The core of this modelling infrastructure would be automated generation of models written using World Wide Web Consortium (W3C) standards based languages and the visualisation of information represented in such W3C standard ways.

End-User Programming

Figure 7 shows that users are by far the largest group of program developers. So, there is huge untapped potential. As well as users there are two intermediate groups, and a small group of professional programmers.

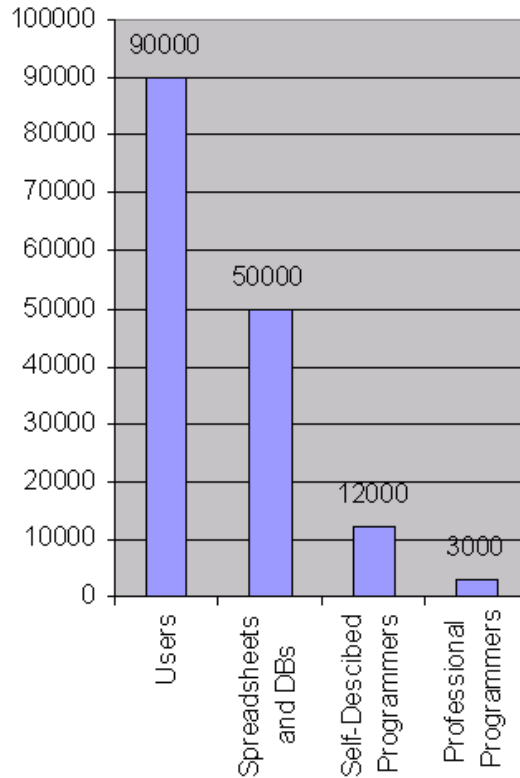


Figure 7 - Categories of User - United States at Work 2006

Source - Scaffidi et al (2005) based on data from US bureau of Labour Statistics.

These statistics were also explained and referred to in Myers et al (2006). In the Dagstuhl Seminar report (Burnett et al 2007) it is stated that "The number of end-users creating software is far larger than the number of professional programmers. These end-users are using various languages and programming systems to create software in tools such as spreadsheets, dynamic web applications, and scientific simulations. This software needs to be sufficiently dependable, but substantial evidence suggests that it is not." This point relates to that of (Ko, 2007) who explains that the goals of end-users may be unrelated to production of code, but instead they are interested in their domain problem, this means they perceive programming barriers as distractions. Ko explains that end-user programmers must be allowed to focus on their goals, and an important part of the solution is to visualise the whole program execution not just the output

Figure 8 illustrates the problem to be solved, this is the need to translate between human and computer. It also demonstrates the complexity of the problem that needs solving. This is the complexity that is present in most software systems. Complexity varies according to the needs of the application. The complexity of calculation increases when moving from the domains of knowledge management through to decision support and simulation. The increase in complexity is of calculation in this direction from knowledge management to decision support to simulation, and of information in the other direction from simulation to decision support to knowledge management.

End User Programming

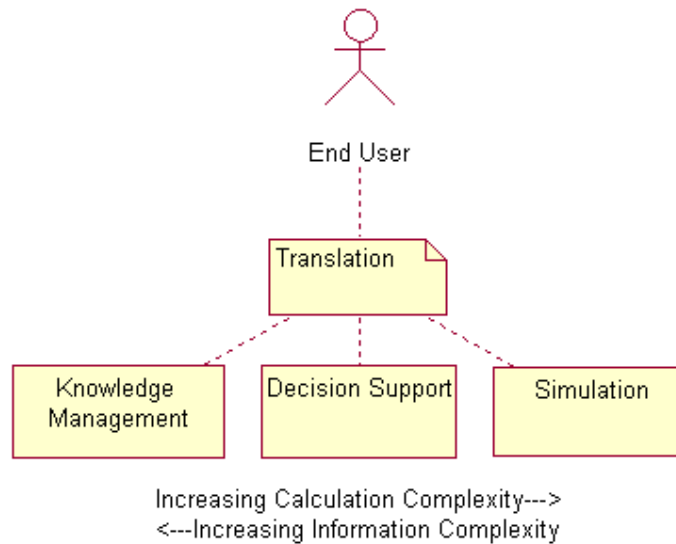


Figure 8 - End-user Programming Diagram

UML (Unified Modeling Language) and End-User Programming

This section examines the need for users to be enabled to program, and the tools that could be created to assist this. The way UML (Unified Modeling Language) (explained further in chapter 6) tools could be adapted to enable developers to provide an environment for users to program, and the methodology for development of new UML tools to help users to create software themselves are also examined.

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, using methods that are as close as possible to those which the domain expert normally uses. The proportion of domain experts in a particular domain (aerospace engineering) for example who can develop their own programs is fairly low, but the proportion that are computer literate in the everyday use of computers is much higher. If this computer literacy is harnessed to allow the domain experts to develop and share models, productivity for software development will be increased and the proportion of misunderstandings between domain experts and developers reduced. The domain experts can then explore a problem they are trying to solve and produce code to solve it. The role of developers would then become more that of a mentor and enabler rather than someone who has to translate all the ideas of experts into code themselves. Other developers may work at providing better translation software for the experts.

The role of UML in this research will be examined in chapter 6. However 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) makes the point that UML tools need to be extended to better enable modelling of collaborative tasks. Repenning (2007) explains the need for enhancements to UML to aid end-user programming. 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 assist software developers in creating a modelling environment suitable for domain experts to use to solve their problems. Achieving this 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.

Information Sharing using Web Technologies

Collaboration for end-user programming should involve Semantic Web modelling that would encourage collaborators who may have different backgrounds, or modelling different problems, to cooperate in the design of Semantic Web applications. Collaborations can consist both of research meetings and forums, and virtual collaboration through the web. This collaboration involves translations of the ideas of collaborators into a user interface designed to capture these ideas. The structure created via the user interface would then be converted via recursive translations of the ideas, through a layered architecture into computer code. Many of the Semantic Web tools to be discussed in chapter 5 allow collaborative authoring and sharing of information, and Auer et al (2006) discusses this issue and their OntoWiki collaborative Semantic Web application. Figure 9 sourced from (Sims, 2007) shows how research can be connected via a web or graph of links, which allow any person to follow a line of thought onwards to a detailed understanding. This sort of connectivity is important for linking of research, learning, subject areas, people, and software tools. This is the basis for the visual design, modelling, and end-user programming environment prototyped in this thesis. This kind of visual representation of connectivity is explained throughout the second half of the thesis, demonstrated in chapters 10 and 11, and further investigated in chapter 12 "Further Research" - Research Connectivity.

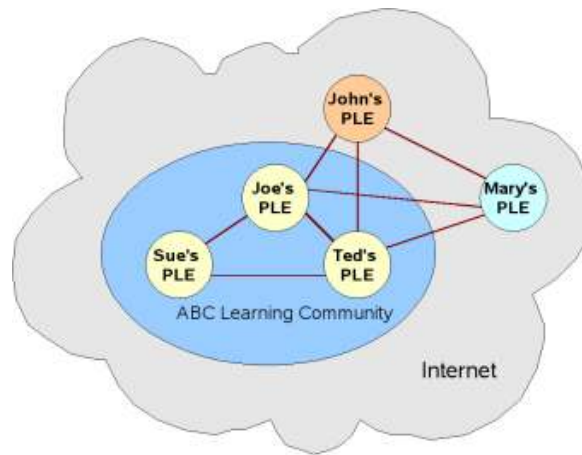


Figure 9 - Interconnection of Research and Communities

User Driven Modelling

The thesis researches provision of solutions for Knowledge Management, Decision support, and Simulation. The system provides automated translation from a model provided by the user, or by other systems into the software, ontology, and database representation. Miller and Baramidze (2005) explain 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." Any required calculations would then be made and translated to provide a model that can be interpreted by users. Johnson (2004) 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. The research for this thesis is intended to solve this problem 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. Nurminen et al (2003) evaluate a system called NICAD that used 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. As well as translating between the user and computer systems it is important to provide translations between different computer systems. Ciocoiu et al (2000) make the point that as it becomes necessary to translate between more systems the number of paths for the translation increases exponentially. To improve interoperability it is therefore necessary to provide either a translator or multiple translators, and the translators would be based on taxonomies or ontologies.

This thesis has involved 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, so it can be applied to work and home use for any task or subject area.

The use of functional language within nodes of a tree 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. Functional languages were explained in chapter 1. An additional advantage is displaying of expressions in the appropriate context, this is essential in order to allow end-user modelling and programming. Crapo et al (2002) 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.

Modelling and Decision Support using Web Technologies

There is much evidence that collaborative modelling using web-based techniques can be accomplished, and that it is important to achieve this. Shim et al (2002) explain the importance of the web for all types of decision support activity "At the beginning of the 21st century, the Web is the center of activity in developing DSS." Shim et al explain how decision support systems can be provided at low cost and for geographically dispersed companies, customers and suppliers. They cite Power (2006b) when stating "Web-based DSS have reduced technological barriers and made it easier and less costly to make decision-relevant information and model-driven DSS available to managers and staff users in geographically distributed locations." This was the reasoning behind the DATUM (Design Analysis Tool for Unit-cost Modelling) project (Scanlan et al, 2006). An open standard web driven method of collaboration is required to make it possible for organisations and individuals to become more deeply involved in projects that are coordinated using web technologies. Shim et al explain how the use of web technologies to standardise user interface design across different models can dramatically improve the ease of use of decision support software. This standardisation can also ease problems of installation and maintenance.

A common problem in industry is lack of communication between designers and production planners. Lack of communication between the different software tools used by designers and manufacturers has contributed to this problem. Much of the cost of a product is committed during the design process. Feng and Zhang (1999) and (2000), Marsh et al (2001) and (2002) and Scanlan et al (2006) explain this. Web-based collaborative modelling is becoming more important and easier to achieve as Semantic Web tools become available. Mecham (1999) examined early collaborative web-based tools for design and manufacture. Morris et al (2001) examine interactivity and collaboration on the web. Su and Amin (2001) and Jiang and Fukuda (2001) explain the internet technologies needed to enable web-based collaboration. Aziz et al (2005) examine how open standards software can assist in an organisation's collaborative product development. This approach is also outlined in Ciancarini et al (2001) that

explains ways of designing a document-centric coordination application over the Internet. Ciancarini et al explain that web documents can be generated on the fly. This can allow the user interface to respond dynamically to choices. Program code can be attached to the documents themselves, and code can activate certain behaviour based on the XML (eXtensible Markup Language) content of the document. Code can also be created separately and called on as a service when a document needs it. Nidamarthi et al (2001) explain how web-based collaboration can aid the design process. Cheng et al (2001) examine how web-based systems can assist a manufacturing firm to understand its supply chain. Huang and Mak (2001a) and (2001b), and Rodgers et al (2001) evaluate issues in the development and implementation of web applications for product design and manufacture. Mohamed and Celik (2002) created a web-based system for allowing building designers to analyse the affect of design alternatives on cost and scheduling. Reed et al (2000) show how web-based modelling and simulation can be used for improving the aircraft design process. Kim et al (2002) explain their approach to modelling and simulation and how a Web-based solution can be applied to distributed process planning. Web-based simulation is covered further in chapter 12. Zhang et al (2004) review Internet-based product information sharing and visualisation. Li (2004) and (2005) examines the role of web-based services for concurrent engineering and distributed process planning optimization. Gutowski (2001) examine web-based costing of composites. The above research reinforced the view that this is a sensible research approach. The next task is to further such research into visual end-user programming, so that it will become easier for those with limited software expertise to make use of the collaboration and modelling tools. Web-based systems for costing are also explained in Hale et al (2001), and user driven programming and web-based costing are examined in Hale et al (2002), and (2003).

The intention is to further the research of others into the approach of web-based collaboration, and use Semantic Web software and techniques to achieve this. The above research reinforced the view that this is a robust approach. Modelling collaborations based on these techniques would bring together experts in engineering, systems modelling, computing, and Human Computer Interaction.

Management of Complexity

Cagle (2006) explains that "You can never eliminate complexity from a system; you can only move it from place to place". Booch (2007) states "Systems are becoming more complex, and often try to overcome complexity with more complexity. A bad process itself can amplify complexity." He also explains that developers should produce simpler software that increases productivity. He recommends to "grow a system's architecture through the incremental and iterative release of testable executables". This is the approach used throughout the thesis, and for this purpose, software has been released via the web in order to receive feedback. The purpose of the research is to allow software developers to deal with complexity as far as possible in order to leave the end-user free to model their own problems. Ciocoiu et al (2000) explain that growing complexity in manufacturing hinders the ability to share information as the meaning is affected by its context.

Representing Complexity as Linked Trees

This thesis aims to adapt or create software systems to provide the visual editor for the source tree, and allow model builders to create a model by editing this. By doing so, the model builders would create a generic model for a particular modelling subject. Vanguard System is used to automatically convert the taxonomy into a decision support and modelling system. Vanguard has made their server available for collaborative model development (Vanguard Software, 2006). UK universities are already linking together to use Vanguard System on a network for aerospace and construction industry modelling. The model users can then use this decision support and modelling system to create their models. These models would be a more specific subset of the generic model, and could be applied for their own analysis. A translation mechanism was provided, to convert information or models into other representations (primarily web-based), and to visualise this information. It is important to be able to visualise the information in many different ways, as the information can be so complex that traditional visualisation techniques cannot show it due to the number of relationships exceeding the number of dimensions that can be shown. Examples in chapters 10 and 11 show how this problem is dealt with in this thesis. Bru et al (2002) and (2004) expand on this subject.

RDF (Resource Description Framework) allows for linking of people's information sources with each other, the structure for this is of interlinked graphs. "RDF is used to model knowledge, where tree-based representations are not enough" (Nilsson et al, 2002). Within the thesis a methodology for linking models using Protégé and then translating to Vanguard System was used and tested. The linking of models is essential in order to deal with the increased complexity of products and collaborative engineering that Cheung et al (2007) explain. The translations made to web standard languages such as RDF, in this thesis, have been successful but Semantic Web applications need more interaction and modelling capability, while Web 2.0 (JISC, 2007) technologies have good user interface and interaction capabilities but lack sufficient structure for complex calculations. Another issue of dealing with complexity is how much you use a top-down approach and how much a bottom-up approach in the collaboration necessary to define information sources. This will be covered in chapter 5.

A clear relationship was found within the thesis research between the end-user programming techniques that involve meta-programming and Semantic Web techniques that involve use of meta-information. Combining these approaches involves a meta modelling management approach that manages complexity on behalf of end-users. This approach deals with linking and using (reusing) information as required in order to solve problems a user is interested in. An implication of this research is that there is a strong need for user-friendly tools for non-programmers to edit meta-data and meta-programs that must be human understandable and machine understandable. "The Web is evolving from being primarily a place to find things to being a place to do things as well" (Uschold, 2003) citing (Smith, 2001). It is important to take advantage of this change to enable end-user programming. This allows model builders and users to seek agreement between individuals and between their models and software representations by intervening in the model building process. This is a practical solution that can be used until machine processable semantics might in future make more automated solutions possible.

Representing Complexity in Other Ways

Sometimes it is difficult to construct an ontology to represent a model, such as when a potential new design is to be modelled and little is yet known about it, so alternatives must be investigated. This kind of problem is investigated by the Institute for People-Centred Computation IP-CC (2006) collaboration network, where representatives from different universities and manufacturers can collaborate to investigate ways of discovering the information required. Also, techniques of Aspect Oriented Programming (Elrad et al, 2001a and 2001b) and (Murphy et al, 2001) can be used to identify functions that can be used in calculating results without the requirement of a detailed underlying taxonomy. Aspect oriented programming could be used to capture and translate user requirements especially where software functions can not 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 can then be translated into a computer language representation such as AspectJ for Java (Kiczales et al, 2001) and AspectXML for XML (eXtensible Markup Language) (Peterson, 2005). This allows for specification of a program as a model and translation to different languages, beginning with Java, and XML. AspectXML also makes use of XML as a programming language not just an information format. Collaborations involving Aspect Oriented Programming can be found at (Aosd.net, 2007) and (AspectXML, 2007). Techniques for representing and visualising uncertainty can also be used to help cope with the problem of a limited specification (Marsh et al, 2002), (Bru et al, 2004).

Chapter 5 - Structured representation of information

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 10; this process makes use of open standards languages, hence the interest in structured representation and open standards.

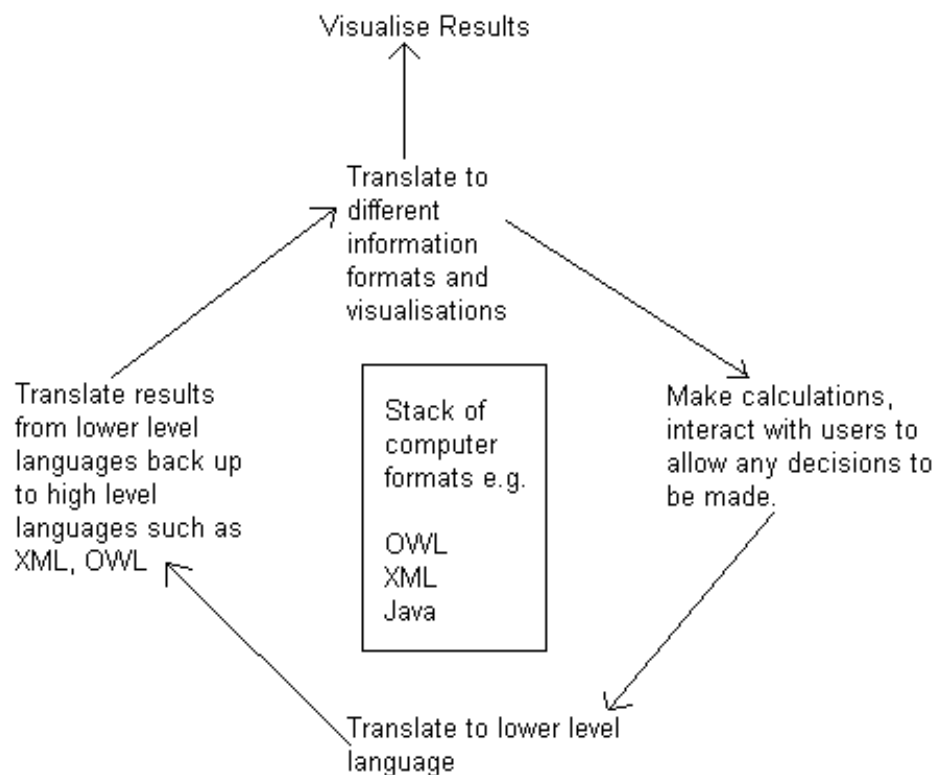


Figure 10 - Translation Process

This chapter and chapter 6 examine technologies that can be used for providing translation capabilities from a domain level representation of the problem to code. These technologies can also be used for separating information from program code so that users can edit the information directly via a visual interface, without having to edit code. The way these technologies were used in this thesis in order to achieve the above aims is described in chapter 9 and demonstrated in chapters 10 and 11.

Structure and Representation of Information

Busy users will not be interested in a system that is difficult to use, but the system must still give reasonable results. So it is necessary to make it as easy as possible for users to enter the information software needs. Structured languages can be the solution to this problem because they can be understood by a user, and the language is constructed using mathematical rules. Therefore the

structured language presents a mathematical representation to the computer and a natural language or diagrammatic representation to the user. Borthick et al (2001) explain however, that ambiguity in natural language can make it difficult to translate natural language into structured languages.

Visualisation and Representation of Problems

This research takes a problem, breaks it down into sub parts that can be represented by a number or equation, and then allows users to see and interact with the whole solution as applicable to each person's defined problem. Gruber (1993b) examines how equations and quantities can be represented in an ontology. Kamareddine et al (2004) examine how mathematics can be represented in order to make it easier for people to enter well-formed mathematical expressions. Examples are given to illustrate how this can be achieved in chapters 9, 10, and 11. Tools to allow visualisation and interaction with models have been investigated and/or created to achieve this. This approach has also been taken by Crapo et al (2002) who 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." Eng and Salustri (2006) discuss 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. So a task for this thesis is to enable the creation and sharing of these visualisations, in all ways that can aid in the understanding of the problems to be modelled.

Structured Languages and Translation - application to decision support

The open standard Stanford University (2006b) ontology management tool Protégé is used within this research. The ontology can be translated into a Decision Support tool Vanguard System (2006b), which runs the model. Software created for this research (demonstrated in chapters 10 and 11), using Vanguard System allows calculations of the cost of a design to be made, and provides a colour-coded representation. It is then possible to output this tree in as XML (eXtensible Markup Language) and schemas, web pages, interactive diagrams and code in programming languages such as the Java based costing tool Cost Estimator. It is possible to search the information both in Protégé and on the Web, as it is represented using searchable Semantic Web languages. Protégé is extensible with plug-ins that can allow extra visualisation and interaction capabilities, which are explained by Storey et al (2004).

When engineering organisations design and manufacture products they categorise this process into stages. From the early concept stage to the final design stage and manufacture, software is used to aid and record the process. It is common for different software to be used at different stages. If the software applications do not use open standards languages to communicate between these stages and between the various teams involved, information gets lost and not re-used as the chain of information

breaks. An open standard language is a language whose structure and syntax has been agreed by organisations such as World Wide Web consortium (W3C) (2006b), OASIS Organization for the Advancement of Structured Information Standards (2006), and National Institute of Standards and Technology (NIST) (2006d) (explained in chapter 6). The detail and accuracy of the information that can be provided to define the product varies along this 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 this analysis. Scanlan et al (2007) make this point; software created for this thesis was used in the DATUM (Design Analysis Tool for Unit-cost Modelling) project for a Rolls-Royce early stage costing project. Cheung et al (2007) make the point that 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. It is important that users who enter information about a design concept be guided by historical values where possible, and guidance information such as explanations, diagrams, and examples. It is essential that visual feedback is provided at every stage in the model development process in order to deal with this complex information. The use of statistical modelling is necessary to ensure a cost can be calculated at this early stage when there is a high level of uncertainty. Validated information for products can be held in a catalogue for case based reasoning.

Ontologies and Decision Support

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 Programming approach and its application to the problem above, is explained 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. The basis of the research for this thesis is an ontology that can be visualised and edited in tree form. Gruber (1993a) 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. 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. Fensel et al cite another Gruber paper where Gruber (1993b) defines the ontology as a formal explicit specification, and explains that formal means the ontology should be machine understandable. 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". McGuinness (2003) explains with the aid of a diagram the level of definition in ontologies, from purely human readable to machine readable. A similar diagram with more specific computing technologies is included in Uschold's presentation (2006) and his paper of 2003 which shows a continuum:-

Implicit - Informal (explicit) - Formal (for humans) - Formal (for machines)

Uschold 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." The research for this thesis has tended to move from left to right towards more formal representations, but only as necessary to ensure the modelling approach works. Horrocks (2002) explains that "An ontology typically consists of a hierarchical description of important concepts in a domain, along with descriptions of the properties of each concept. The degree of formality employed in capturing these descriptions can be quite variable, ranging from natural language to logical formalisms, but increased formality and regularity clearly facilitates machine understanding."

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

Results visualiser - inputs visualiser - calculation engine - ontology visualiser - ontology engine - database

The ontology created 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. 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. Miller and Baramidze (2005) advocate the use of interrelated ontologies rather than one monolithic ontology, this approach is used for this thesis. Logical constraints are informal and failures are caught by the modelling tool which 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.

It is possible to create an extra layer 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. Such systems can be made much more powerful if the information is codified into a structured database; Wang et al (2005) used this approach. Then On-Line Analytical Processing (OLAP) data mining can be used for more sophisticated data collection and analysis, OLAP is explained by Shim et al (2002). Lau et al (2001) explain how OLAP displays a multi-dimensional view of aggregated data, and presents a Rule-Based Analytical Processing

(RBOLAP) model which can be used for decision support. Shim et al demonstrate use of RBOLAP techniques using a case study of a mould and die information network.

Sutton (2001) illustrates how codifying 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. This highlights the difficulty of getting information into a knowledge base when it may be either only in individual's minds, or completely unstructured. Huber (2001) examines the issues in ensuring people transfer their knowledge and suggests the organisation's culture is important in peoples resistance to this. 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.

The Need for Ontologies to aid Modelling

Information is scattered within organisations and often not held in such a structured way as to be easily accessed by employees or software. This problem was 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 this. This structuring using an ontology could improve the accuracy of searches. XML may not be sufficient on its own for defining ontologies. The XML syntax defines relationships by their position within the text file. Thus XML syntax always implies a sequence whereas in reality the order of items may be unimportant, also there is no explicit way of representing associations between items, or differentiating between an 'Inheritance' (e.g. type of) and a 'Contains' (e.g. part of) relationship. XML schemas and DTDs (Document Type Definitions) can be helpful in defining these relationships, but there is then scope for differences in the way they are defined. RDF/XML has provided a layer of standardised semantics which overlays the basic XML. RDF is explained using a web page annotated with RDF (Hale, 2007i). XML and RDF are explained in more detail in chapter 6. McGuinness (2003) explains the role of markup languages in defining content to be machine readable. McGuinness cites a diagram from a presentation by Berners-Lee (2000) that contains representations of the place of each language in a stacked representation alongside the purpose of the language. This is shown in Figure 11 and explained further in chapter 6.

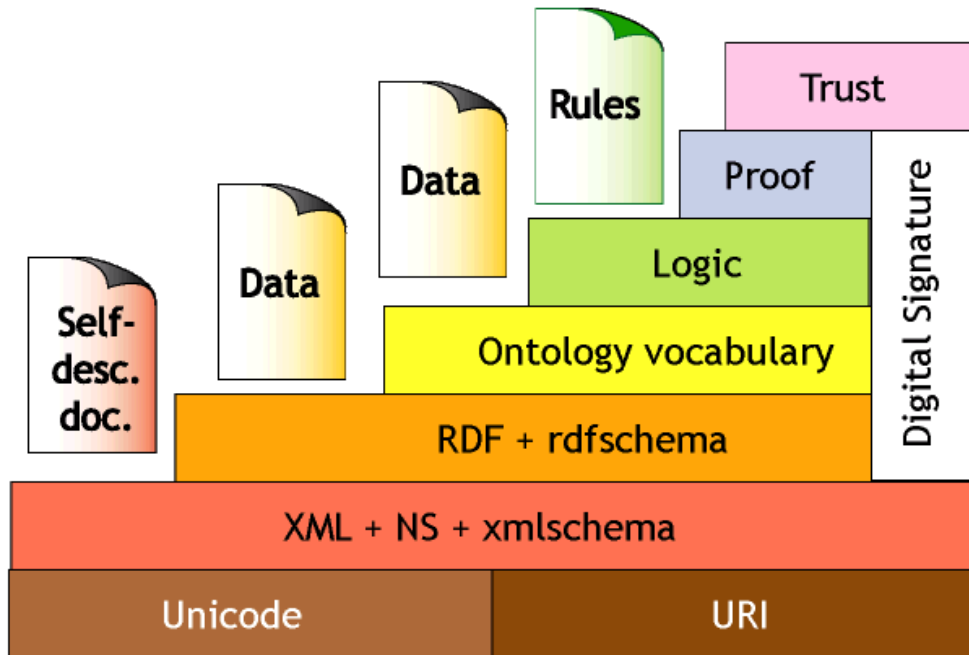


Figure 11 - Architecture, sourced from Berners-Lee (2000)

Taxonomies, Ontologies and Structuring of Information

An ontology is a classification structure. A taxonomy can be just a convenient structure to assist programmers, or part of an overall 'thesaurus' which describes and agrees the meaning of things. This thesaurus structure is the ontology and may contain one or more taxonomies. 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. 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. This should also 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 in (JISC, 2007). 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. Ontologies can also enable communication between computer systems and users (Garcia-Castro and Gomez-Perez, 2006). Hunter (2002) explains how taxonomies can be the basis of the definitions for an ontology, and that commercial software is available. Hunter gives examples of the Ministry of Defense technology taxonomy, and the Boeing online ontology. The taxonomy "Type-Of" and "Part-Of" relationships can indicate how to construct the taxonomy. Ciocoiu et al (2000) explain that taxonomies or ontologies of manufacturing concepts and terms can assist with avoiding misunderstandings and can aid communication. Veryard (2001) and McGuinness (2000) and (2003) provide useful guides on how ontologies can assist in linking

distributed data. This linking and connectivity is also explained by Uschold (2000) and Uschold and Gruninger (2004).

Knowledge based systems need to allow a variety of people in different disciplines to share knowledge across functional, departmental, and disciplinary boundaries. Consideration is needed of the further problem that certain knowledge should be shared with others outside the organisation such as suppliers, and customers.

There is a strong need for uniting of the approaches of top down ontology definition by a small group of experts with that of the bottom up approach of allowing all users to define the ontology. Anderson (2007) talks of the need for communities to build ontologies. Software applications are needed that allow users with little software knowledge to edit and update ontologies themselves. The extent to which an organisation allows this depends on its structure but if this is completely prevented or not enabled in the first place, there will be user dissatisfaction resulting from their lack of involvement. It is also likely that progress in defining and editing the ontology would be delayed.

The varied user base of knowledge systems results in a further problem, which is that of fragmentation of the language itself. As users are in different trades and professions they will not necessarily understand the same words, or assign to them the same meaning. Again this makes it necessary to structure the information in a knowledge-based system carefully, to ensure it can be well visualised, and agreements can be reached.

Relationships between terms such as 'type-of' and 'part-of' are as important as the term itself, as the relationship defines the meaning of the term by relating it to the other terms. These relationships can then be represented in diagrammatic form and navigated, in order to allow the meaning of terms to be agreed and explained. This classification structure is the ontology. The objective for the thesis is to build a catalogue and make use of it for decision support and costing systems, while demonstrating that the same approach could be used for other types of system(s). It is essential that this catalogue can query information from organisation's existing database systems. Most large organisations have key operational knowledge and information dispersed across different types of information systems, often in relational databases. This has the advantage of allowing the use of the standardised language Structured Query Language (SQL) to access this information.

Top Down and Bottom up Ontology development

Cheung et al (2005) explain that until recently ontologies have been predominately applied in the medical informatics field. Linking ontologies with modelling tools will also make ontologies very useful, in engineering and science, and mathematics whenever calculations are required. The open standard Stanford University (2006b) ontology management tool Protégé has been used for this purpose, although there are other ontology tools that could have been used. This ontology can be translated into a Decision Support tool Vanguard System (2006b). Vanguard is creating a modelling network where universities can share decision support models over a network. This software and the collaboration are part of the DATUM (Design Analysis Tool for Unit-cost Modelling) project (Scanlan

et al, 2007), undertaken with Rolls-Royce to make costing by analysis possible early in the design of a new product. This project involved Rolls-Royce Aerospace, UWE and Southampton University. This tool was used because it was selected during a project to evaluate, and then use software to solve costing problems. As part of the DATUM project a modelling network is being created (University of the West of England - SEEDS Group, 2006) that will link to that of Vanguard.

Top Down Example Rolls Royce CAPPe

This ontology based approach has been used in the DATUM project for representation of the information held in a process-planning tool used by Rolls-Royce aerospace, and this solution is being implemented at Rolls-Royce. The research builds on previous work to automatically produce tree representations of information requested by the user for the DATUM project. Within this project one of the tasks was to access and visualise manufacturing information held in a Rolls-Royce database system called CAPPe (T-Systems, 2007). This information was then visualised in Decision Support software Vanguard System. A program written within Vanguard System 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. Cheung et al (2007) used a similar approach of using ontology and XML based languages to communicate between different tools.

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 CAPPe link and can be made to work with any data structure. The user interface to enable this is connected to and reads from the ontology. The ontology is held in a T-Systems 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 the project to translate the CAPPe data to this ontology.

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 here is that knowledge can be collected and interacted with visually, and without the requirement to know a programming language. If specialist knowledge or deep software skills are required, this will have 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 used within this thesis, they were hidden behind a visual taxonomy layer, which allows the user to interact with the information without writing code. Macías and Castells (2004) use the approach of defining model based user interfaces using an ontology. This approach was also used in this thesis, but the ontology defines both the user interface, and the software model.

Ontology tools and the standards on which they are based are reviewed later in this chapter.

Bottom up Example - Faculty Information System

The idea behind the Faculty OnLine Data (FOLD, 2006) system is that information is provided by members of staff and students via pages developed for them, and pages developed themselves e.g. Wikis, Semantic Wikis, Blogs, and Web Forms (Orbeon Forms, 2006). This contrasts with the approach in the 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 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.

The University of the West of England (UWE) information structure is much less centralised than that used by Rolls-Royce, to facilitate ease of information entry by and on behalf of staff and students, while still allowing checks on the quality of the information. In the FOLD project, information is defined using XML (eXtensible Markup Language) and searched using XQuery (World Wide Web Consortium, 2006i). XForms (Bruchez, 2006) is an important enabling technology for creation of interactive web pages to allow people to populate the ontology. Related technologies that are important in enabling user driven content provision are those of the Semantic Web, these are explained in this chapter, and in (Hale, 2007k), (Bruchez, 2006), (Cagle, 2006), and Web 2.0 (JISC, 2007).

Combined user Tagging and Ontology Development

Anderson's JISC (2007) (Joint Information Systems Committee) report makes the point 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. Al-Khalifa and Davis (2006) and Schmitz (2006) use this approach of user tagging combined with centralized ontology development. Cayzer (2004) also discusses the role of relating ontologies, and that of tagging. Information about this is available at (Hale, 2007a), and these technologies are also referred to throughout this thesis.

Interoperability for User Driven Modelling and Programming

Anderson's (JISC, 2007) report explains that as an application becomes more popular, more people use it in order to communicate with others who use it; eventually it becomes hard to use anything else. The report talks of exposing information using web technology, for re-use. Horrocks (2002) discusses ontology languages and their role in assisting with interoperability. 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 mention that ontology tools could assist with concurrent engineering and design. 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 could be created using an open standard language such as XML (eXtensible Markup Language). Cheung et al (2005) demonstrate the importance of XML for interoperability and knowledge re-use. As the high (user) level translation, this would depend 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 end-user modellers. This is why tools such as Protégé and Vanguard System (2006b) have been created for modellers. Quint and Vatton (2004) and (2005) describe tools available for creating and editing XML documents. Until recently XML has been used to represent information but languages such as Java, C++, and Visual Basic have been used for the actual code. Semantic languages such as XML could be used in future for software development as well as information representation, as they provide a higher level declarative view of the problem.

Research into the Semantic Web is explored in this chapter; this research has been developed from the work of Berners-Lee (1999). Uschold (2003) defines the Semantic Web as being machine usable and associated with more meaning. Semantic Web technologies and the use of agents and ontologies are explained by Hendler (2001), Horrocks (2002), Horrocks et al (2003), Bloodsworth and Greenwood (2005), and Uschold (2003) who explains that "In order to carry out their required tasks, intelligent agents must communicate and understand meaning". The methodology for this is examined and demonstrated for application to decision support and modelling for engineers. The use and application of Semantic Web technology for modelling end-user programming is examined.

Structured Language, Layered Architecture, and Translation

An objective of this research is to enable users to navigate between tools such as illustrated in Figure 12 without necessarily being aware of which tool or technology they are using. The way to achieve this is through an end-user programming environment that makes use of these technologies. A person should be able to model their domain using a visual modelling language, this modelling language should then translate the representation to an abstract representation, which can be translated to open standard formats, able to be held as structured data files that can be understood by computer software. The Modelling/Programming layer provides a translation service and can perform calculations, therefore converting a source tree to a result tree. The intention is for the user to be able to use the domain layer tool without having to interact directly with any of the layers below. The results are then fed back to the user, who can drill down through the result tree in order to find the reasoning behind the results. Figure 12 shows tools and technologies that can assist in this.

| Domain Representation | Modelling/Programming | Abstract Representation | Structured Data File |
|---|-----------------------------------|--------------------------------|-----------------------------|
| ACUITy Kaon Metatomix M3t4 TopBraid Composer | | | |
| | Jena Protégé Semantic Wikis | | |

| | | | |
|---|--|--|----------------|
| PSL, STEPml, PMXML, XML with domain schemas | AspectXML, AJAX/Web2.0, XQuery, XForms, SPARQL | RDF, RDFS, DAML+OIL, OWL, RSS, SVG, VRML, UML, XMI, MathML, RuleML | XML, Databases |
|---|--|--|----------------|

Figure 12 - Language and Tool Mapping

Ontology Editors and Modelling Tools

Ontology editors and modelling tools are used either in a specific domain, for general modelling and programming or both; they make use of structured languages to represent their information. Ontology editors are used for this thesis to provide the necessary functionality to allow editing of the information used in the software models. The ontology management environments evaluated in this research are outlined below. Since the research began, ontology tools investigated and used have been improved with advances made to the translation mechanism and the quality of the web applications that are output. Applications that are built with ontology tools and include a development environment for calculation and decision support are Metatomix m3t4 (2007b), TopBraid Composer, General Electric ACUITy enterprise modelling tool, and Visual Knowledge Semantic Wiki visualiser. Such tools include Java Eclipse (Eclipse, 2006) extensions for programming. This new generation of tools make use of technologies pioneered in Ontolingua, KAON, Protégé, and Jena. Transformations that can translate the ontology into representations in other languages and tools have also been investigated. The result documents could be searched using XQuery within Exist (2006) and SPARQL (SPARQL Protocol And RDF Query Language) (World Wide Web Consortium, 2006h), and edited using XForms editors such as Orbeon Forms (2006).

Domain Representation and Ontology development tools

These tools provide a mechanism for translation of domain level modelling into open standard representation

ACUITy (Adaptive Work-Centered User Interface Technology) is an ontology-based 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" (KAON, 2006). Volz discusses issues in the development of an ontology based software environment, and explains why such an environment is required.

The Metatomix Semantic Toolkit M3t4 (2006a) is a set of standards-based plug-ins for the open-source Java Eclipse development framework, which allows software engineers to create and modify semantic applications. Metatomix provides for creation and editing of Resource Description Framework (RDF), and Web Ontology Language (OWL) resources.

TopBraid Composer (2006) is a visual modelling environment for creating and managing domain models and ontologies in 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. TopBraid Composer is based on the Java Eclipse (Eclipse, 2006) platform and uses Jena as its underlying API.

SKOS (Simple Knowledge Organisation System) is meant specifically for publishing a thesaurus on the Semantic Web. This is explained by Mikhalenko (2005).

Ontology and Semantic Tools and Translation for Modelling/Programming

Jena is a Java framework for building Semantic Web (Berners-Lee and Fischetti, 1999), applications. It provides a programmatic environment for RDF (World Wide Web Consortium (W3C), 2006f), OWL (Bechhofer and Carroll, 2004), and SPARQL, and includes a rule-based inference engine.

Protégé allows storage of information in the semantic languages Resource Description Framework (RDF), Web Ontology Language (OWL), and the relational database Access. Protégé was used on an example of a wing spar. The Protégé information from this example is read into Vanguard System decision support software.

Semantic Wikis

Ontolingua (2006) is a web-based ontology development system, which uses hyperlinks to relate concepts. It provides a distributed collaborative environment to browse, create, edit, modify, and use ontologies, with a standard web browser. The Visual Knowledge Semantic Wiki (2006) uses a model-driven architecture from the application layer down to its infrastructural levels. It utilizes packages of Semantic Agents to provide web service and web-enabled applications. These models can also be built from ontologies using OWL or RDF.

Stutt and Motta (2004) explain that "while it often seems from accounts of the Semantic Web that the various ontologies are of the greatest importance, in fact the Semantic Web will not be fully realized until a range of applications is built on top of these ontologies. Thus it is likely that semantic web services, where agents seek out suitable web services using semantic descriptions, and more particularly educational semantic web services will be most important in the next ten years. The ontologies provide the interoperable data while the services do the interesting things". This was also the theme of the 2006 Jena applications conference where the ACUIy system was presented (Aragones et al, 2006).

The methodologies used in this chapter and the technologies explained are all enablers for the technique of Meta-Programming which is applied in this thesis to make it possible for users to program.

Chapter 6 - Meta-Languages and their usefulness for User Driven Programming

Meta-programming is the writing of programs that write or manipulate other programs (or themselves) as their data (Wikipedia, 2007d). The idea behind use of this technique 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. This can prevent problems of misunderstanding, delay, and expense that often result from communication of difficult concepts between people who are experts in different areas (domain expert and programming expert). Fischer (2007) explains the related 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. Examples in chapters 7, 10, and 11 demonstrate the application of meta-programming to the problem of program development.

Meta-languages (Dmitriev, 2006), (Whiteside, 2006), (Lemos, 2006), describe the structure of information to enable it to be searched more easily by software systems. XML (eXtensible Markup Language) has become the base for many Meta-languages; this is explained by Bishop (2006), who also discusses the use of declarative languages in general. XML standards are important for the Semantic Web, many computer based reasoning systems, and for communication between different software applications. Software developers should consider XML based representations before any others as it is a common standard and semantic layers can be added to meet most needs (Figure 11 illustrated this, and this was explained in chapter 5). Any software system that does not use these standards will have difficulty communicating with other software systems. Use of a generic standard keeps open the possibility of communication with the widest possible range of other systems. Use of a domain specific standard targets the communication to a particular domain. The initial cost of a software tool is dwarfed by the investment an organization makes in populating it with data so it is important to protect this investment by storing the data in an interchangeable format. Many commercial costing and other software tools fail to provide this basic capability.

XML is an important standard in the development of ontologies. This language allows for the construction of text documents in which the relationship between concepts is represented. Because it is an accepted standard it is possible to use XML on any type of computer. Further developments such as Resource Description Framework RDF and Web Ontology Language (OWL) add a layer of standardisation of semantics, above the standardised syntax of XML (Bechhofer and Carroll, 2004). Lacy and Gerber (2004) examine how OWL can be used to aid modelling and simulation. It is also possible to represent diagrammatic, and graphical information using a variety of XML called Scalable Vector Graphics (SVG) (World Wide Web Consortium, 2006g). These Meta languages and standards also link with an objective of networked Meta-Programming using Semantic Web and Semantic Grid Technologies.

Semantic Web and Semantic Grid Research - Application to Meta-Programming

Research has been undertaken into application of the work of Berners-Lee and others in the World Wide Web Consortium (W3C) (2006b), (Berners-Lee, 1999). In order to represent information it is necessary to use Meta-languages. The use of standards for sharing information and resources is core to research into the Semantic Web. The Semantic Web involves making the Web into a repository of knowledge, which can be catalogued and searched intelligently. Software agents could then undertake this search task. 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 (Berners-Lee et al, 2001) and (Berners-Lee, 2002). To achieve this, the structure is created using XML Meta-tags, and a stylesheet provides formatting. Stylesheets are also defined using XML. The Semantic Web should make information more understandable by machines and by humans. This can help people, and intelligent agents find the information they need.

The Grid and Semantic Web areas of research are converging. The ideas and technology behind the Grid are explained in (Foster et al, 2001a) and (Foster et al, 2001b). Universities involved in Grid and Semantic Grid-computing research are University of the West of England (Olive et al, 2007), Southampton and Portsmouth Universities (De Roure et al, 2001a), (De Roure et al, 2001b), (De Roure et al, 2001c), Exeter, Liverpool John Moores (Alan et al, 2003) and (Naylor et al, 2003). The Semantic Grid involves sharing of computer resources as well as information, and does not just apply to high performance computing applications. Semantic languages and ontologies can be part of a larger effort to provide a Grid of information and applications, which can be requested as required. If users requests the answer to a problem using one computer, this can then get the help of others to solve it, Goble and De Roure (2002) explain this. Machine intelligence does not then reside in one machine but in the complex system of interacting machines. Shim et al (2002) state "Modeling Environments are becoming available as APIs so that these can be called directly into an end-user application." This enables models to become linked and inter-dependant.

Semantic and XML Standards

Generic Standards

XML (eXtensible Markup Language)

XML is used for structuring information and the World Wide Web Consortium (W3C) (2006b) ensures that the language is standardised so different software systems can interpret it. This Meta language is useful because it fits well with both an object oriented and a rule-based approach to problem solving and creating software. Using Meta-tags defined with XML it is possible to create documents, which define their own structure. Using scripting languages the information is then passed to a program which can take the value and process it or pass it to other objects within a system to facilitate decision support. The programming language objects can find the relevant XML tag within a web page and check whether the user has provided the information related to this tag (otherwise it would need to use a default value). Intelligent agent objects can be used to achieve this. An intelligent agent is an object

that can react and adapt autonomously to changes (Sycara, 1998) and (Grove, 2000) explain this. Shim et al (2002) examine how software agents could be used to search for information using user defined search profiles.

XML pages can be linked to XSL (eXtensible Stylesheet Language) pages to define appropriate formatting and so output the information in a standardised and comprehensive way. This makes it possible to provide a consistent and understandable user interface. XQuery (World Wide Web Consortium, 2006i) can be used to search XML data sources.

RDF (Resource Description Framework)

RDF represented using RDF/XML was used in this research, as this makes it possible to continue using XML tools for visualising or searching the XML in RDF/XML while also allowing the use of tools available for representing RDF. RDF consists of a resource, a property, and a property value. This 'Triple' corresponds to 'Subject', 'Predicate', and 'Object' in logic. Each RDF triple represents a fact. So RDF structures information into individual facts that link as a graph, each fact is a triple. This can be thought of as a sentence representing a fact such as 'Aircraft is a Vehicle'. An example of an RDF graph is shown in Figure 13 and the table illustrates rows of facts that make up the graph.

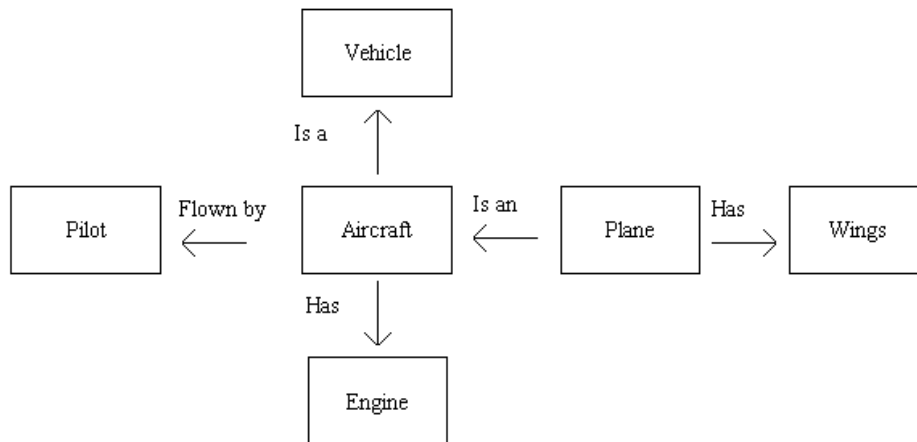


Figure 13 - RDF Graph Example, Description of Aircraft

| | Resource | Property | Property Value |
|-------------|-------------------------|-----------|----------------|
| | Subject | Predicate | Object |
| Fact | Aircraft | Is a | Vehicle |
| Fact | Aircraft | Flown by | Pilot |
| Fact | Aircraft | Has | Engine |
| Fact | Plane | Is an | Aircraft |
| Fact | Plane | Has | Wings |
| | Uniform Resource | | |

| | | | |
|--|-------------------------|--|--|
| | Identifier (URI) | | |
|--|-------------------------|--|--|

A Resource is anything that can have a URI (uniform resource identifier). A URI can look like a web address and can actually be a web address, but this is not always the case, it is a way of representing an entity. A URI consists of the name and location of the entity. An RDF Resource is described through a collection of properties and property values called an RDF Description. RDF has a mechanism for describing collections, which are special kinds of resources, and a sequence is an ordered collection. A collection does not have to possess its own URI but it can.

RDF/XML has provided a layer of standardised semantics which overlays the basic XML. RDF does not have to be based on XML there is also a format called N3 (RDF:about, 2006). RDF/XML extends the XML model and syntax to be specific for describing resources. For example an engine ring manufacture sequence can be represented as a sequence of groups of sequential operations. If a linked web page existed for this sequence there could be further RDF/XML on this web page, to describe this manufacturing sequence. This allows resources to be linked to each other indefinitely, which is why it is such an important technology for the Semantic Web. Because it is XML based, an RDF/XML Web page can be formatted with an XSL stylesheet to produce a visual representation of the structure. This is also explained in Hale et al (2003), and by (Cayzer, 2004) who uses RDF to provide structure for Semantic blogging. Oren et al (2006) also use this approach of combining RDF and Semantic Web search with ease of editing in a Semantic Wiki. An example based on an engine ring manufacturing sequence shown is in the appendix. This demonstrates implementation and code for using and displaying RDF/XML. RDF is a W3C (World Wide Web Consortium, 2006b) recommendation, this means it is a stable specification. Because a resource can represent anything, knowledge from any domain can theoretically be represented in RDF. This ability and its standardised syntax that allows it to be machine understandable are the reasons why RDF is such a useful and important technology for the Semantic Web.

RDF-S (RDF Schema) is a vocabulary description language for RDF, RDF Schema, is a semantic extension of RDF. It provides mechanisms for describing groups of related resources and the relationships between these resources. RDF Schema vocabulary descriptions are written in RDF. The RDF vocabulary description language class and property system is similar to the type systems of object-oriented programming. This is the role of the domain and range mechanisms. These class, property and relationship mechanisms are built on in DAML+OIL and OWL ontology languages explained later. RDF-S is explained in (W3C, 2006e). RDF, RDF-S, and DAML+OIL are explained by Horrocks (2002), and Horrocks et al (2003).

SPARQL is the query language and protocol for RDF recommended to the World Wide Web Consortium (W3C, 2006h).

RSS

RSS allows web users to find information by subscribing to websites that provide information they are interested in. RSS is explained in (JISC, 2007) and by Cayzer, (2004) who explains its use in semantic

blogging. An RSS feed is a list of articles in the website and a short summary of each article with a link to the full information. Software available on the web or downloadable can track the RSS information for sites the web user subscribes to. The Systems Engineering Estimation and Decision Support website has an RSS feed (Hale, 2007j), and this can be visualised using an RSS reader (Hale, 2007b). RSS has split into different syntaxes and can stand for RDF Site Summary, Rich Site Summary, Really Simple Syndication, and a third alternative called Atom. All of the RSS syntaxes are based on XML and some are also based on RDF. The incompatibilities however do not seem to hinder searches using these formats much, and use of RSS has become a useful method for making information on the web easier to find.

Tools and browsers are available for searching RSS feeds. An example of this is the downloadable Flock Browser (2007) that includes an icon by the web address to indicate an RSS feed is available for that page. RSS and Flock projects are also related to the concept of blogging that gives individuals who may not be computer literate the opportunity to put their thoughts onto a web page without needing to edit HTML. This is a similar concept to that of Wikis such as Wikipedia (2006f). RSS allows for a more structured representation of the contents of a web page or blog.

DAML+OIL

DAML+OIL is a web ontology language, resulting from a merger between DAML-ONT developed as part of the US DARPA Agent Markup Language (DAML) programme and OIL (Ontology Inference Layer) developed by a group of mostly European researchers. DAML+OIL provides an extension to RDF and RDF-S and takes an object oriented approach enabling the standardised representation of classes, properties and inheritance relationships. It also allows restriction, unions and intersections. This makes the language more expressive and more accessible to automated processes than XML or RDF on which it is built. DAML+OIL is explained by Horrocks (2002), Horrocks et al (2003), and McGuinness (2003). DAML+OIL code can be displayed with a stylesheet; this is demonstrated in the appendix.

OWL

OWL (Web Ontology Language) is a semantic markup language for publishing and sharing ontologies on the World Wide Web. W3C's Web-Ontology Working Group (WebOnt) developed OWL. OWL is derived from the DAML+OIL Web Ontology Language. OWL is a language for representing ontologies and emerged from the DAML projects and OIL projects, and was influenced by the SHOE (Simple HTML Ontology Extensions) language. The W3C OWL Web Ontology Language Overview (World Wide Web Consortium, 2004d) outlines OWL and explains "The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full" (World Wide Web Consortium, 2004c).

The intention of OWL Lite is to capture many of the commonly used features of OWL and DAML+OIL and therefore make it relatively easy for developers to create tools for its use. The differences between the sublanguages are explained in the OWL Web Ontology Language Guide (World Wide Web Consortium, 2004c) and the OWL Web Ontology Language Overview (World Wide Web Consortium, 2004d). OWL is explained by Horrocks (2002) and Horrocks et al (2003).

UML (Unified Modeling Language)

UML is explained in this thesis because it can be used for Model-Driven Programming (further examined in chapter 8). UML can therefore assist with end-user programming as explained in chapter 4). As was also explained in chapter 1, Model-Driven Programming is an important part of the user driven programming approach. Booch, Rumbaugh and Jacobson united the Booch and OMT (Object Modeling Technique) methods for representing object oriented structures. This allows software design to proceed independently of a specific language. There are now many UML tools available, one of which was used in this research.

The XML Metadata Interchange specification provides a standard format for describing UML model elements. Most UML tools contain functionality for exporting data in the XMI format. UML design translated into XMI and rendered with a stylesheet is shown in the appendix. This enabled use of a UML tool as an interoperable ontology editor. The UML tool was used to illustrate creating an object oriented hierarchy and using this as an editor for a web-based ontology. Later research used the same approach with ontology editor tools linked to a relational database to create a wing cost modeller. Nilsson et al (2002) also used UML as a meta-model for Semantic Web research. Either approach could be used for User-Driven-Programming, but whatever tool is used, the user interface to create the ontology needs to be customised to support ease of use for people who are computer literate but not specialist software developers. Kogut et al (2002) and Baclawski et al (2001) explain how UML can be used as a tool to produce ontologies. UML is important as it is a key part of the Model Driven Semantic Web. 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. El-Ghalayini et al (2005) have researched reverse engineering of conceptual UML models from ontologies. So UML is important in the concept of model driven software development. The Eclipse Open Source Java Framework (Eclipse, 2006) provides this kind of environment. The Metatomix m3t4 application (2007b) makes use of this. Metatomix provides a free semantic toolkit for Eclipse developers (2007a). This research is explained in detail in Chapter 8.

Engineering Domain Specific Standards

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) (2006b) of the National Institute of Standards and Technology (NIST) (2006d) 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 created by, users with the user driven modelling

approach, can 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 flagged 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 shop floor manufacturing. Additional information on this rationale is available at (National Institute of Standards and Technology, 2006c).

A processing sequence can be represented in PSL; this is illustrated in the appendix. PSL adds a layer of engineering semantics for communication between process modelling tools. This was devised by the Manufacturing Systems Integration Division of NIST (National Institute of Standards and Technology, 2007a). 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.

PSL-XML uses RDF (Resource Description Framework) and its own semantics to add a layer of engineering meaning to XML for communication between process modelling tools, and for use in defining ontologies (Lubell, 2006). The example in the appendix shows a section from an example PSL process sequence rendered using an XSL (eXtensible Stylesheet Language) stylesheet. This example illustrates the standard, could be expanded to a full ontology, and could allow visualisation, navigation, and interactivity. The demonstration application created is available as a web page (Hale, 2007h). These semantic languages can be part of a larger effort to provide a Data Grid of information and applications, which can be requested as required (Walker, 2003).

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 is explained by Chan et al (2003), in PDES Inc (2007), and Cover Pages (2001), STEP tools are listed at STEPtools (2007). Further explanations and an example with code are available in the appendix and at (Hale, 2007m). It would be possible for the ontology used in this thesis to integrate 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. Efforts to map STEP into the Hypertext Mark-up Language (HTML) were not wholly successful because HTML includes display-oriented tags amongst the content tags. XML (eXtensible Mark-up Language) has since become the more obvious choice.

For project management PMXML (Project Management Extensible Markup Language), (Cover Pages, 2002), (Virtual Projects, 2002), 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) which has a wider use in e-commerce and business (OASIS, 2007).

XML for Visualisation and Interaction

AJAX (Asynchronous JavaScript And XML) is an overall name for techniques to create highly interactive web pages. Ajax techniques for creation of interactive web pages assist computer literate end-users in programming tasks on the web by enabling the use of web development environments and web-based office technology such as the Google Spreadsheet (Google, 2007), Ruby on Rails (2007) and Writely (now part of Google). Ajax is explained in (Whats Ajax?, 2007) and by Cagle (2006). This type of Rich Internet Application technology research is useful for providing an environment for end-user programming. This can make office applications available over the web using just a web browser. This technology is generally called Web 2.0 because of the intention to provide much greater interactivity than previously found in web pages. Information about Ajax and Web 2.0 is available on the Ajax/web2.0 page (Hale, 2007a). XML (eXtensible Markup Language) is a programming language, not just a language for representing information, as illustrated by its use in Ajax and rich internet applications, and many products including Adobe Flex2 (2006). Quint and Vatton (2004) and (2005) describe tools available for creating and editing XML documents including Amaya (2007). XML as a programming language is covered in chapter 8, and AspectXML research into use of XML as a programming language is also explained (Peterson, 2005).

SVG and VRML for Visualisation

SVG (Scalable Vector Graphics) is a language for describing two-dimensional graphics and graphical applications using XML and is a W3C recommendation widely in use. SVG is explained in (Hale, 2007e). SVG has proved useful in the research for this thesis as an output format for representing diagrams that have been translated from a taxonomic representation to the CAD style diagrammatic representation. A simple example of use of SVG is shown in chapter 10; a more complex example is illustrated in chapter 11 and at (Hale, 2007n). SVG is also useful for providing graphics, site maps and geographical maps. Quint and Vatton (2004) and (2005) describe tools available for creating and editing SVG and other XML documents including Amaya (2007). Anderson's (JISC) report (JISC, 2007) explains the importance of SVG. The report states "At the WWW2006 conference in Edinburgh, when asked by TechWatch 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'."

VRML (Virtual Reality Modelling Language) is used for 3D modelling and animation. Kuljis and Paul (2001) explain how XML and VRML can be used for simulation. Kuljis and Paul give a summary of the aims of the AARIA project for manufacturing simulation over the Internet. The new web 3D standard is X3D Extensible 3D. This makes it possible to parse the XML and obtain values for attributes using XQuery (World Wide Web Consortium, 2006i) or XSL transformations and

stylesheets. CAD (Computer Aided Design) type 3D models can be represented using VRML. VRML enables CAD diagrams to be represented as 3D objects, which can be manipulated using the interactive facilities of a VRML player. A significant advantage of VRML simulations over other media is that they contain a subset of CAD information that can be communicated into the system. Kim et al (2001) explain how VRML can be used for collaborative viewing and amending of mechanical parts. Phillips and Rodden (2001) examine the use of VRML for building of collaborative virtual worlds. Nidamarthi et al (2001) made use of VRML to allow scientists and engineers to collaborate on the design of Nanotechnology parts.

Representation of Equations and Rules

MathML is a specification for describing mathematics as a basis for machine to machine communication. It provides a way of including mathematical expressions in web pages. This is explained at World Wide Web Consortium Math Home (2007). There is support for creation and editing of MathML documents in Amaya (Amaya, 2007), (Quint and Vatton, 2004 and 2005).

The research of the Rule Markup Initiative (RuleML, 2007) is important for this thesis. Rules for the Web have been a mainstream topic since inference rules became important in E-Commerce and the Semantic Web, and since transformation rules were put into practice for document generation from a central XML repository. Rules have continued to play an important role in artificial intelligence, knowledge-based systems, and for intelligent agents. This is now combined with standardisation in XML/RDF, enabling use of declarative rules for web services. The Rule Markup Initiative has taken steps towards defining a shared Rule Markup Language (RuleML), enabling both forward (bottom-up) and backward (top-down) rules.

Methodology

Chapters 7 to 10 explain how the problem of user driven programming described in chapters 1 to 4 was tackled using a methodology based on the meta-programming and Semantic Web standards described in chapters 5 and 6. This description of the methodology begins with early research (chapter 7), and puts the thesis methodology into the context of related methodologies of others.

Chapter 7 - Early Examples - attempts to create re-usable modelling software

User Driven Programming Early Research

The earliest research in this thesis was into providing a user driven model development example that would provide the kind of user interface and visualisation required for interactive costing. The prototype demonstrates how wing design could be aided by modelling visually and interactively with feedback provided immediately. As the inputs changed, calculations and the diagrammatic visualisation changed in response. This was provided as a visual basic executable and so is a low cost option compared to using a full CAD system. Figure 14 and Figure 15 show this.

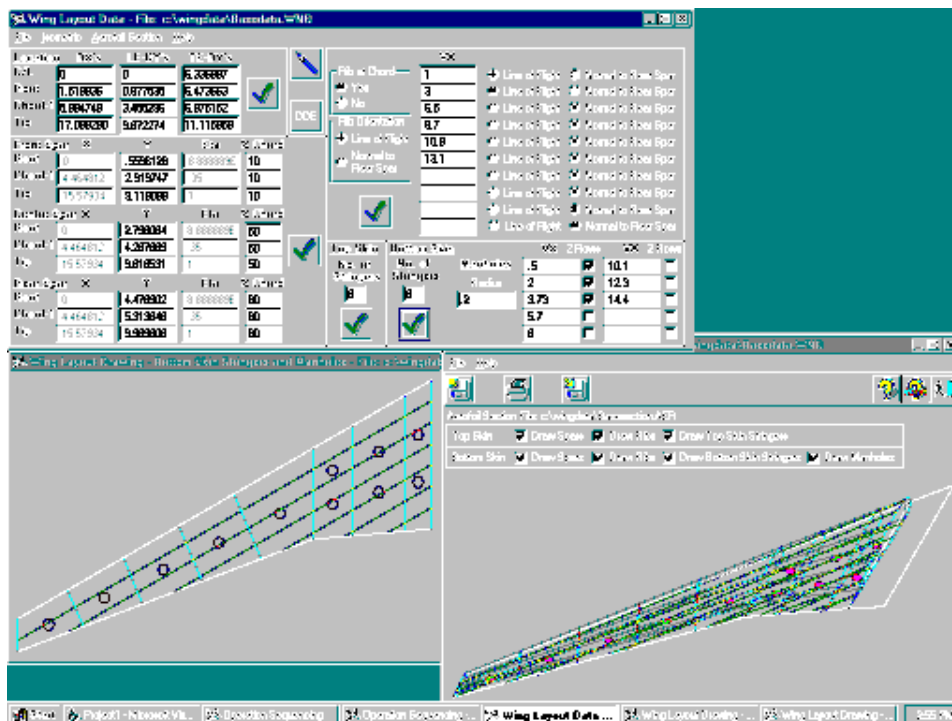


Figure 14 - Early End-user Modelling Example

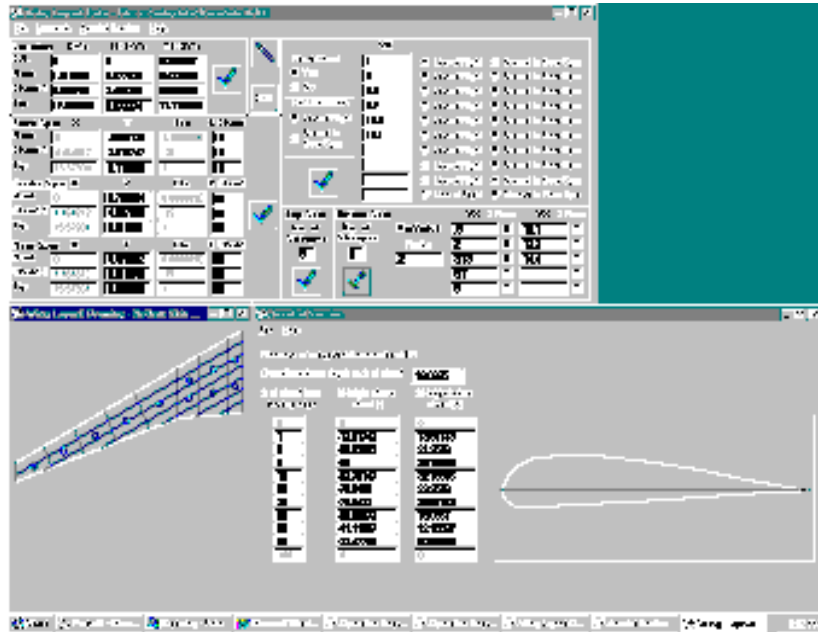


Figure 15 - Early End-user Modelling Example 2

The theory behind this is that of showing examples using whatever method most puts across the information in an understandable way. This illustrates the concept that the information represents and allows users to manipulate the information and get feedback on what has changed. This approach is used in (Cypher, 1993), this 'Programming by Example' approach is examined next.

In the mid 1970s (Smith, 1977) introduced the technique of programming by example with a program called Pygmalion. Smith elaborated on this in Chapter 1 of Watch What I Do: Programming by Demonstration (Cypher, 1993). This demonstrated the need to describe algorithms, through concrete examples rather than abstractly. 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. Guibert et al (2004) explain and expand on Smith's 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. Gross (2007) explains work involving end-user programming by designers using diagrams and scratchpads. Including the context of information allows people to discover meanings or relationships in the information that would not always be obvious. (Hanna, 2005) provides an interface for direct manipulation of shapes in this analogical way by creating an interactive triangle manipulation example using the Haskell functional programming language (Hudak et al, 2007). Jackiw and Finzer (1993) describe programming by drawing diagrams that are converted to graphical representations; they call his 'spatial programming'.

Semantic Web languages allow for the context of the information to be represented in documents and so make it possible to represent information in an analogical way, as well as allowing the kind of two way interaction promoted in this thesis, leading to an improvement in information discovery. This is the theory behind the conversions to interactive SVG (Scalable Vector Graphics) (Hale, 2007e) and tree based representations of information and functions (Hale 2007d). The Visual Basic implementation works well for making certain wing design decisions, but information and software is locked into this specific system and for this problem. For this to be re-implemented for a different problem or using different software languages more work is required. So the next stage in the research was to investigate information representation languages that could be adapted and re-used for different problems, particularly for representation of information and calculations in a generic way.

The flexibility of information representation languages could enable administrative users to maintain, adapt and extend a system themselves. The users can interact with XML via programs written in interpreted languages such as JavaScript. This AJAX (Asynchronous JavaScript And XML) approach was covered in chapter 6, and implementation with XML based languages is illustrated in the following chapters. This approach does not require the user to install software as this works within a browser. The advantage of an interpreted language is that code can be created on the fly. It is then possible for simple code to be written which follows rules in order to generate more complex code, which can then run. If a higher level visual interface is added which can communicate the wishes of users, it then becomes possible for user driven programming. Software then acts with the user in a way that seems 'intelligent' and responsive. Instead of commissioning for software development, users can communicate with a visual interface, and this translates to controlling code, which writes temporary code to achieve what the user wants. This is little more difficult than the current techniques for allowing code to generate documents, database tables, and XML on the fly. This early research was later re-implemented using SVG (Scalable Vector Graphics), an ontology and a Decision Support system with an open architecture; this is explained in Chapters 10 and 11.

For users to create their own software it is first necessary for them to be able to access and manipulate information held in an ontology. To support this, user interfaces for managing ontologies were investigated. Figure 16 shows an example of thesaurus management using an applet (Sun Developer Network, 2007), which allows a user to find information by means of a word-based search or by pressing a labelled button. The applet shows an abstract of the web page contents, and links to the full page. This and other examples from the early thesis work (Figure 14 and Figure 15) are explained in Hale et al (2001) and Hale et al (2002) as the first prototypes for user driven programming.

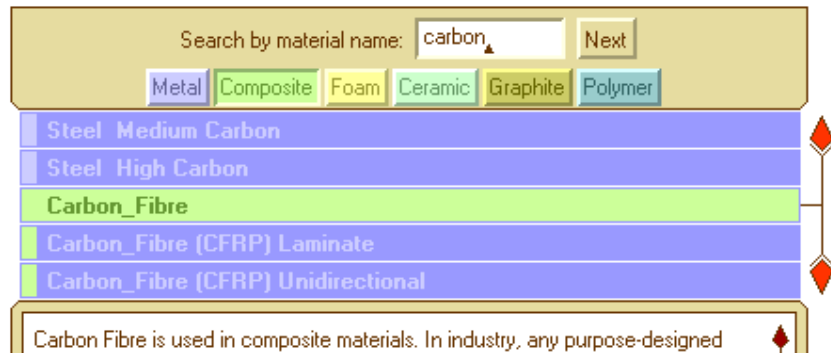


Figure 16 - Interactive Applet

This example was used to test the use of taxonomies for 'supplying a controlled vocabulary', 'site organization', 'expectation setting', 'browsing support', 'search support', and 'sense disambiguation support'. These are 6 of the 7 uses of simple ontologies explained by McGuinness (2003). Within this thesis time was spent improving the use of representations for these purposes, particularly those using a web interface. It was also important to work on making such representations more maintainable, extendable, and machine readable. Kemp and Buckner (1999) developed a taxonomy of guidance for hypermedia design making use of a design structure (the taxonomy) to relate information on design. The simple ontologies used within the thesis website were an attempt to deal with the problems that users experience with being lost in web-based systems. Otter and Johnson (2000) explain that these problems are - not knowing where to go next, knowing where to go but not knowing how to get there, and not knowing where they are in the overall structure of the document. Nilsson (2002) also explains about "surfing-sickness", where a person loses the context of the information and does not know how they got to it. For this reason tree based menus were constructed for this thesis that could be created automatically from a centralized ontology. Cayzer (2004) discusses tree based web navigation. The examples in this chapter explain this research. Eventually some ideas were abandoned and others re-used, the final implementations are shown in chapter 10 and 11.

A computer should interpret and understand the information provided by users. It is not very practical for users to specify changes to the software provider every time the manufacturing process to be modelled changes. It would be much more useful to allow users to drive the program by specifying what they want and allowing the program to respond dynamically, as is achieved with the Cost Indicating and Estimator (COINER) (Marsh et al, 2001) and (Marsh et al, 2002). (Lee et al, 2000) present a distributed visual reasoning system for intelligent information retrieval on the Web. 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.

Figure 17 illustrates experimentation with a Web-based interface. This is a modification of the Graph Layout demonstration that comes with Java jdk1.1.4. This type of user interface could allow a user to create a program using structures and visual queries and is similar to the COINER visual interface.

The following example shows an early attempt to design a web-based representation of an Airbus spreadsheet. The web pages that reproduce the functionality link to an ontology, and respond to decisions made by the user. The information used to produce these spreadsheets (but not the equations) are held in DAML+OIL XML files. This approach of adding layers of structure to a basic XML file is used in ontology editors. These files can be edited to alter the information on which the spreadsheet is based. 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 demonstrates how information from the ontology can be read into the controls and calculations are made, using the ontology information.

Machining Wizard Step 1 of 10

Figure 19 - DAML Oil Web Program Step 1 of Wizard

The yellow text boxes show calculated values. These calculations are made using JavaScript code in functions. This causes a loss of flexibility as only a programmer can alter the equations used 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. The same 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.

Although the HTML and JavaScript interface was abandoned in exchange for a more maintainable XForms implementation, (covered in chapter 9), this work demonstrated it was possible to structure information appropriately in linked taxonomies to create a structured ontology that is created using an open standard language. The use of XForms also enables users to maintain the ontology, in step with the application. The example proved 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 chapter 11.

The problem domain for the user driven programming system is that of design for manufacture of aircraft, though the theory behind this would be valid for any problem. Development of the user driven program depends on use of a catalogue view of the sections of the ontology relevant to any user, and it must be easy to navigate and edit. This information is then available in a readable form independently of the user driven-program, in order for it to have the potential to be used by other applications.

In Figure 20, the database is used for the categorisation of components and related information into a tree hierarchy. The user selects the component from an XML based menu and queries are run which return the appropriate costing. 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', and 'Weight' but this could have been extended to provide a more detailed list.

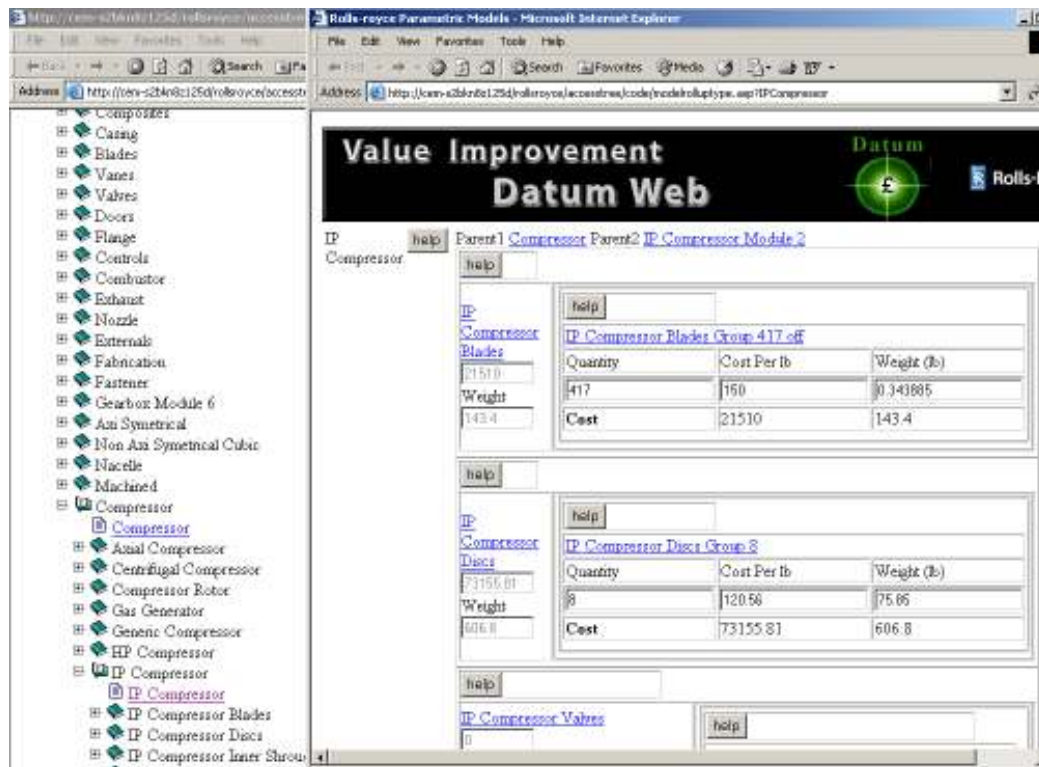


Figure 20 - Ontology of Engine Components and related information

The left hand menu uses a stylesheet created by (De Andreis, 2007). This example can be extended to allow for the choice of Materials for each component from a similarly structured Materials hierarchy. Figure 21 shows this and also illustrates how costs can be calculated and totalled.

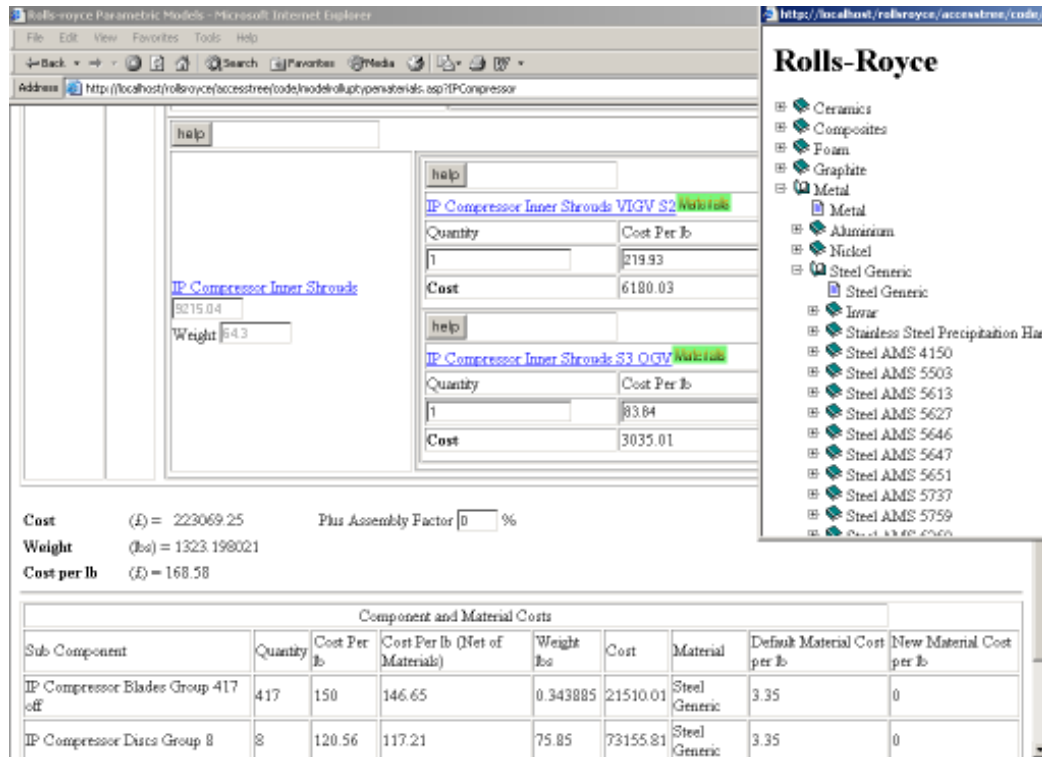


Figure 21 - Selection of Material for components

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 illustrated in chapter 9.

Ontology User Interface improvements

This research led to work on combining the use of ontologies and Semantic Web/Web 2.0 techniques with visualisation and interaction to enable users to program with a domain level view and transformation to code. Chapter 8 explains the use of ontologies and open standards for user-computer translation, chapter 9 explains application of the translation techniques. Examples in chapters 10 and 11 illustrate how ontologies can be represented, visualised, edited and translated.

Chapter 8 - Use of Semantic Web techniques for Model Translation

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. The Semantic Web has massive potential that is as yet only partially realised, and can be a useful technology to allow us to move towards domain level software development. The problem that needs to be solved is that of creating or using Semantic Web applications in large highly complex organisations, or collaborations to pull together information from diverse sources, and enable it to be used for modelling problems. The research for this thesis examines ways of structuring information, and enabling processing and searching of the information to provide a modelling capability. The thesis also investigates ways to increase user involvement in software, and providing templates to enable non-programmers to develop modelling software for the purposes that interest them. It is very important to involve users in software development (Olsson, 2004). To assist in this, it is essential that new ways of enabling collaboration between all those involved in software creation and use are investigated. Johnson et al (2003) and Johnson (2004) examine how this kind of collaboration can be achieved and tested. The main advantage of open standard representation of information provided by 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 user to computer and back to the user for conveying results of a modelling run.

Translation for De-abstraction

Crapo et al (2002) citing Jones (1996) "observes 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. Representation will be central to any attempt to enhance the creation and use of models." The purpose of the research into this problem for this thesis is to find a way of translating the conceptual model, as defined by a visual and human language representation into a structured representation. The requirement of this structured representation is that it can be made use of by computers in order to generate program code. This involves the use of model-driven programming as a technique to assist this. 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. Spahn et al (2007) explain that end-users are domain experts not IT professionals, and 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.

Translation for the Modelling Process

The long-term goal would be to automate the steps from design to manufacture. This implies the use of intelligent automated reasoning at all stages of the design process. This is a difficult task, (Duverlie and Castelain, 1999), (Eaglesham, 1998), (Qu-Yang, Lin 1997), and (Feng et al, 1996) discuss this. As a

step towards this aim it is possible to automate some decision making, but this decision making would be driven by the software user. As mentioned in chapter 4, Nurminen et al (2003) explain that their testing indicated that users prefer usability over automation. The user can then make some high level decisions which allow the software to perform relevant cost calculations and make more detailed decisions. The system can then be used at the start of a project, late on in development, and after production has begun (to evaluate possible design changes). Some of the software might be re-applied for other product types. This makes the ability to translate the software into different representation and software environments essential. Accurate cost estimation requires a high level of part definition but the best opportunities for decisions, which reduce product cost, are early in the product life cycle when part definition is still ill defined. So the level of detail which users can provide varies along the design process. It should be possible for the user to create a model early on even if only high level information is available, and return to improve the model whenever necessary, as more detailed information becomes available.

So far in this research an ontology representation was edited of the program to be created, in an ontology tool Protégé, and Jena was also investigated. For each node in the tree, there is a value (end nodes), or a mathematical formula that relates that node to other nodes. The ontology created is saved in a database and then read by a decision support tool (Vanguard System). This tool iterates through the ontology, and for each node in the tree calculates the results of all the formulae. The calculated tree is then displayed with the results; the calculation examples in chapters 9 to 11 illustrate this.

Software created with Vanguard System allows calculations of the cost of a design, and provides a colour-coded representation of the product tree. It is then possible to output this tree as web pages, interactive diagrams, and code in programming languages such as Engineous Java based costing tool Cost Estimator (Koonce et al, 2000). This is explained with examples in chapters 10 and 11. It is possible to search the information both in Protégé and on the Web as the information is represented using searchable Semantic Web languages. The results can also be exported into representations using various combinations of HTML, XML, Java, JavaScript or Flash as required for a web view, and can be output in virtually any language by adapting the existing translation code. Working models including (Hale, 2007c), (Hale, 2007e) demonstrate this. A further transformation can be made to translate the tree based representation of a component e.g. a spar to a diagrammatic view of the design (Hale, 2007e). Program transformation is explained at (Program-Transformation.Org, 2007a) and (Program-Transformation.Org, 2007b).

Model-Driven Programming

Model-Driven Programming involves creation of a high level interface for editing models to represent user's ideas, which can be translated to program code and alternative visualisations. Model-Driven Programming and the Semantic Web are explained in Frankel et al (2004). Model-Driven Programming and Visualisation is examined by (University of Victoria, 2006), (Storey et al, 2004) and (Elenius, 2005). 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 was covered in chapter 6. Model-Driven Programming and Meta-Programming together with Semantic Web and end-user programming techniques are vital ingredients of the User Driven Programming approach used in this thesis. Horrocks (2002) explains the use of meta-data annotations and how they can make resources accessible to agents.

Model-Driven Programming also 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 (covered in chapters 4 and 6), architectural descriptions, and requirements specifications. A fuller explanation is available at Program-Transformation.org (2007a). 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. An explanation is available at Program-Transformation.org (2007b).

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. Coutaz (2007) explains how Model Driven Engineering and Service Oriented Architecture can be combined.

Translation Steps

The research aim is 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 should not have to write any computer code. Instead diagrams, natural language, and formulae would be used to define the source model. As far as possible the tools and techniques used should be open standard for ease of use, re-use or transformation for other hardware or software systems. The main advantage of open standard representation of information provided by 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 at (Hale, 2007f) and in chapters 5 and 6.

A model as defined by a user could be translated to a model that is more suitable for a computer to interpret. Software can 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 representations. Trees are defined recursively because their structure is recursive, so it is natural to traverse them recursively. This approach deals with hierarchies and relationships, but for requirements outside this scope Aspect Oriented Programming (Elrad et al, 2001a and 2001b) and (Murphy et al, 2001) could be used to capture and translate these requirements. Aspect Oriented Programming can be 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 can then be translated into a computer language representation such as AspectJ for Java (Kiczales et al, 2001) and AspectXML for XML (eXtensible Markup Language) (Peterson, 2005). Aspect Oriented Programming makes use of XML as a programming language not just an information format. Collaborations involving Aspect Oriented Programming can be found at (Aosd.net, 2007) and (AspectXML, 2007).

A further translation is necessary from the program to a result model 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 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. The result model could be represented diagrammatically or as categorized and linked web pages. 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 this meets their expectations, the translation will be successful. Decisions a user makes can affect both the content and the presentation style of the results received.

Figure 22 shows a plan for weaving Aspect-Oriented Programming into translation for end-user to computer communication. Aspect-Oriented Programming can be used where certain tasks or properties do not fall within a natural hierarchy. These are called Cross-Cutting Concerns (Wikipedia, 2007a); these cross-cutting concerns could be tasks the program needs to perform such as providing printing or security. This same technique could also be used for attributes that the program is to model, for example if the program is to model an aircraft wing. A user with sufficient computer literacy skills can model the representation of the wing as a hierarchical diagram. The user can specify relationships between these items that make it possible to make calculations and decisions. For this model some parameters such as efficiency, weight and cost might not fit well in this hierarchical representation. So the parameters could be weaved into the program as cross-cutting concerns in a similar way to the computing parameters.

Once all parameters are weaved into the program, it can be translated from a format most suitable for holding user's information (e.g. OWL Web Ontology Language), into a computer language such as Java for implementation. OWL (covered in chapter 6) is a language for representing information in a standardised and searchable way, and is built on XML (eXtensible Markup Language) (Hale, 2007s) and RDF (Resource Description Framework) (Hale, 2007i). The program would calculate results, and these could be translated back to the user. XML (eXtensible Markup Language) can also be used in the translation as either a programming language (AspectXML, 2007), or a language for representing results. The results will be fed back in the language that is best for user interaction and visualisation. Results can be visualised using stylesheets and interactive software, and translated further into kinds of

representations other than trees e.g. SVG (Scalable Vector Graphics) diagrams and graphs. Examples of this are available at (Hale, 2007d). Figure 22 shows the translation process.

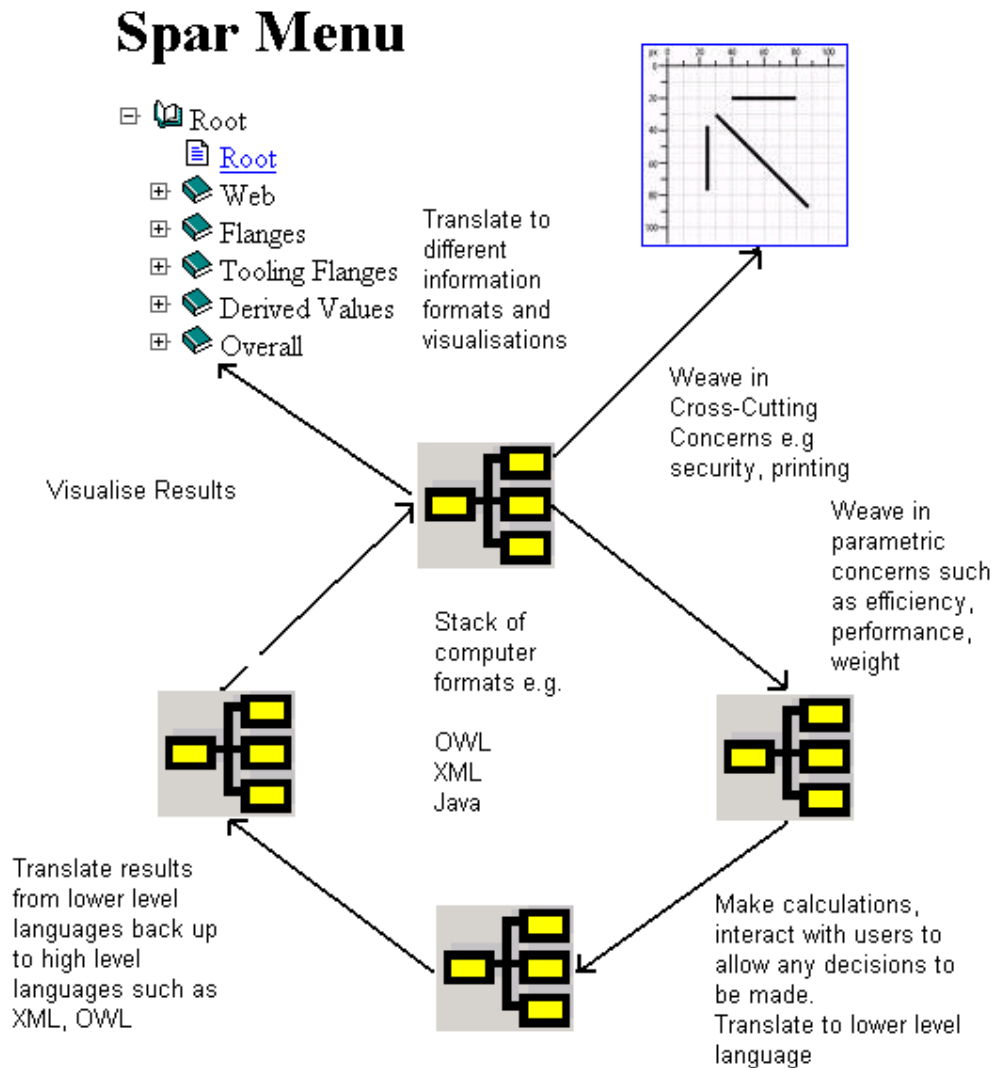


Figure 22 - Translation and Aspect-Oriented Programming

Shim et al (2002) examine translation from a user's model to equations and explain "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". For a simple practical example Figure 23 explains the concept for the representation of the equation $E=MC^2$. This relationship can be defined by the user. Here this is 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. 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 Java for human friendly visualisations viewable as web

pages/diagrams. Units have been left out as the type of equation used and values are not important to the concept.

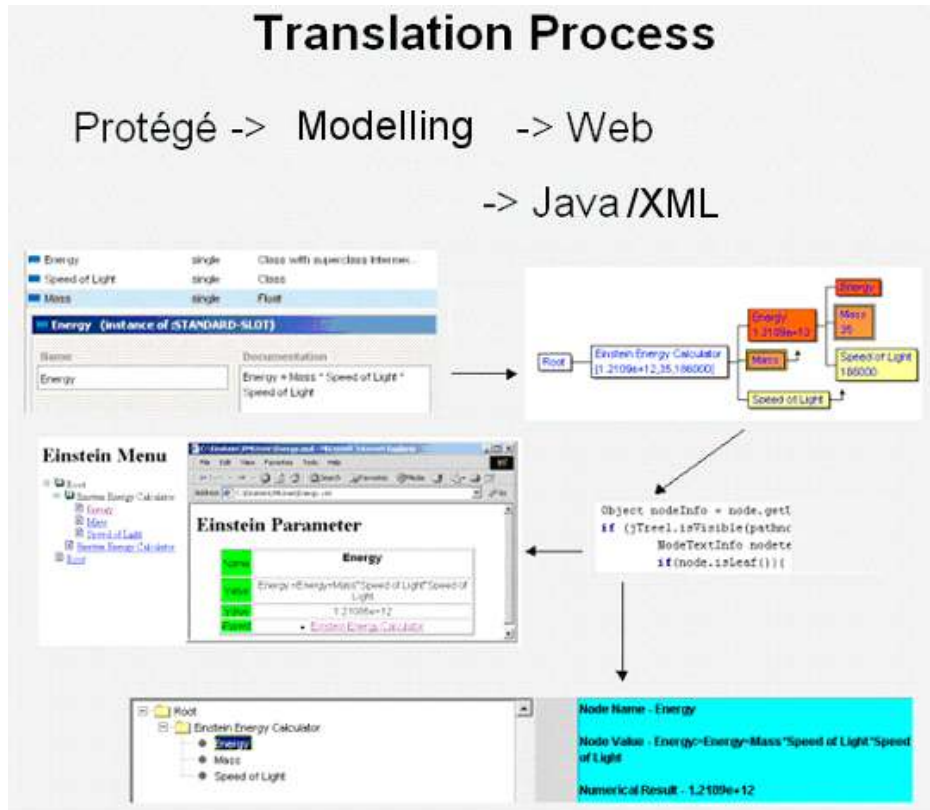


Figure 23 - Translation Process

Dynamic and Responsive Software Translation Mechanism

Figure 24 illustrates the aim of a two way translation between all levels in a hierarchy of translation between human and computer, and between different software environments. The definition of interaction used by Simons and Parmee (2006) referencing (Wikipedia, 2007c) explains the inspiration behind the translation research in this thesis "a kind of action that occurs as two or more objects have an effect on each other. The idea of a two-way effect is essential to the concept of interaction, as opposed to a one way causal effect. Combinations of many simple interactions can lead to surprising emergent phenomena". This communication could improve opportunities for end-user modelling and programming, sharing of information, and education of both users and computer software. The analogy of educating computer software to do what the user intends is called programming by demonstration (Cypher, 1993). The user has the role of an educator of the software which acts as an apprentice to learn what is required. The user is thus able to instruct the software and so program.

From experience gained during this research it is evident that the main challenge in software projects is to achieve a match between what is required from users of software and the functionality and capability of the software system provided. The methodology for this thesis involves automation of the

programming task to enable user driven programming (UDP). This provides computer literate users with a tool to enable them to drive the creation of software without the requirement that they become familiar with the syntax of a programming language. The system outlined is an attempt to help users who are experts at their own jobs and not computer professionals, to develop models that aid in their work. This is a very common situation for designers and other engineers. The research and implementation is later demonstrated through the wing box example, used in a spreadsheet in chapter 3. Figure 24 illustrates the alternative approach to provision of a system to enable users to create and/or use their own models.

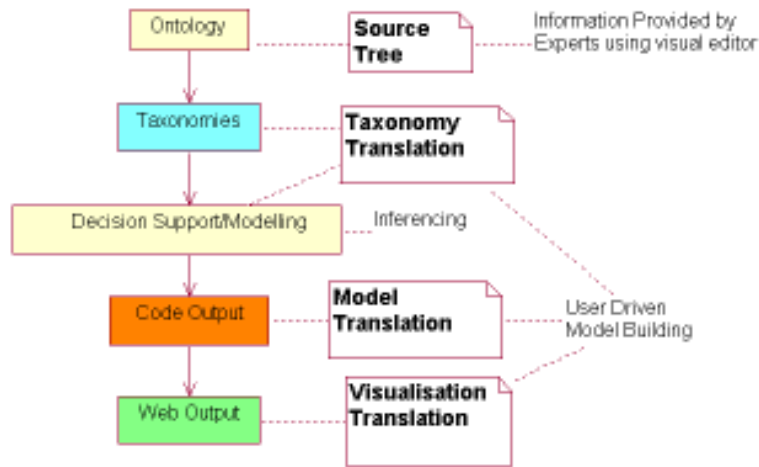


Figure 24 - Translation Process Chain

This research involves adapting or creating software systems to provide the visual editor for the source 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 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. These models are a more specific subset of the generic model, and could be applied for their own analyses. This thesis concentrates on provision of a translation mechanism to convert information or models into many languages (primarily web-based), and to visualise this information. Examples of this are shown in chapters 10 and 11.

Research Theory influencing this Translation Mechanism

As mentioned in chapter 1 the use of the Semantic Web in this thesis is a means for open standard representation of information (built on XML), 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 can 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 has been prototyped, and it is important to further this research in order to establish a user base, and make the translation generic.

As mentioned in chapter 1 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 mismatch of data between object oriented and relational database systems identified by Ambler (2003). The information for the examples is highly structured and visualization 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 was used that consisted of several related taxonomies. This ontology provides a design costing capability, but the ontology and the techniques used to build it could be re-used for other purposes, Bloodsworth and Greenwood (2005) use a similar 'ontology-centric' approach. 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. Issues about visualisation of light weight ontologies are examined by Fluit et al (2003). Hunter (2002) evaluates engineering ontologies and gives examples. An important reason for creating an open standards central ontology is that it can be accessed by many different applications. Research of others in this field has been investigated (Corcho and Gómez-Pérez, 2000), (Corcho et al, 2003), and (Noy, 2004). The open standard OWL (Web Ontology Language) was used; this is explained in chapter 6, and by Bechhofer and Carroll (2004).

The wing box 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 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 chapter 6. This can make the translation simpler and enable interaction with other systems. This kind of solution is referred to by Uschold and Gruninger (2004).

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). During this project one task was to automatically produce tree representations of information requested by users. Information held in a relational database was visualised and exported in structured languages, this was explained in chapter 5. The information was visualised in decision support software (Vanguard Software, 2006b). This enables modelling of product design and manufacture, and provides statistical techniques for modelling uncertainty. Da Silva et al (2002) examine how semantic knowledge can be shared for modelling of uncertainty. For the thesis examples, information was visualised in a colour-coded tree. This information was also output as XML (eXtensible Markup Language) and SVG (Scalable Vector Graphics), and linked to stylesheets to create a web-based tree representation. Also parametric cost models were created with Christophe Bru for web use (Hale, 2006g) and a tree based menu for browsing of these or other web pages (Hale, 2006c); these examples are explained in chapter 11.

Dynamic software systems such as outlined by Huhns (2001) have been examined for this thesis. Huhns explained that current 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 created today.

Paternò (2005) 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'. Stages take development through to a user interface that consists of interaction objects. 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". 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".

The essence of the research methodology is that a high level visual interface is used to create a Meta-program, which can communicate the wishes of users. Figure 24 illustrated the theory behind this and chapter 9 to 11 the implementation. This requires the definition of taxonomies to provide the library of information to be used in the decision support modelling. A model builder could populate and maintain this without needing to know programming languages. It 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 designer can communicate with a visual interface, and this translates to controlling code, which writes code to achieve what is required. A program 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 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. As mentioned in chapter 3 (Schrage, 1991) explains how difficult it can be to find the underlying assumptions that are represented in a spreadsheet scenario.

The alternative in this thesis to use of spreadsheets or commissioning of software 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, 2006) or Simkin (Whiteside, 2006). 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 describes this "The Web is a network of interlinked nodes (HTML documents linked by hypertext)" (JISC, 2007).

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 the 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 this 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.

Chapter 9 explains how this translation methodology combined with visualisation techniques enables users to create models/programs, and examines how user interfaces can be created for this. Chapters 10 and 11 show examples from this approach.

Chapter 9 - Enabling User Driven Model Development

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 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 was also important to examine visualisation, to create a human computer interface that allows non-experts to create software.

Criteria necessary for User Driven Model Development

This chapter explains the factors necessary to make the User Driven Model development approach possible. Firstly it is necessary to find a way for people with little programming expertise, to use an alternative form of software creation that can later be translated into program code. The main approach taken was the use of visual metaphors to enable this creation process, although others may investigate a natural language approach. The decision on what combination of diagrammatic or natural language representation to use may be influenced by the type of user and the domain to be modelled. Engineers usually deal with diagrams as a regular part of their work, so understand this representation particularly well. In fact developers also use metaphors from engineering diagrams in order to provide a user interface for software design e.g. (Quigley, 1999).

As explained in chapter 8, a translation method can then be provided that converts this representation into program code in a number of languages, or into a Meta-language that can then be further translated. In order to achieve this, it is necessary for the translator to understand and interpret equations that relate objects in the visual definition, and obtain the results. In order for the user to understand the translation that has been performed it is then important to visualise the translated code, and this must be accessible to others who use the translated implementation. Web pages are a useful mechanism for this as they are widely accessible. The visualisation of results is essential to express clearly their meaning. Words in a report document can be ambiguous, and there often is insufficient indication as to how the results were calculated. So the relationship of results to inputs must be clearly shown.

Figure 25 shows how a definition can be created without entering code in a computer language. In the expression above the diagram there are no variable declarations or computer specific keywords. The only special characters are the square bracket. The text representation is converted into a visual representation of the equation $E=MC^2$. This conversion would be essential in a larger model to achieve a useful visualisation. If the model builder defines the model in the ontology editor there is then no need for the user to enter the equation, instead it would be available to all models automatically. This is achieved using a visual ontology editor (Protégé in this case), and this definition can be read directly by Decision support software (Vanguard System) that can visualise it and colour code it. For a more complex example (chapter 11) the user interface allows choices to enable a user to define the problem.

Automated translation from the ontology allows collaborative modelling in the decision support tool based on the shared information. Units have been left out as the type of equation used and values are not important to the concept. The software can translate the source model into a program and calculate results.

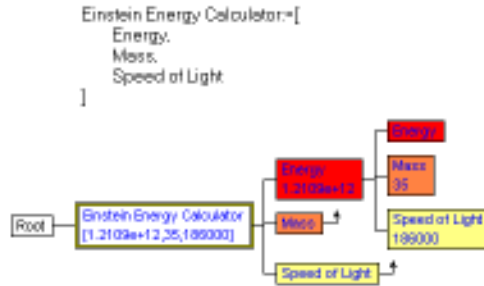


Figure 25 - Visualisation of Equation

In the examples, from this simple one to other simple examples and more complex examples presented in chapter 11 common definitions and relationships for items, values and equations for an ontology were not agreed with others, but just defined as convenient to the test models. (Gruber, 1993a) examines how agreement could be achieved, and using an engineering case study examines how useful such an ontology would be to engineers, and others who often make use of equations and values with standard units. The implementation of the research would also require the use of high level tools for editing ontologies and models; these were investigated in chapters 5, and 9, and will be examined again in chapter 12. A simple example of the methodology used is illustrated in chapter 10.

Application of Semantic Web and Meta-Programming to User Driven Modelling

This research can be taken further than just translations. The Meta information that describes the concept of an object and its relation to other objects can be used for automated construction of software as well as representation of knowledge. This automated construction could give computer literate users the tool to enable them to drive the creation of software. This is the core of user driven programming. Structured language can be visualised in ontology editors such as Protégé. This representation can be translated into computer code. This could shorten the time software development takes by allowing domain experts to construct software directly, without requiring them to learn a computer language. It also has the potential to be computer language and system independent, as one representation could be translated into many computer languages and Meta languages. The increase in computer power expressed in Moore's law (Voller and Porté-Agel, 2002) and achieved to date, makes development of such a system achievable. A prototype of an ontology to computer code translator has been created and is explained in chapters 10 and 11.

Collaboration, simulation and modelling have been investigated for this thesis, to assist in determining the requirements for future research in modelling of problems. Thomson et al (2001) also investigated alternative approaches to software development, which give users greater involvement, this partially

automates the process of user interface creation by providing a means to manage a hypermedia concept map. Huhns (2001) and Paternò, (2005) both explain that alternatives to the current approach to software development are required. This should allow translation from a model-based representation of software to the actual software as explained in chapter 8. This can involve 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) standards based languages, and the visualisation of information represented in such W3C standard ways. Modelling languages such as Alloy (2007) explained by (Wallace, 2003) and LPA VisiRule 1.0 (2007), can be developed as an interface to an End-User Programming environment. Transformation from a model building environment to program code has been investigated by (Gray et al, 2004). Experienced programmers can build a modelling environment that can 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. MathML (World Wide Web Consortium Math Home, 2007) can 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 was explained in chapter 7. 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. As explained in chapter 7 a representation should visualise the concept and so be 'analogical'. This is the theory behind the conversions to interactive SVG (Scalable Vector Graphics) and tree based representations of information and functions demonstrated in chapters 10 and 11.

Capturing Information

Highly interactive web pages that act like programs to provide a user interface can be used to provide an interactive user driven programming environment. One methodology investigated is use of XForms (Bruchez, 2006) (Dubinko, 2005) web forms to capture the information required for each node in a tree. 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 information from users for an ontology. An example of this is Orbeon Forms (2006), which is open source software that can store information in an Exist (2006) 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 (chapter 7). XForms can be used for managing information from XML files. It is possible to add, delete and edit any XML file, so this provides 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 from its text representation. Figure 26 shows example wing component information edited using this system.

| Category: | Component | Sub Component(s) | Total | Qty |
|---|-----------------|---------------------------|---------|--------------------------------|
| <input type="button" value="Spar"/> | 1. Spar Part3 | | \$11.99 | <input type="text" value="0"/> |
| <input type="button" value="Stringer"/> | 2. Spar Part2 | | \$12.99 | <input type="text" value="0"/> |
| <input type="button" value="Skin"/> | 3. Spar Part1 | Spar Part1a Spar Part1b | \$6.99 | <input type="text" value="0"/> |
| | 4. Spar Part1b | | \$3.00 | <input type="text" value="0"/> |
| | 5. Spar Part1a | Spar Part1a1 Spar Part1a2 | \$3.99 | <input type="text" value="0"/> |
| | 6. Spar Part1a2 | | \$2.00 | <input type="text" value="0"/> |
| | 7. Spar Part1a1 | | \$1.99 | <input type="text" value="0"/> |

Figure 26 - XML Based Software for capturing information

This application enables selection of components and sub components, and calculates the total cost. XForms was also used to manage staff information, so 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. Also Protégé was used in the research to perform this task offline, future work will involve construction of an online user interface for model creation software that makes use of XForms and other XML and RDF programming standards, this is explained in chapter 12. The interactive examples shown in chapters 10 and 11 demonstrate the kind of calculation and modelling that could now be achieved more easily using XForms. For these examples, combinations of XML and JavaScript were used that work in a similar way to XForms and enable the same kinds of user interaction. XForms can be used in combination with XQuery (McGovern et al, 2003) to allow for completely XML based interactive applications, as shown in Figure 27.

Edit Spar Part1

Staff Details

Name

Price

Quantity

SubComponent

SubComponent

SubComponent

[Home](#)

Description Spar Parts and Sub Parts 1

Figure 27 - Automated Generation of Web Forms

The interactive examples explained here can be found at (Hale, 2007r). 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 enables testing of the techniques advocated in the thesis (explained in chapter 12).

Final Research - Putting it all together

Chapter 1 explained the importance of the research for this thesis as the translation of requirements of users into software, and results back to users. This is currently time consuming and causes errors and confusion. The other main problem this research addresses in order to facilitate this translation is collaboration of users between each other and with software models. At present many problems and delays are caused by the failure of interoperability within the software structures of organizations. Further possibilities for automation of the User-Driven interface involve the use of additional components to ontology tools that would allow for integration and higher level manipulation of the ontology, translation software, calculation software, and the visual interface. User Driven Programming applies the interactive form based editing capabilities evaluated in this chapter to model creation, editing and visualisation. Garcia-Castro and Gomez-Perez (2006) discuss how this can be achieved and measured. Software evaluated in this thesis enables translation from user specifications to software, and creation of a User Driven Programming application environment. The research for this thesis concentrates on creating or using applications for modelling, searching and sorting information, and translating this information model into code. The technique should be usable for most types of program development. Other's research relevant to User Driven Programming in general has been covered, as this can be applied to the problem.

User Driven Programming and User Driven Modelling can increase user involvement in software by providing templates to enable non-programmers to develop modelling. If more users of software are involved its creation, and the source of the code is open, this enables creation of development communities that can share ideas and code and learn from each other. These communities could include both software experts, and domain experts who would be much more able to attain the expertise to develop their own models than they can with existing software languages.

Ontology systems such as Protégé (Stanford University, 2006b) (examples in chapters 10 and 11), Jena (2006) and KAON (2006) either individually or in combination, can be used to create an ontology in order to organise models. The above systems can read and write the ontology in a database, or use Semantic Web languages such as XML, RDF and OWL. New systems have been released that act as a development environment to enable Semantic Web programming in order to create software systems overlaying an ontology. These new systems can abstract some of the details of developing in ontology tools. This can make it easier to provide an end-user programming environment, and to deal with complex systems. The software created for the thesis can make use of the new Semantic Web decision

support applications that are being provided by other organisations, such as General Electric's ACUIity enterprise modelling tool Aragonés et al (2006), Metatomix m3t4 (2007b), and TopBraid Composer (2006). Generally these tools build on or incorporate the earlier work of tools such as Protégé and Jena, and can also make use of the Java open source development tool Eclipse (2006), future research on this is discussed in chapter 12. The new applications provide extra layers of software for accepting input then calculating and visualising results. This can enable collaboration by making the use of visual menu and search tools possible for finding and sharing models. This is similar to the way RDF/XML syntax is used to enable the searching and sharing of web pages. The use of RDF/XML allows XQuery and SPARQL (SPARQL Protocol And RDF Query Language) (W3C, 2006h) to be used for searching, as mentioned in chapters 5 and 6. The ability of Jena and Protégé to save in relational database format also makes it possible to use SQL (Structured Query Language). Using these standards it is possible to represent information in various ontology systems. This flexibility is useful when different organisations are not all using the same applications, and new applications are being developed. An important reason for creating an open standards central ontology is that it can be accessed by many different applications. Further semantic representation capability is provided by the open standard OWL (Web Ontology Language) which builds on the layers of XML, and RDF, as explained in chapter 6 and by (Bechhofer and Carroll, 2004). RDF/XML can be used to encode an ontology. Fensel et al (2000) and (2001) describe Ontobroker (or On2broker), and the use of XML and RDF within this ontology tool. The use of ontologies is being driven by e-commerce and e-procurement where trading is online (UN/CEFACT and ebXML 2007). The advantages of open standard representation of information in allowing the sharing of ontologies were explained in chapters 5 and 6. Ontologies can then be read into a decision support system which outputs results onto the web, this allows models that are relevant to different organisations to be shared. The creation of a web-based visual representation of the information allows people to examine and agree on information structures. This is combined with a translation system to create software from diagrams that represent the problem. The background research has been on Semantic Web techniques that can be applied to this. A key objective of this research is to provide a tool that can be used by people who do not have access to CAD tools or other specialist software. This objective is to aid communication of product information throughout an organisation.

Figure 28 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 have already been prototyped, but a modelling system has not yet been developed for use outside the university.

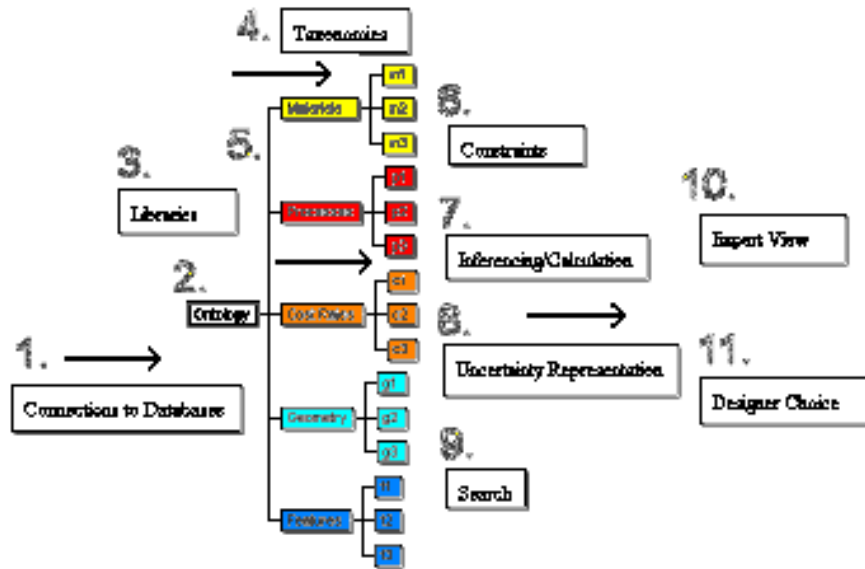


Figure 28 - Semantic Web Modelling system

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 (Bechhofer and Carrol, 2004), and an interface built to allow domain experts to edit the ontology.
3. Libraries are created in a partnership with domain experts.
4. Taxonomies are populated by model builders who want to use them for their modelling problem. These are based on the libraries created in step 3.
5. Taxonomies are colour coded for ease of understanding. A link is created between the ontology tool and a decision support and calculation tool (Vanguard System), 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 in any calculation, this produces a multicoloured result tree 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 can also be used to represent cost, time, or uncertainty.
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, 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, and this filters into changed results. This can then support a cycle of results and rework.

The above approach to modelling is explained in (Hale, 2007f). Visualisation is very 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 10 and 11 and in (Bru et al, 2002) and (Bru et al, 2004). The elaborator 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.

Chapter 10 - User Driven Modelling Techniques implemented with a simple example

This simple model explains all the concepts behind the visual and language representation of the problem to be modelled. It also shows the translation steps required to convert this representation to program code. It explains how alternative tree based and diagrammatic based representations can be used to convey different views of the translated software. These examples illustrate the methodology for this thesis, into improving the modelling process and enabling end-user involvement, the production of models is not the main aim.

The methodology involves creating structured taxonomies that would eventually be part of an overall ontology of information relating to aerospace, and the automated generation of software to access and process this information. This approach could also be used for other types of modelling problem. The explanation uses the example of the taxonomy definition of a simple rectangle. This will be used to illustrate how models can be created. The representation is transferred to the Vanguard System decision support software that is used as a calculation engine and translator. The transformation from the Protégé ontology representation to the modelling tool is automatic because recursive ODBC calls to the Protégé data structure were written and embedded into the translator code.

A movie illustrating this example is available at (Hale, 2007o). The ontology editor can be used to produce anything from a simple taxonomy Figure 29, to a large and complex ontology (chapter 11).

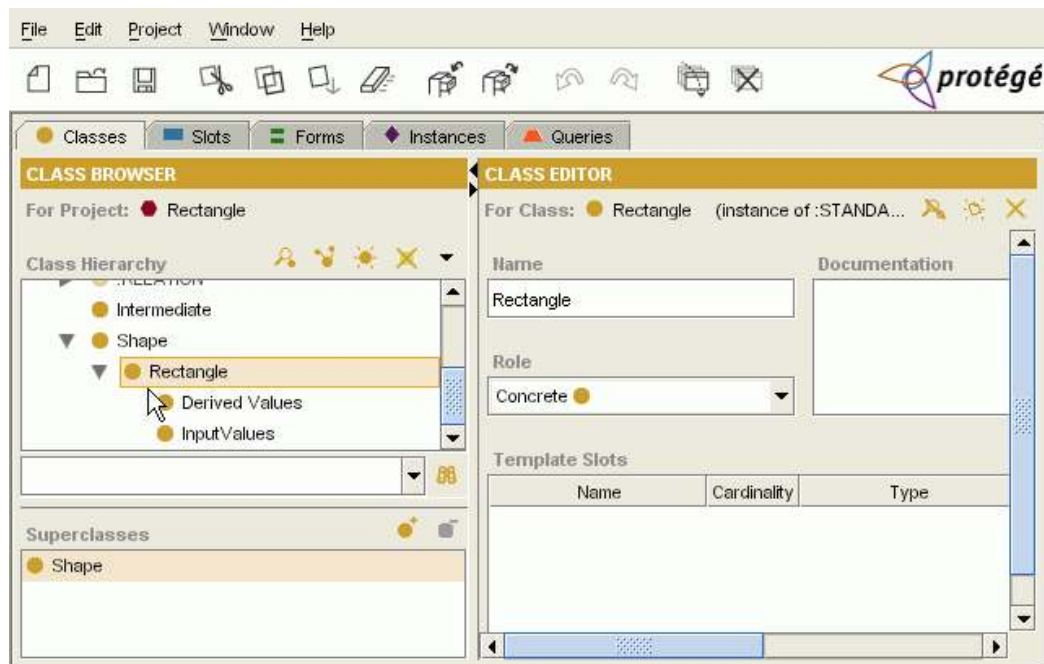


Figure 29 - Rectangle Explanation - Protégé 1

In Figure 30 the component is created, 'Input Values' are defined and these can be used in a calculation.

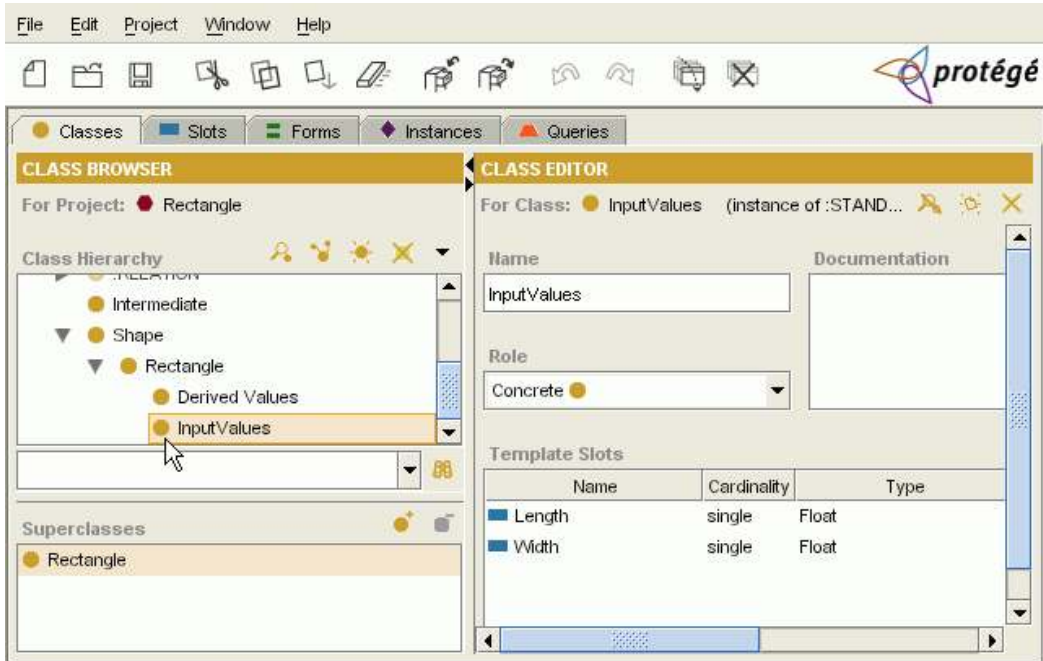


Figure 30 - Rectangle Explanation - Protégé 2

In Figure 31 the length of the rectangle is defined. This is given a value of 4 metres.

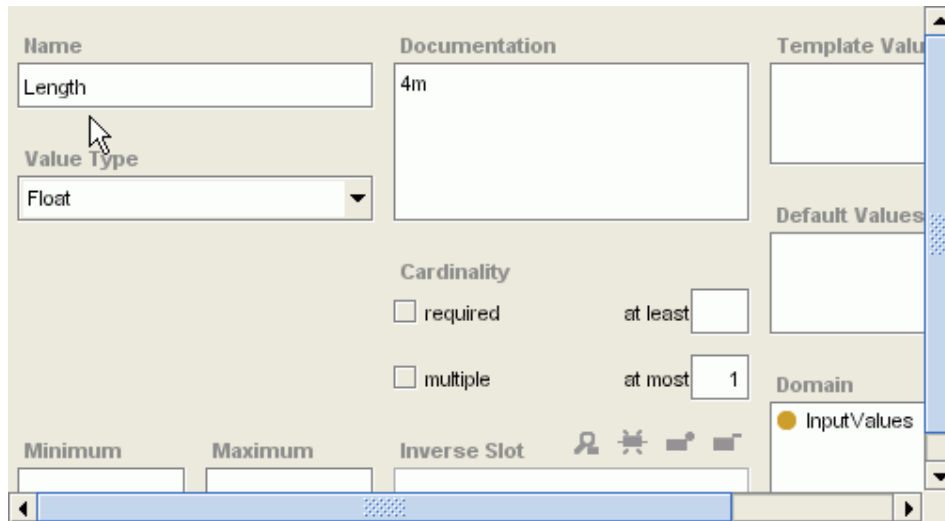


Figure 31 - Rectangle Explanation - Protégé 3

In Figure 32 the width of the rectangle is defined. This is given a value of 2 metres.

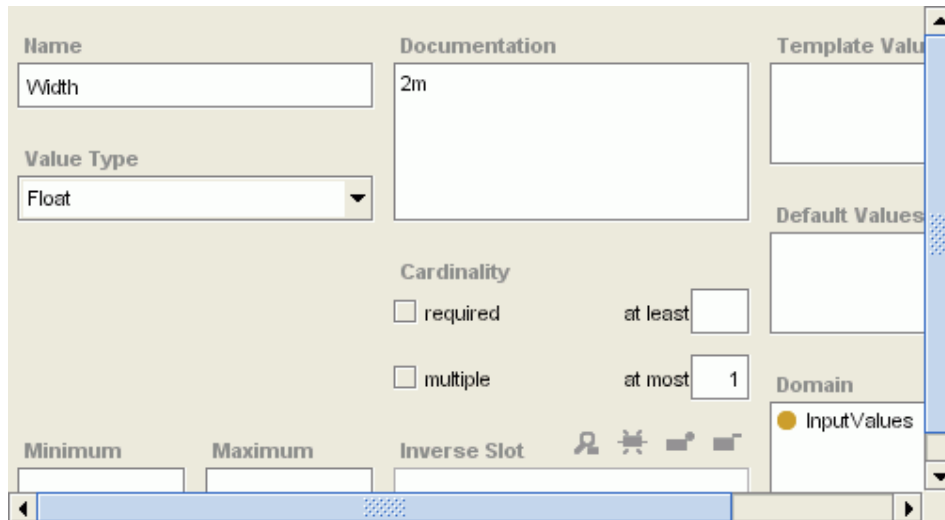


Figure 32 - Rectangle Explanation - Protégé 4

In Figure 33 another class is created for the results of calculations. This is called 'Derived Values', but could be given any name. In this example there is just one derived attribute 'Area'.

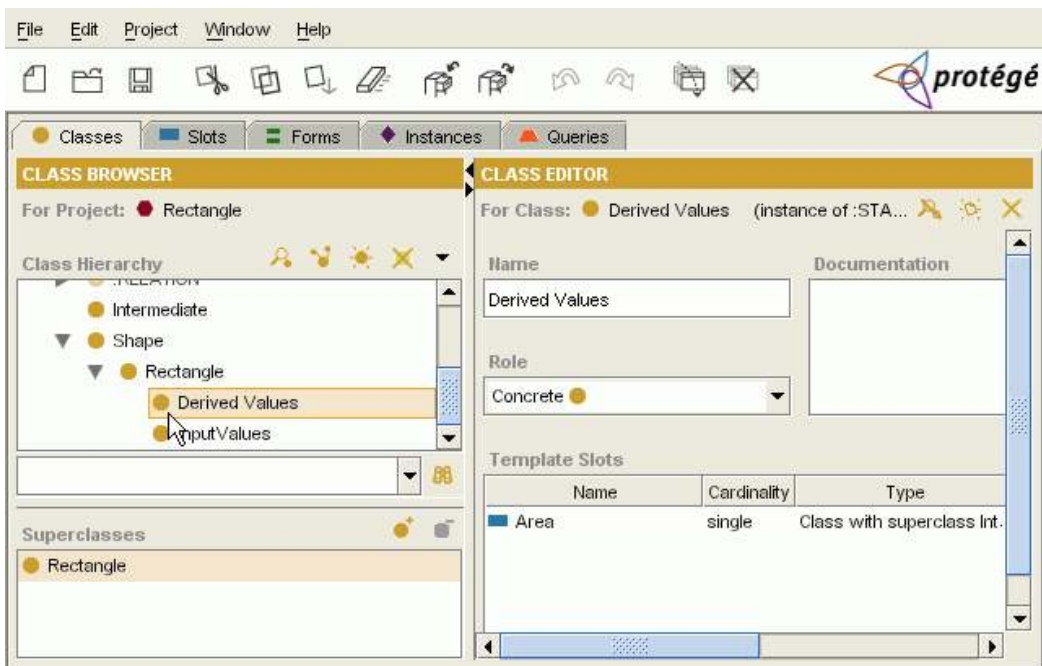


Figure 33 - Rectangle Explanation - Protégé 5

In Figure 34 'Area' is assigned a value of 'Length' * 'Width'. This is a simple equation that will be used to calculate the result.

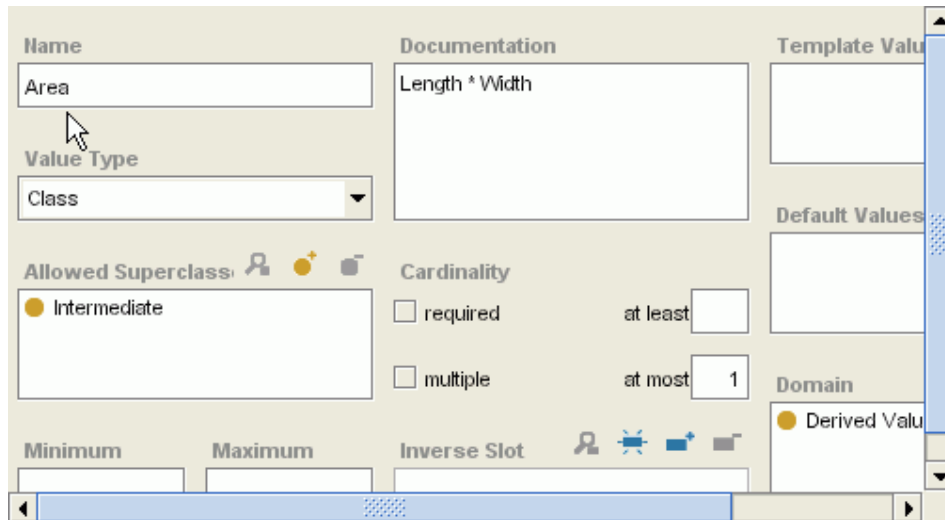


Figure 34 - Rectangle Explanation - Protégé 6

This illustrates how modelling calculations are performed. They are all defined by equations that relate attributes of the taxonomy. The taxonomy can be read by the decision support system.

Figure 35 shows the empty tree prior to importing the taxonomy (a translation process).

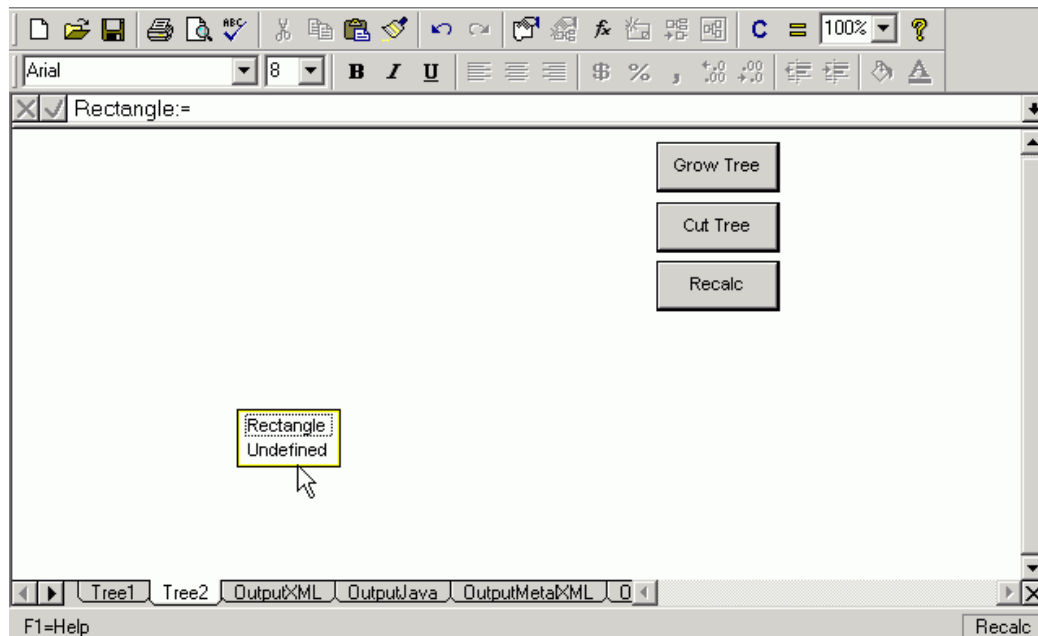


Figure 35 - Rectangle Explanation - Vanguard System 1

In Figure 36 the decision support system will read the taxonomy when the 'Grow Tree' button is pressed.

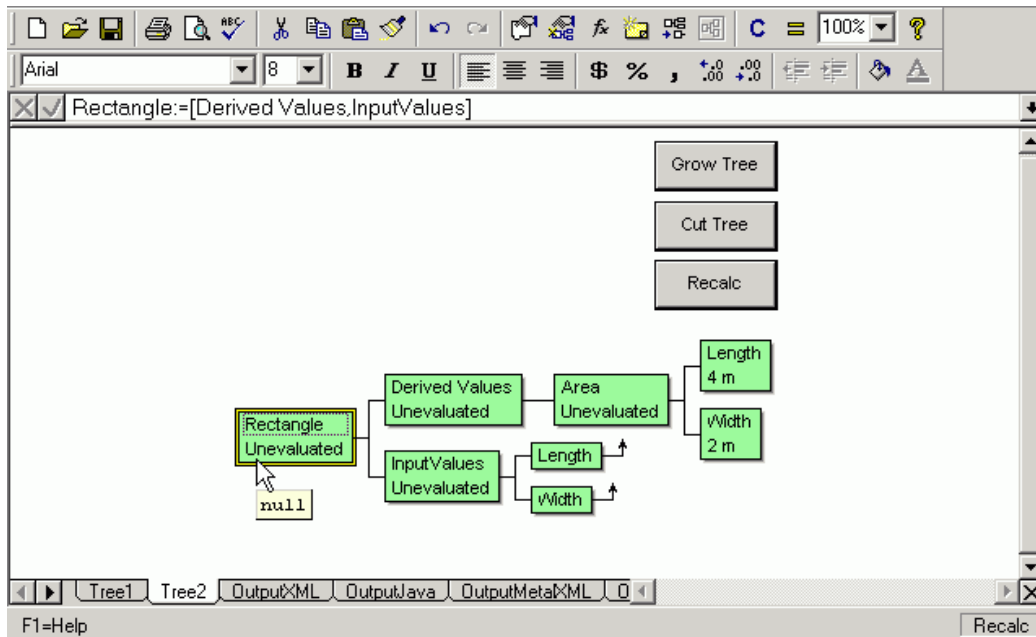


Figure 36 - Rectangle Explanation - Vanguard System 2

The taxonomy tree has been read in but the calculation has not been performed yet. The taxonomy was stored in a relational database using a system of key values that define all relationships and data types. This achieves the desired result of representing a hierarchical structure in a relational database. Generic code written using the Vanguard System editor reads from this database and recreates the tree.

The advantage of translating the taxonomy into a decision support system is its facility for performing calculations and statistical analysis. This allows the calculation to be made and visualised without any need for the user to create code. In Figure 37 the calculation uses the equation(s) defined in the Protégé editor that relate nodes in the taxonomy tree. Pressing the 'Recalc' button causes the calculation to be performed.

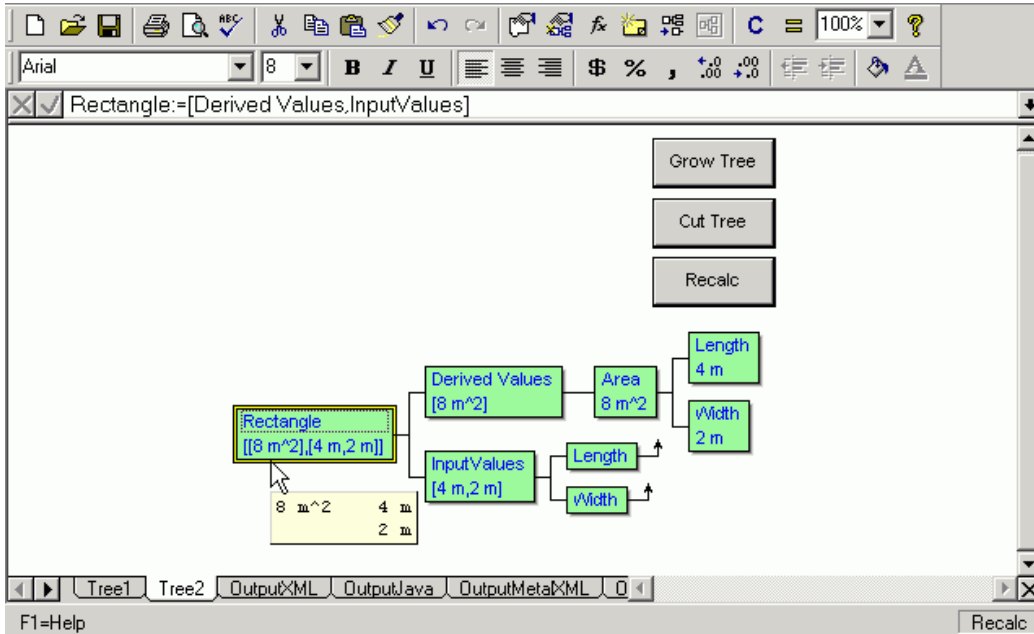


Figure 37 - Rectangle Explanation - Vanguard System 3

The result tree can then be output onto the Web, in W3C compliant languages.

XML (eXtensible Markup Language) and stylesheets are used to display this output in the browser, but any language or format could be provided for. Figure 38 shows the user deciding to output the tree as XML. The tree is then displayed in a browser, and it is possible to click on individual items to view them.

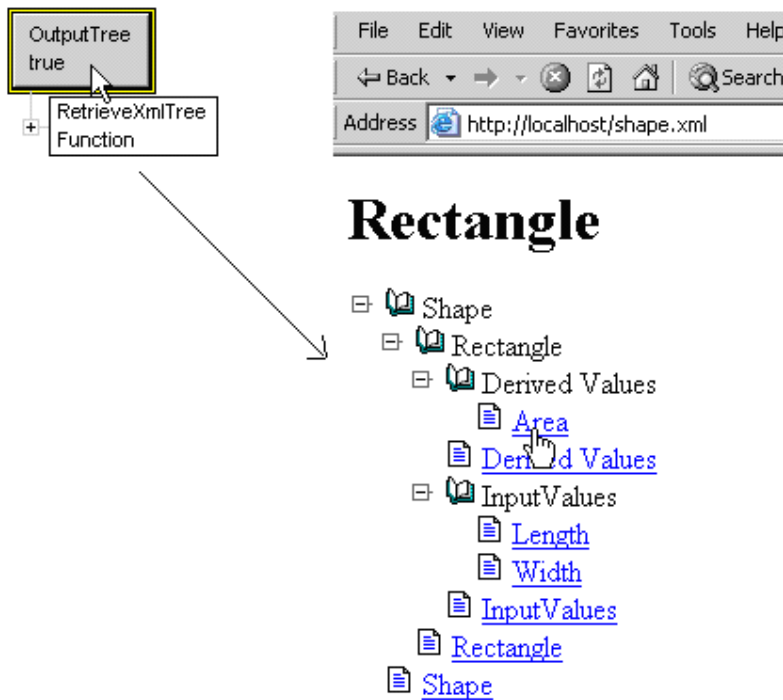


Figure 38 - Rectangle Explanation - Vanguard System 4

In Figure 39 the user clicks on an item and can view the calculation. The equation and result are shown on the web page that opens.

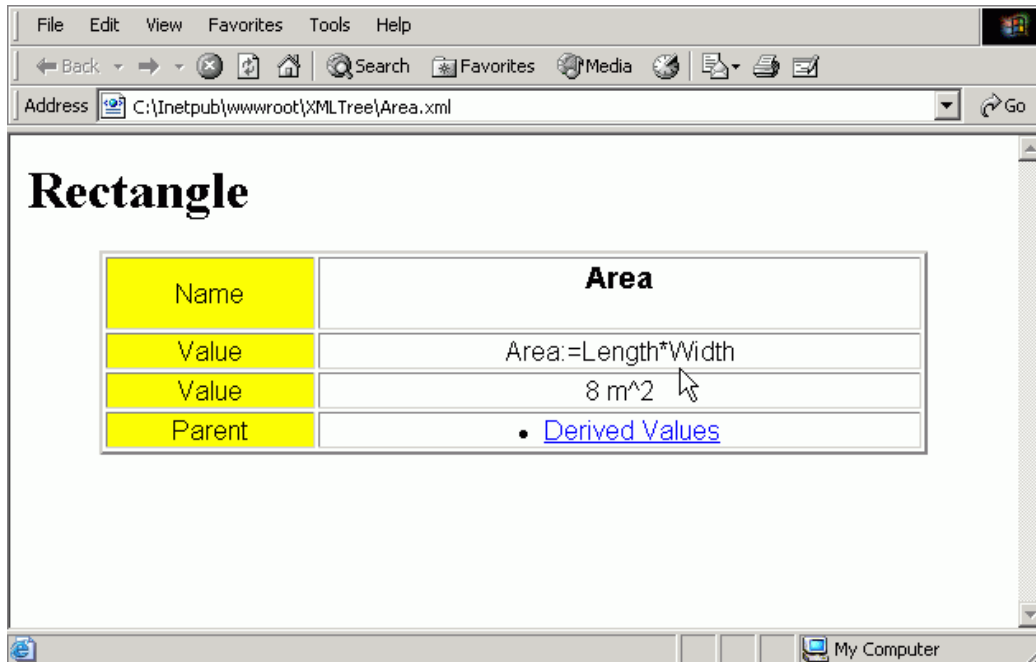


Figure 39 - Rectangle Explanation - Web View Tree 2

Sometimes an alternative is needed to the tree view of a taxonomy, such as a diagrammatic representation of the object. The alternative view is created using an automated transformation that converts the tree into an interactive diagram. This alternative can be displayed on the web page using SVG (Scalable Vector Graphics), an XML format. This visualisation enables interactivity, and allows for inputs to be changed dynamically.

Figure 40 shows an output SVG rectangle diagram that includes interactivity. This has been translated from the tree-based representation.

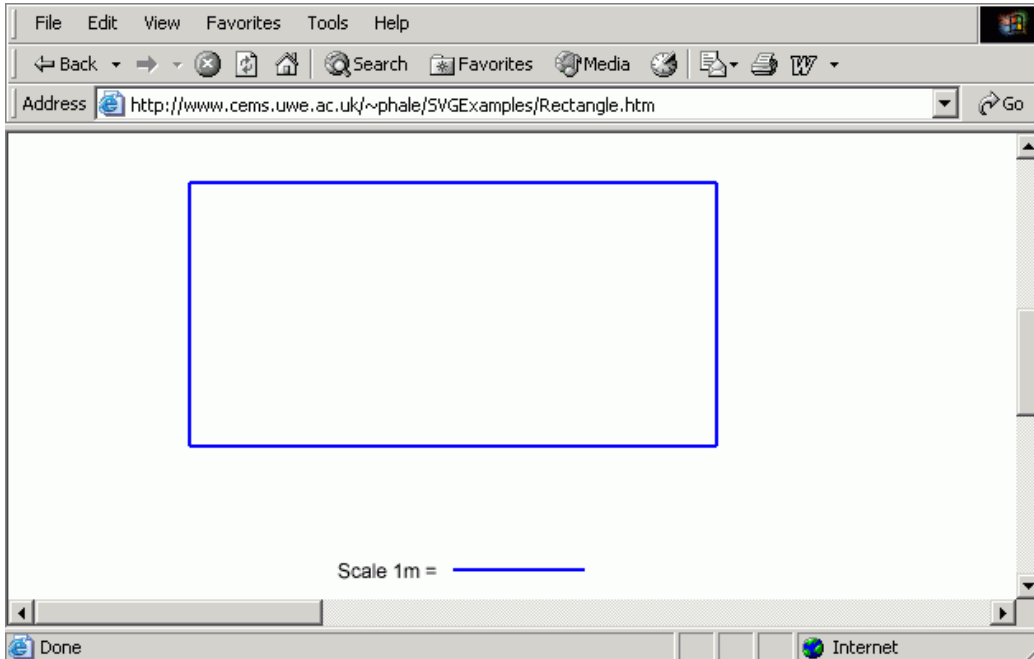


Figure 40 - Rectangle Explanation - Web View Diagram 1

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. This is shown in Figure 41.

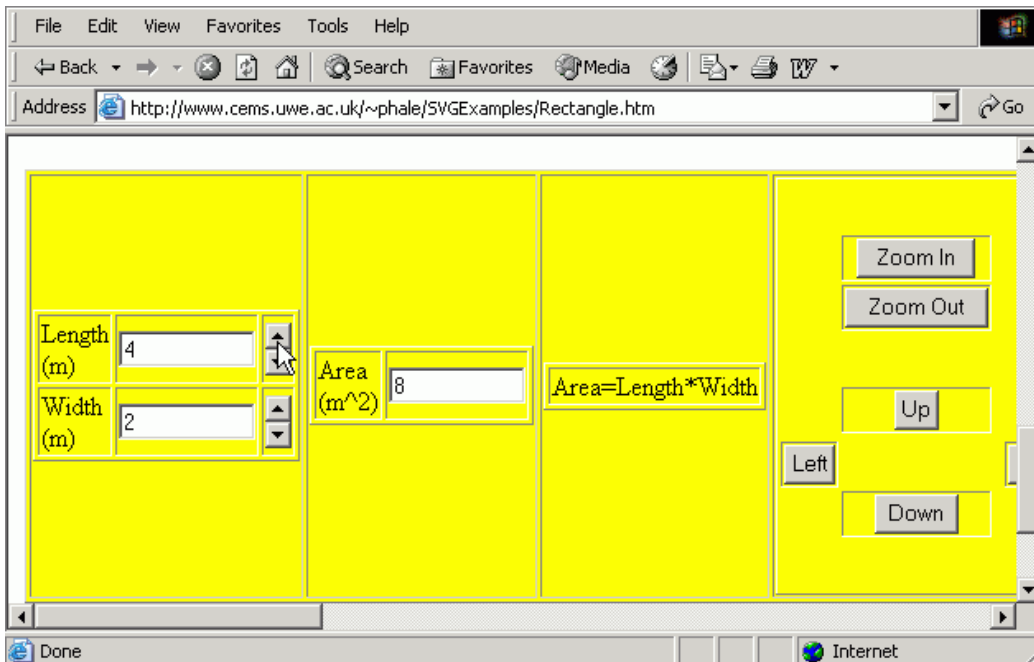


Figure 41 - Rectangle Explanation - Web View Diagram 2

The area and the shape of the diagram respond dynamically to any changes in the inputs such as holding down the up or down arrows to change the input values. In Figure 42 the user has held down arrow keys to alter the rectangle proportions.

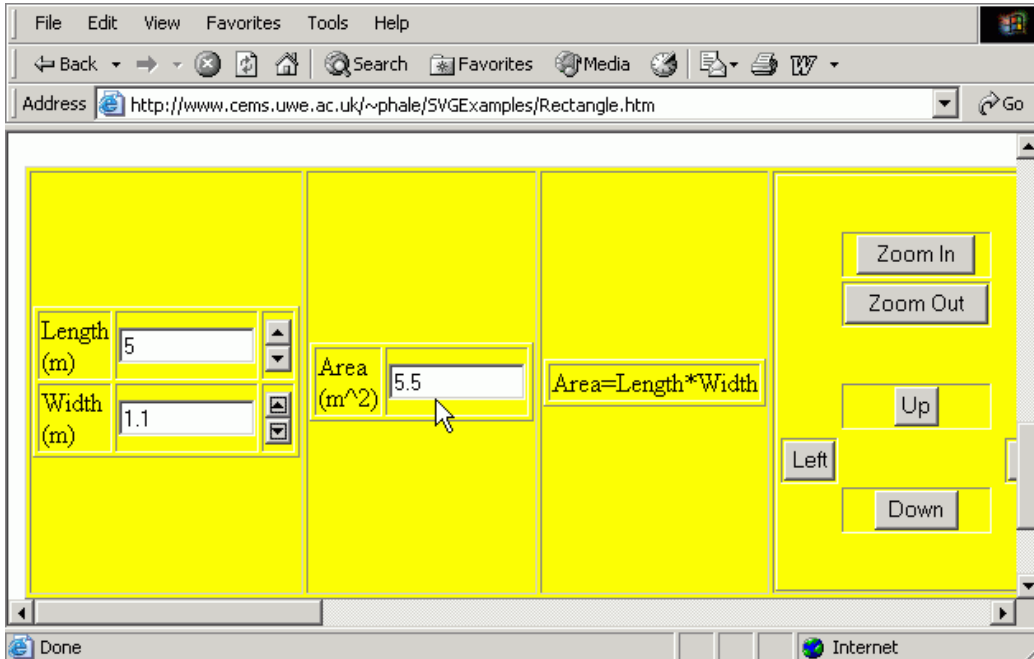


Figure 42 - Rectangle Explanation - Web View Diagram 3

Figure 42 showed that 'Area' has recalculated, and Figure 43 that the display has changed to match the change in shape.

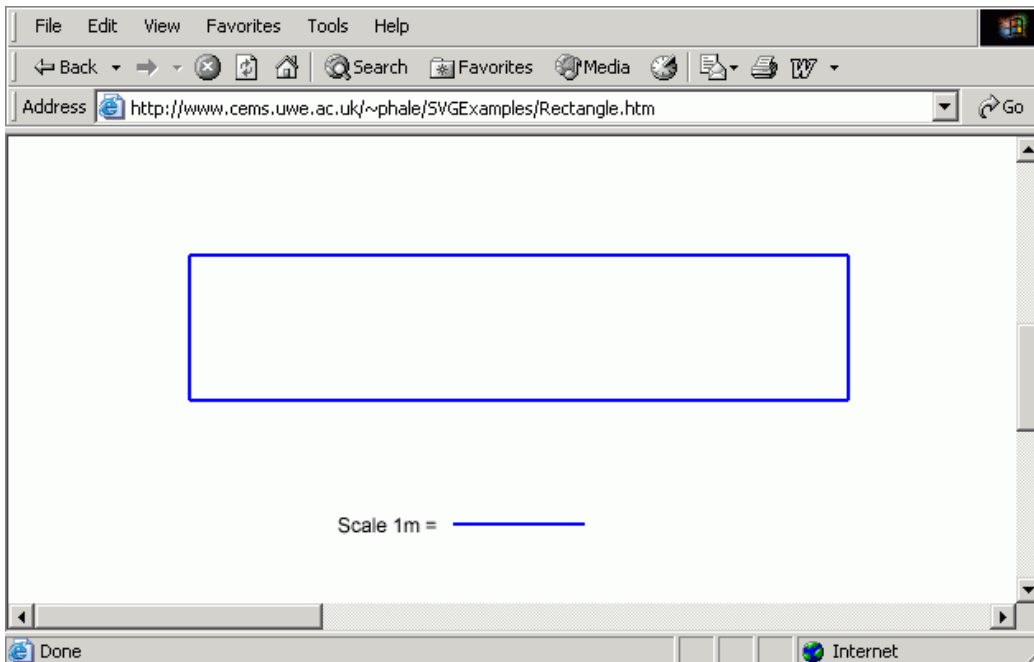


Figure 43 - Rectangle Explanation - Web View Diagram 4

Figure 44 shows other controls for moving and resizing the diagram.

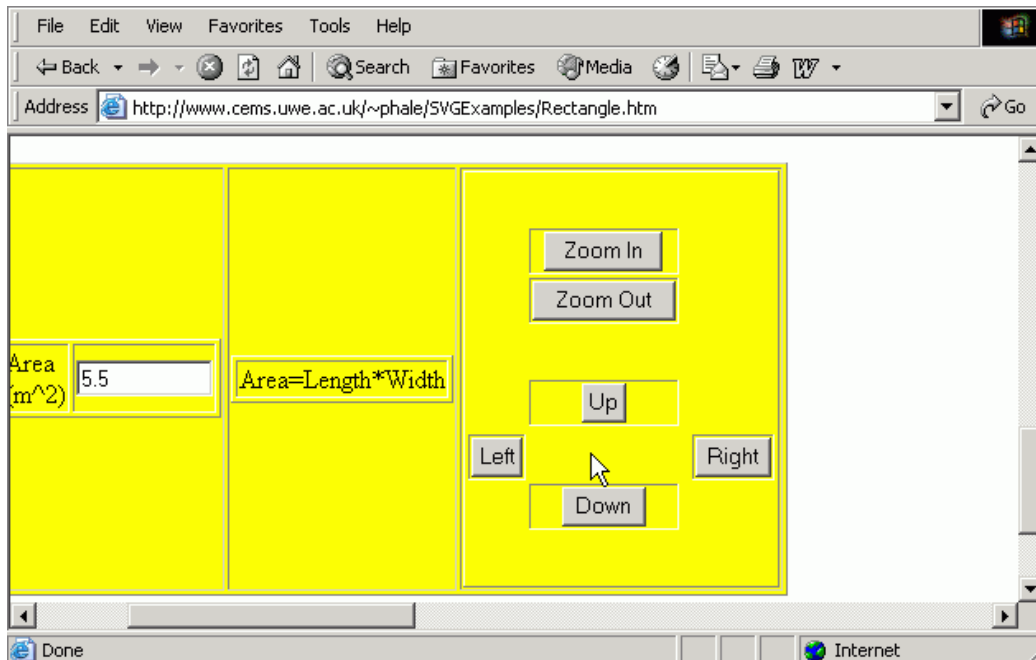


Figure 44 - Rectangle Explanation - Web View Diagram 5

The translation software can create an SVG representation that uses SVG plug-ins available for Internet Explorer, or create a native XML version that allows the Mozilla Firefox browser to render the SVG directly without the aid of the plug in.

Summary

This example shows that it is possible to create models visually without the user needing to write code. Calculations and translations are performed automatically. The code for performing the translations is generic. This means it is possible to translate the taxonomy representation into different programming languages, and visualise results in different ways as best suits individual users.

Chapter 11 - User Driven Modelling Techniques implemented with a complex example

User Driven Programming Examples

The example illustrated here is much more complex than that in chapter 10, the information is re-used from the ACCS (Aerospace Composite Costing System) spreadsheet described in chapter 3. The example in this chapter illustrates how the translation software can work efficiently with, and visualise very large and complex trees. The model visualised is one that contains thousands of nodes read in from the ontology and linked as required for the model. It will be possible to reproduce any model created in spreadsheets and any other legacy software systems in such a way, as well as creating a new model from scratch. This system has the potential to be used for any type of system modelling because it allows for the specification in Protégé or other ontology management systems of high level semantics, to express any problem. This specification is flexible enough to be re-used for solutions to different problems. This approach expands on earlier research on translating from the information held in a process planning tool (CAPPe) for Rolls-Royce, summarised in chapter 5. The main difference is that the translation link is to a Protégé based ontology rather than a relational database as used in the Rolls-Royce work. This ensures that the information is held in a standardised way, and so maximises re-use of the work. The ontology is created from taxonomies for 'Parts', 'Materials', 'Consumables', 'Rates', 'Processes', and Tooling. These taxonomies are read by the modelling tool, and any node in any of these taxonomies can be related to any other, in response to triggers from the user. Because the nodes translated from the ontology can be linked in many ways as required by the user, this multiplies the size of the model tree. The user begins with a default model, and can alter any equations, and these equations relate the nodes. So, the size of the model makes it impossible to visualise it as a whole due to screen space availability, instead the visualisation that allows users to navigate consists of views as called by the user. These views are represented in the colour coded diagrams illustrated in this chapter.

Implementation

Translation Process

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 information with colleagues. The information used in the models resides in an ontology, and from this ontology models can be automatically produced via a recursive translation tool. The research for this thesis uses a technique 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 other computer languages and systems. This is achieved by automated translation of the Vanguard System information into other languages. The basis of this is that elaborators are nodes in the tree, which are automatically created and dynamically write objects. This allows the wing box definition to be translated to the decision support system for costing and then to other software such as web pages for

further processing and visualisation. An open standard semantic editor Protégé created by Stanford University (2006b) was used to structure this information into related taxonomies. This ontology holds the definitions of nodes representing information, and calculations to be performed. Taxonomies were 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 can 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. Figure 45 illustrates how code is produced from the semantic relationships.

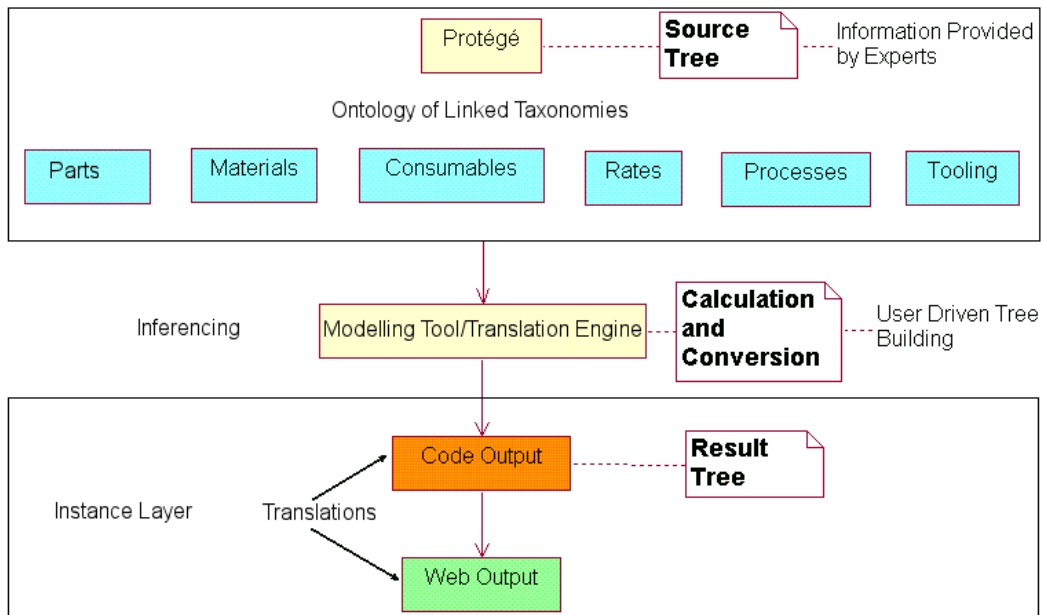


Figure 45 - Translation Process Implementation

This model 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 shown in Figure 46. 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. Vanguard's tree based decision support tool Vanguard System (2006b) reads this information and represents it as colour-coded nodes. The code written for this thesis automatically queries the taxonomies that make up the ontology and links the information as required for the model. The code builds in all the links required for the equations and thus links information from different taxonomies, the information is colour coded according to which taxonomy it is from. This same code can be reused for any 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 decision support tool can perform calculations and so output results. Figure 46 shows how the decision support tool can automatically construct and represent a branch in the tree, visualise an equation and calculate a result. 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 also 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.

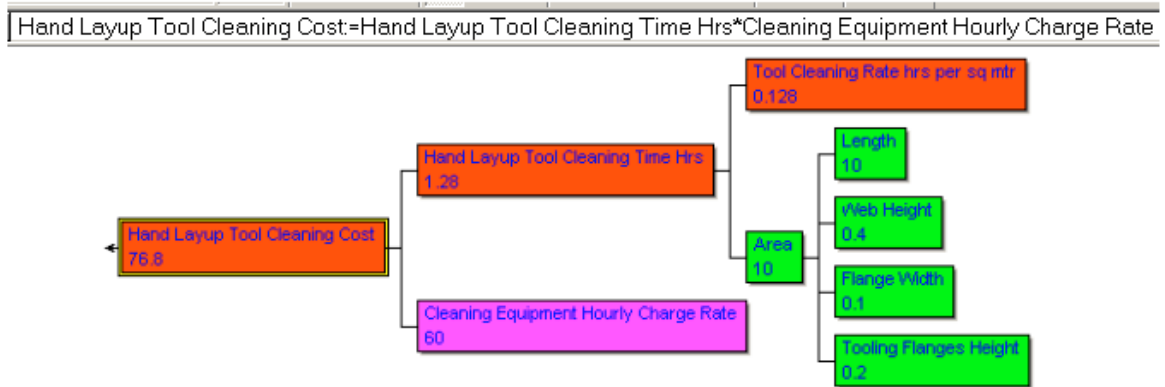


Figure 46 - Ontology to Model Conversion

For the prototype to be extended and applied for practical use, each taxonomy is 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.

Implementation Example – Spar Hand Lay-Up Process

Figure 47 shows a simple diagram of a spar, the spar is a component whose details have been entered in the ontology.

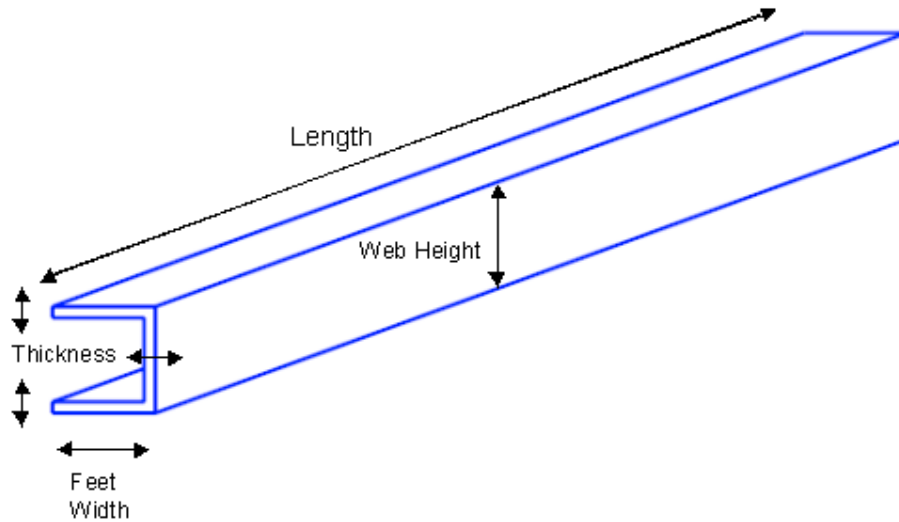


Figure 47 - Spar Diagram

Figure 48 illustrates how the decision support system tree view of the spar branch from the wing box cost model is created with information translated from the related taxonomies, and where necessary from user's 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.

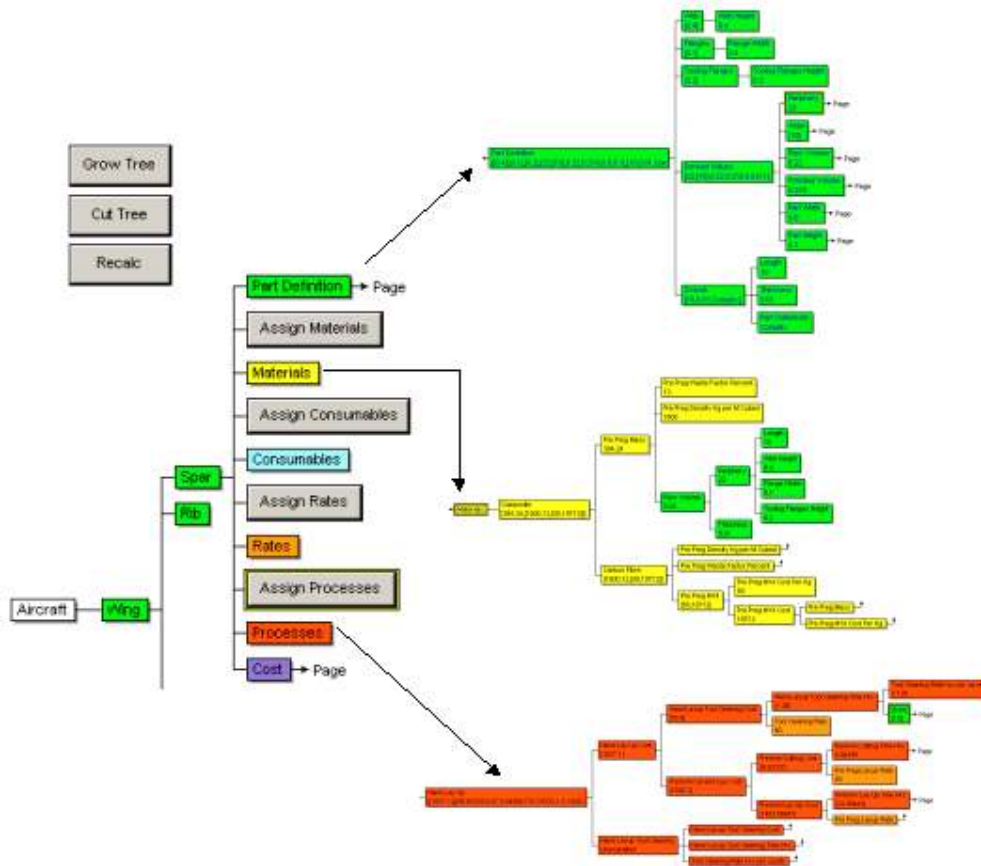


Figure 48 - Spar branch automatically created from information source

The user makes choices so the decision support result tree will be a subset of the information source tree.

Vanguard System visualises large trees by breaking them into individual pages, 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 49. The 'Derived Values' branch contains parameters of the spar that are calculated from the spar dimensions.

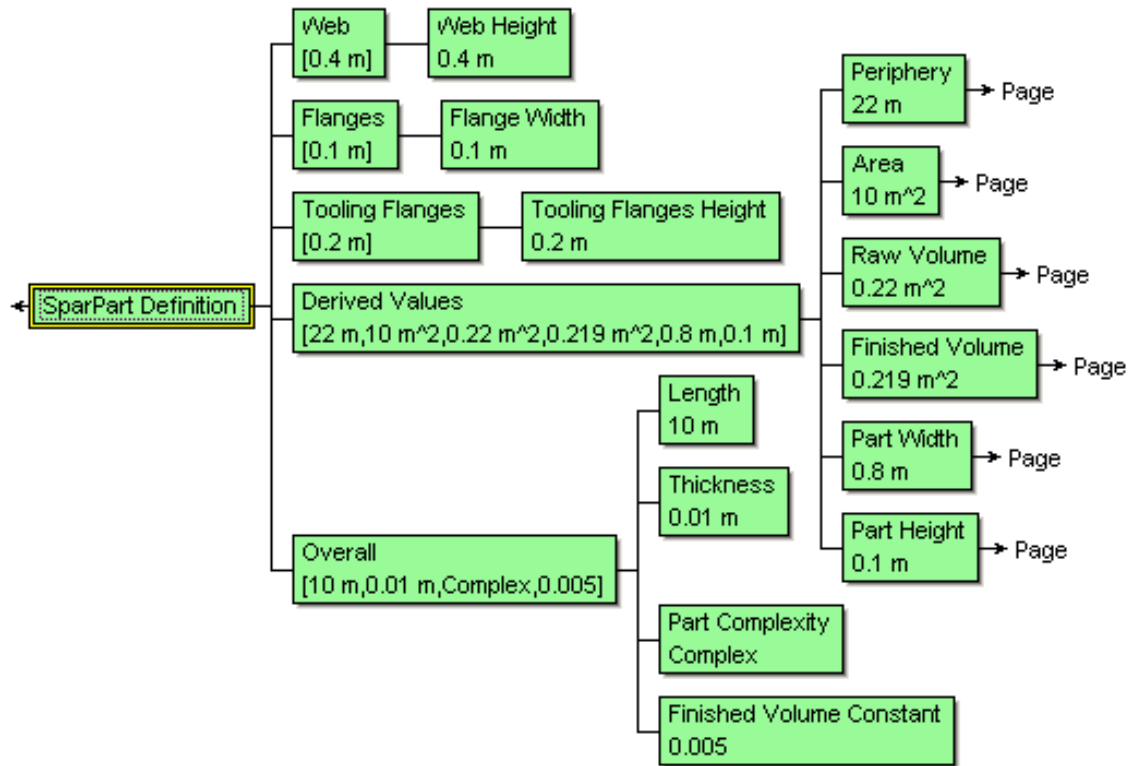
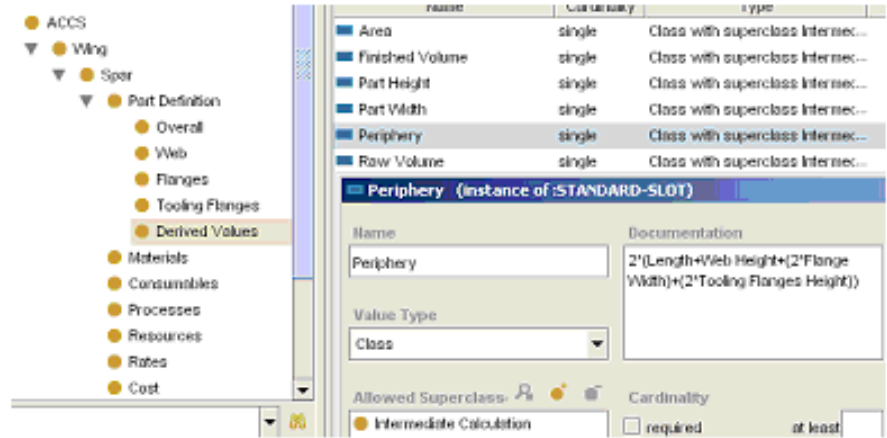


Figure 49 - Part Definition Branch

Figure 49 is a Vanguard System reproduction of the part definition from the Protégé taxonomy. 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 seen in Figure 50.



$$\text{Periphery} = 2 * (\text{Length} + \text{Web Height} + (2 * \text{Flange Width}) + (2 * \text{Tooling Flanges Height}))$$

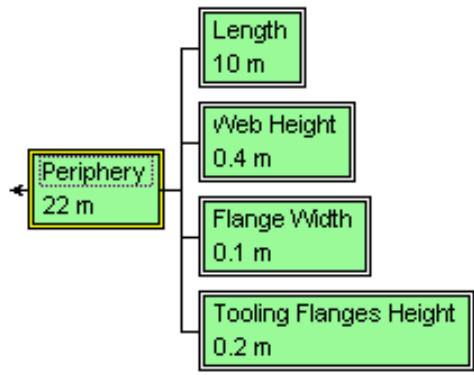
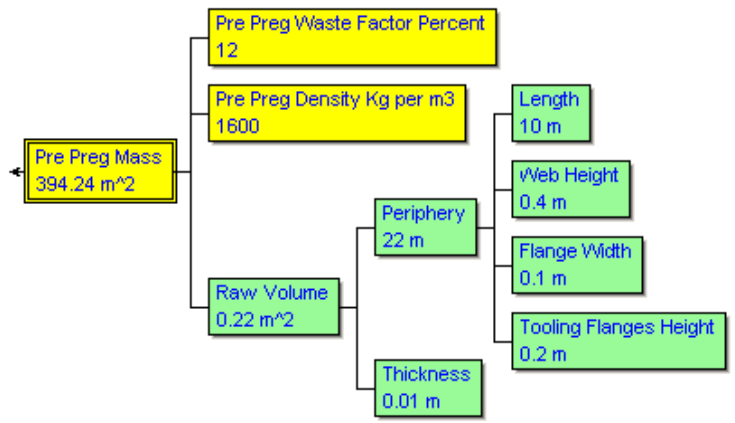


Figure 50 - Translation of Equation Representation from Protégé to Vanguard System

Different types of information indicated by colour coding may be combined in a calculation. This is illustrated in Figure 51. '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.

$$\text{Pre Preg Mass} = (1 + (\text{Pre Preg Waste Factor Percent} / 100)) * (\text{Pre Preg Density Kg per m}^3 * \text{Raw Volume})$$



This approach to colour coding, of illustrating type with colour, is related to this work on using colour to indicate cost, time, or uncertainty (Bru et al, 2002), (Bru et al, 2003), (Bru et al, 2004). The need for tools such as developed in this thesis, and those of Bru, was made apparent when models such as the wing box spreadsheet started delivering volumes of information, which could not be easily absorbed and analysed in their textual form. The information visualisation 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 54, 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. The following interactive web page examples further illustrate this point. This interactive web page can be found at (Hale, 2007q).

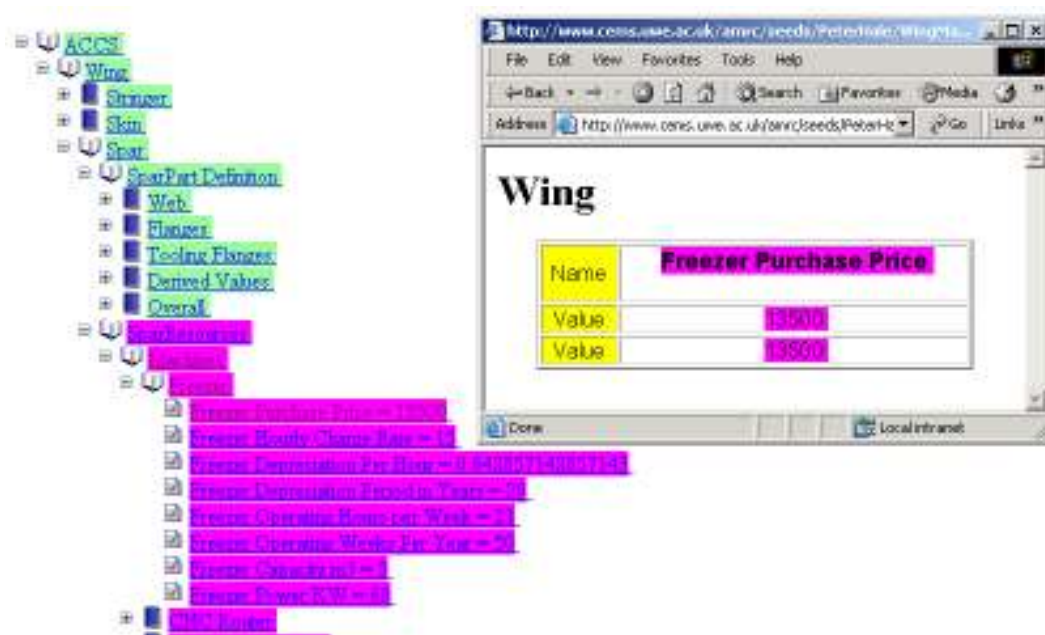


Figure 54 - Web page showing translated XML displayed as interactive web application

Web Output

Creation of web pages for Rolls-Royce led to the view that this could be a popular method of delivery for information and models. Web output is important as an alternative to spreadsheets because of the problems with spreadsheets explained in chapter 3, and because web standards provide a multi-user, standards based, freely accessible, method for conveying information and models. Google Spreadsheets (Google, 2007) was evaluated for this purpose and online spreadsheet tools including (Softartisans, 2007) and Java based spreadsheets. It turned out to be better to use 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. Research in this thesis into providing web-based models is partly based on previous Rolls-Royce work for providing parametric models online. It is important to separate the data from the implementation so the data can be re-used by other software that exists now, or will be created in the future. This also ensures that if the data changes the models will update automatically, and

models can be added just by adding information to the database or ontology. As the data can be changed the model author does not need programming language skills. As in the example above this represents progress towards the goal of allowing user driven modelling. Although these are very simple parametric models they do represent a further advance on earlier work (chapter 7) as the equations can be edited in XML files as well as information. Figure 55 shows an implementation of a simple XML based parametric model. This was created using JavaScript code that will take any equation provided in the XML data source and calculate the result, it uses an XSL (eXtensible Stylesheet Language) stylesheet for formatting and display. Other implementations were created, one using Microsoft MSXML control and another using ASP (Active Server Page) code. These examples were created with Christophe Bru (Hale, 2003), (Bru et al, 2004) and used for this thesis. Examples of this can be accessed at (Hale, 2007g).

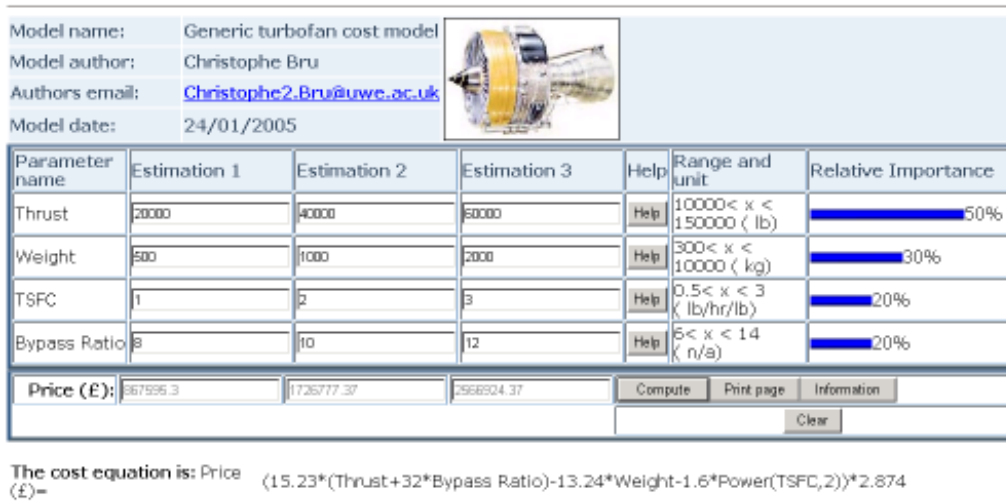


Figure 55 - Parametric Cost Estimation

In the above example the model author can edit the data source without changing the program code. This is because provision of the modelling environment makes any further programming unnecessary. This meant the code could be used for development of this thesis and immediate implementation at Rolls-Royce.

It is possible to output text files so decision trees can be translated into other programming, meta-programming or structured languages. This was achieved by writing generic recursive code that accesses each node and translates this to other language representations. This enables provision of a visual web interface to the models without having to be locked into proprietary solutions, and ensures the maximum amount of access for users. The production of part diagrams using SVG can be automated in a similar manner to that used for the automated production of Vanguard System costing models. Figure 56 shows an example of such an interactive visualisation of a Spar. This interactive diagram was produced from the part definition automatically as defined in the part taxonomy. This is a prototype of translation techniques used in the thesis. 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 user's tasks. They illustrate programming by demonstration, where the user visualises a problem by drawing a diagram of it. As explained in chapter 7, this is an 'analogical' representation of the problem because it shows the visual representation of it, rather than a 'fregean' representation that just shows values in text form. The importance of this is as a mechanism for user interaction. Figure 56 has been created from a tree based 'analogical' representation and could also be translated back to this tree form, and to a 'fregean' representation that is required for computer code.

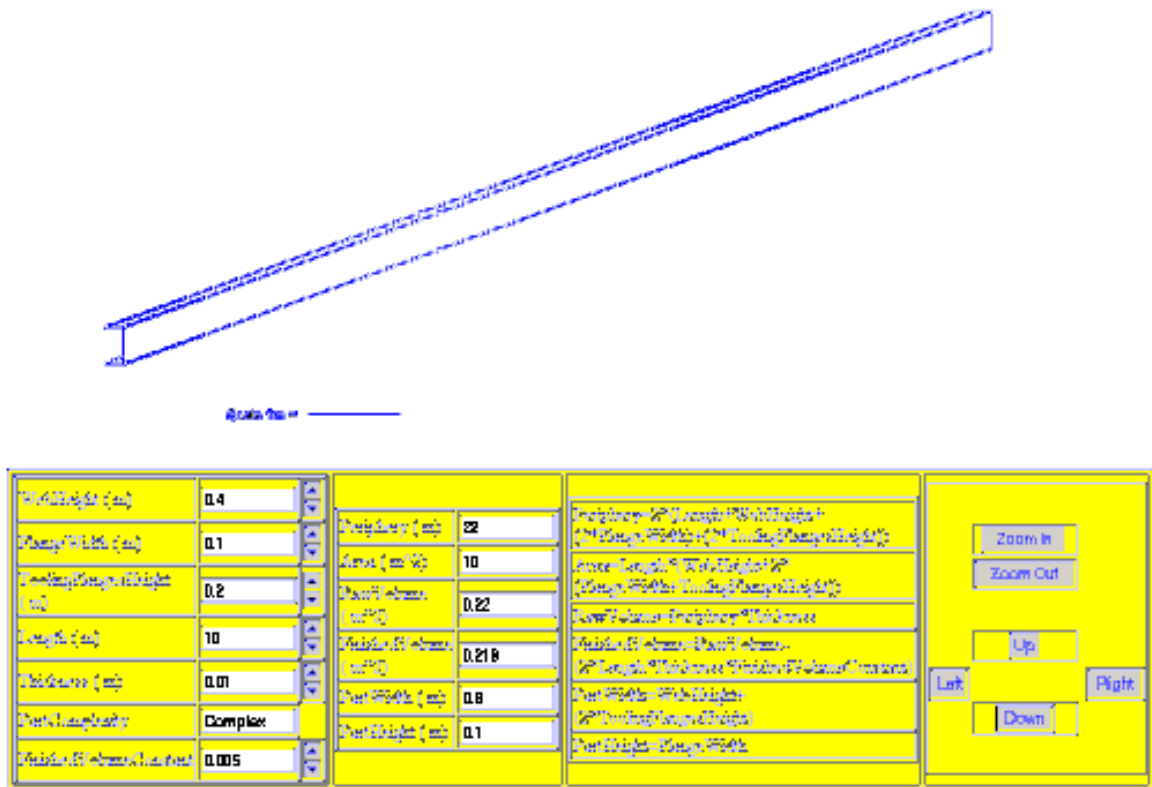


Figure 56 - Interactive Spar Diagram (SVG)

This example was produced via an automated conversion from a tree representation of this 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 will change 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. Coutaz (2007) 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. The way this interacts with the user can be seen by viewing these examples at (Hale, 2007d). These are examples of wing box components. The Internet Explorer examples require an SVG plug in, which can be downloaded. The Mozilla Firefox examples are produced using a native XML implementation of SVG and so do not need this plug in and are created with an SVG renderer, so SVG contents appear as an integral part of the document instead of an embedded object. Both sets of examples are produced automatically using

an intermediate tree that draws the component outlines dynamically from the design parameter values. Once defined, a component or a feature is held in a library and re-used. SVG developers should introduce this automated construction and storage of SVG diagrams, and so make this available to those who do not have access to CAD software. This could allow 3D modelling and collaboration over the web as envisaged by Fishwick and Miller (2004), and Park and Fishwick (2005). Mohamed and Celik (2002) used a similar technique when they created a system to map classes to a schematic drawing of a building, and allow for online editing of a table of values for this building. This assists designers to see the effect of their design on attributes that could affect cost and scheduling.

Figure 57 demonstrates the Vanguard System tree translated into XML and displayed as a web tree using a stylesheet created by De Andreis (2005).

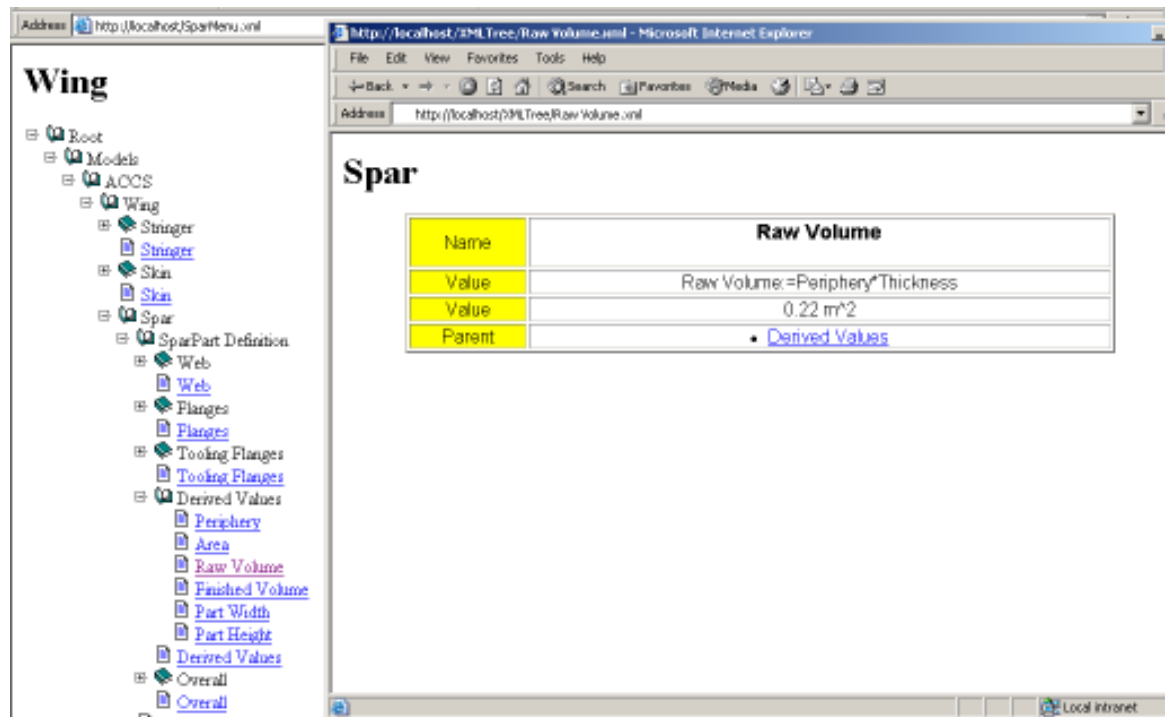


Figure 57 - XML web translation from Decision tree

An example of use of this stylesheet is available at (Hale, 2007).

XML can also be displayed on the web using a Flash program created by Rhodes et al (2002). This creates a tree with a three dimensional look and use of colour, shading, and movement of the 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. This interface is illustrated in Figure 58.

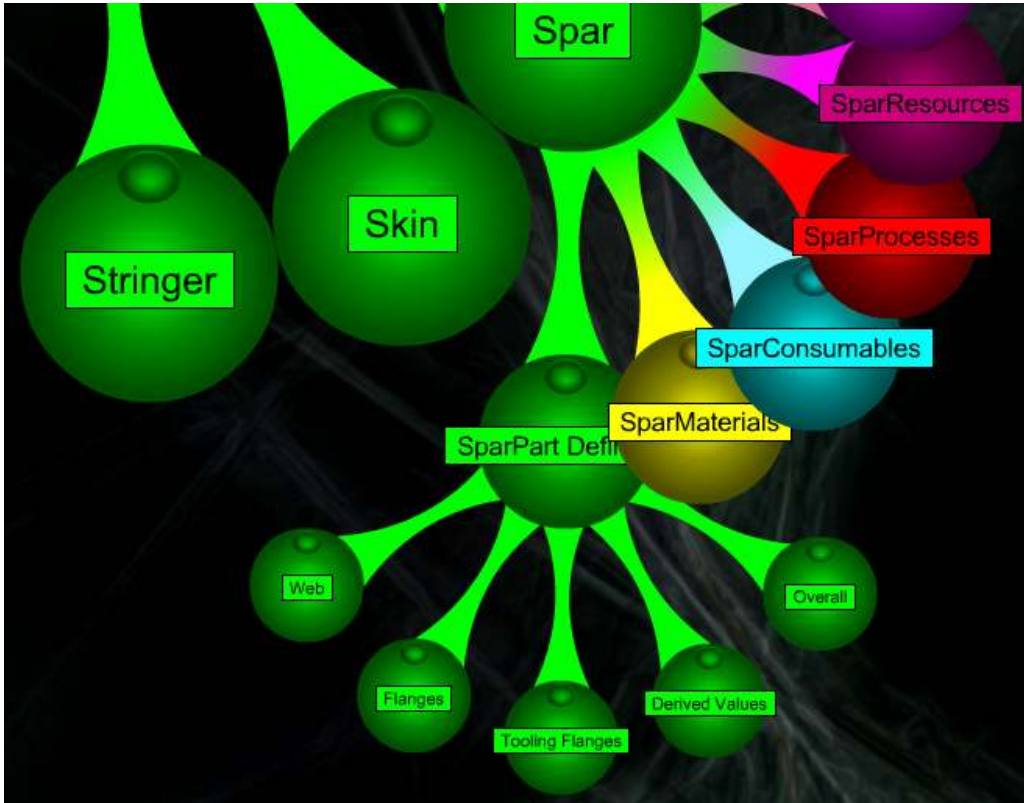


Figure 58 - Flash interface for navigating exported XML tree

Figure 59 shows the view resulting from choosing the 'Spar Part Definition'. This shows the parents, children, siblings, and contents of that node. It also allows navigation to any of the related nodes.



Figure 59 - Flash viewing of Spar Part Definition node

This example is on the research website (Hale, 2007c).

Figure 60 shows the Vanguard System tree translated into Java and visualised.

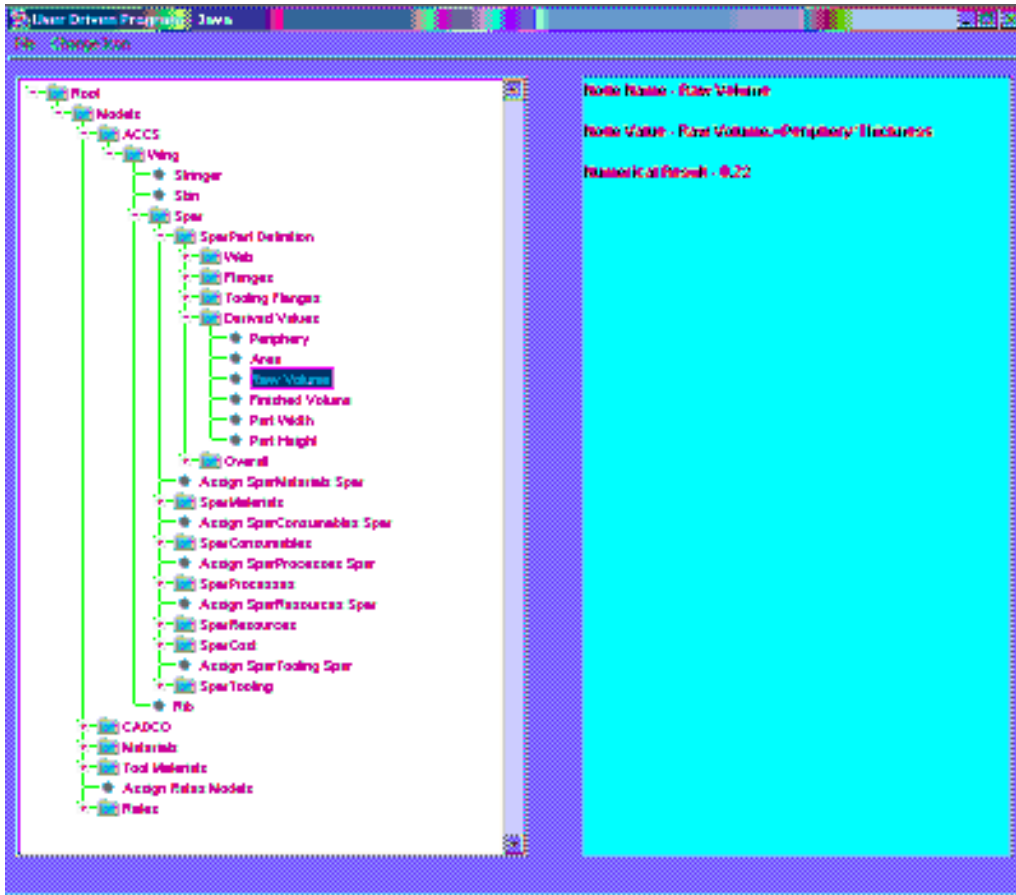


Figure 60 - Translation from decision tree into Java

This can also be created and visualised as a Java applet (Hale, 2007p), Figure 61.

Automatically created Spar

- ☐ Root
 - ☐ Web
 - ☐ Flanges
 - ☐ Tooling Flanges
 - ☐ Derived Values
 - ☐ Periphery
 - ☐ Area
 - ☐ Raw Volume
 - ☐ Finished Volume
 - ☐ Part Width
 - ☐ Part Height
 - ☐ Overall

Node Name
Raw Volume

Node Value
Raw Volume:=Periphery*Thickness

Numerical Result
0.22

Figure 61 - Translation from decision tree into Java Applet

A translation into the Java based Cost Estimator System (Koonce et al, 2000) (Wujek et al, 2002) was also created. Figure 62 shows this.

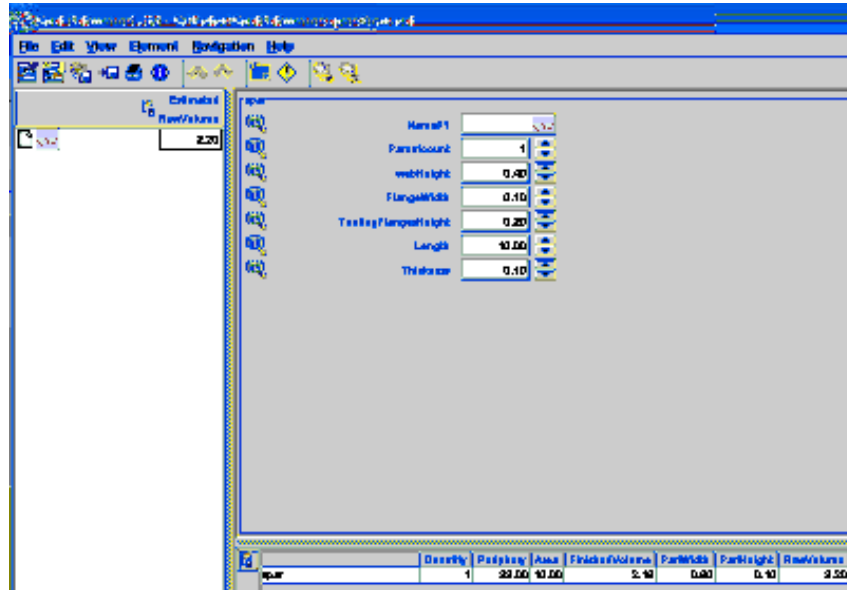


Figure 62 - Translation from decision tree to Cost Estimator

Semantic Search

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 63 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 Exact Match:

Components

Query - Like '*Spar*'

[Result Tree](#)

Figure 63 - 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 64 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 64 - Results from semantic search

The information is held in a taxonomy 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. XML files can be queried using XQuery a W3C working draft (World Wide Web Consortium, 2006i). RDF files can be queried using SPARQL a W3C specification, for which a demonstration is available at Dodds (2007b) and a tutorial based on this has been developed by Dodds (2007a). 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". A web interface has been developed for Protégé (WebProtege, 2007). An example of the

use of this is Figure 65 where a search is made for information on the cure cycle for composites manufacturing.

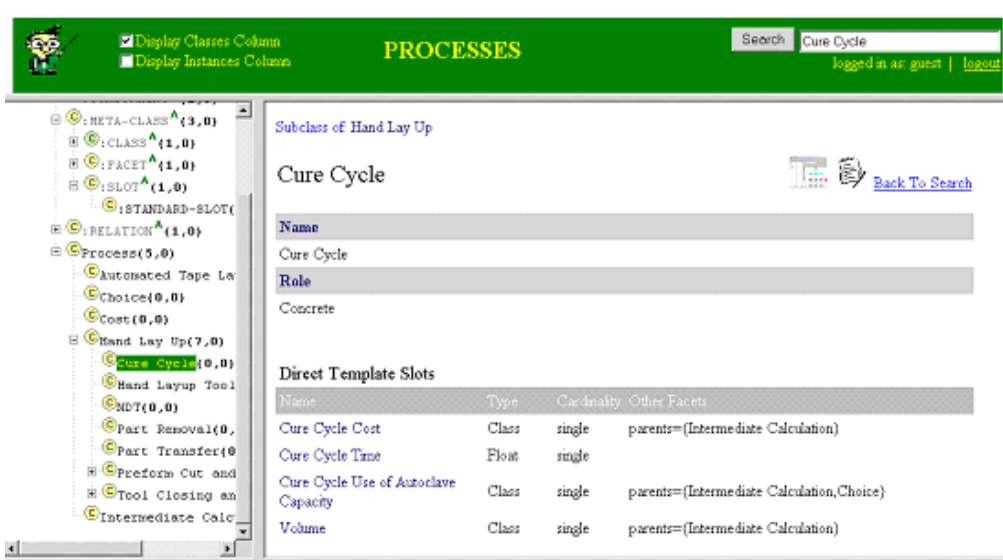


Figure 65 - WebProtege Searching - Cure Cycle

Lee et al (2000) present a distributed visual reasoning system for intelligent information retrieval on the Web. The user can design a query by linking active icons, and then inputting the required parameters. The user can then see the structure of the query and obtain results from the database.

Semantic Web languages can be a useful programming tool, and be used for creation and editing of E-Learning objects (Stutt and Motta, 2004). For enabling user driven programming, it was necessary to experiment with using Semantic Web languages as programming languages. The two main approaches used in the previous examples were:-

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.

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. 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 into JavaScript or Java. This is different from the AJAX (Asynchronous JavaScript and XML) approach, which is more like option 1.

The advantage of this (option 2) form of declarative programming are 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 can release their solution over the web.

Technologies such as XForms, XQuery, and SPARQL make it possible to provide the sort of collaborative interactivity that Berners-Lee calls 'Intercreativity' in 'Weaving the Web' (Berners-Lee, 1999). He also discussed the use of Semantic Web languages as programming languages. He makes 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. Berners-Lee sums up the advantage of a Semantic Web program over programs in other languages. He writes "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." If these rules are also visualised, they are exposed to everyone, including non-programmers. Begel (2007) explains 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.

These advances make it practical to develop a high level visual interface that can allow people to develop open source, open standard, interoperable programs and share them. This can allow the development of open source communities similar to those developing software currently, but only requiring the level of skill it takes to get started in visual collaboration tools such as MySpace. In Weaving the Web Berners-Lee writes "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.

Chapter 12 - Further Research

This chapter outlines future research that is required for the advancement of representation, search, and visualisation of information, and at recent and future developments in the use and representation of taxonomies and ontologies, and visualisation tools that can aid in their use. Berners-Lee et al (2006) explain 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."

Objectives for future development of Ontologies

Horrocks (2002) explains the advantages of moving towards a more formal ontology. 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 are meaningful to people as well as being searchable by computer software. The intention is to enable tagging of this ontology and eventually editing of it by users, in order to allow users and domain experts to be involved in the ontology construction.

So far the taxonomies used in this thesis include traditional object oriented relationships such as child, parent, sibling, attribute, and instance. There are other types of relationship that would need to be modelled in order to maximise the capabilities of software that would use the taxonomies. Key relationships used within the object oriented programming domain between objects have been modelled. These key relationships depict families and aggregations of 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 allow a more expressive depiction of a problem domain, and can aid some forms of search within a model. One of the main advantages of a semantic net description, in terms of automated model generation, is that labelling relationships between objects allows the depiction of a number of aspects of a domain in one model, and with a consistent syntax. Ciocoiu et al (2000) 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. More expressive semantic descriptions are possible through the use of the standard OWL dialects. These more expressive descriptions require sophisticated visualisation tools.

Ontology Visualisation and Interaction

As mentioned in chapter 5, Protégé 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) (2006) has developed Jambalaya (Ernst et al, 2003) for visualisation of knowledge and relationships. Ernst et al 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. Researchers at the University of Queensland Australia have developed a hyperbolic browser to display RDF files, this is explained in Eklund et al (2002). Cheung et al (2005) provide an ontology editor for knowledge sharing in manufacturing.

It is also important not to stay limited on one ontology development environment but instead explore how ontologies can be developed using a range of development tools and translated between each where necessary (Garcia-Castro and Gomez-Perez, 2006) are testing this as explained in chapter 5. For this reason, a large range of ontology management tools have been investigated for this thesis, these were explained in chapter 5, and meta languages were explained in chapter 6. 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), Horrocks et al (2003), and Zhang (2005) explain the SWRL language. Horrocks et al talk of defining properties as general rules over other properties and of defining operations on datatypes, within the thesis this research could assist in providing a visual rule and equation editor. 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 (Miller and Baramidze, 2005) will assist with the construction of a modelling system with sophisticated editing of rules.

Modelling and Simulation Objectives

A future task to be undertaken would be the inclusion of uncertainty in the automatically produced models, for situations where accurate information cannot be provided for the model. This would require provision of a way of handling uncertainty for parameters within the ontology, e.g. as 3 values describing a triangular distribution rather than a unique absolute value. The decision support meta-program could be expanded to write out the code to run Monte-Carlo sampling, hence making use of the statistical uncertainty capability, Sweeting (2000) explains Monte-Carlo simulation.

Miller and Baramidze (2005) examine efforts to develop mathematical semantic representations above the syntactical representations of MathML. This should enable standardisation of the representation of mathematical expressions that relate nodes, and their values and expressions. 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. It would be very useful to create 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. 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." As end-users are 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 will be necessary to research the interface between Meta-Programming, Modelling and Simulation, and Semantic Web Model Creation, shaded in Figure 66. This could allow end-users to develop their own Semantic Web based simulation and modelling tools using a graphical visual interface.

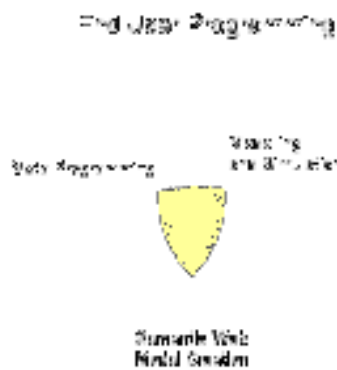


Figure 66 - Future Research Area

The thesis research has involved creating an OWL (Web Ontology Language) ontology to enable users to specify parameters in diagrammatic form. It could be possible to extend the semantics used in the specification of models to allow the creation of a framework for simulations. Because the ontology uses open standards, these simulations could be made broadly available on the web. It is important that the necessary infrastructure is created to allow this facility to be added. The approaches of others to this problem have been examined. Page (1998), Page et al (2000) and Page and Opper (2000) examine the nature of web-based simulations. 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. Fishwick and Miller (2004) examine the use of ontologies for modelling and simulation. The authors were involved in the RUBE project that developed a system for battle simulations. The RUBE project uses open standards and Protégé for the ontology, and outputs some code automatically. Kuljis and Paul (2001) evaluate progress in this 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. Kim et al (2002) explain how techniques of generating executable code from documents specified in standardised XML can be used to create simulations. Simulations could also be used for optimization, Chen and Yücesan

(2001) investigate this. The use of process models can allow accurate manufacturing times to be generated. This requires dynamic models of factories, cells and processes. Future research for this thesis will involve developing a web-based simulation building framework, e.g. for determining a process plan and monitoring resource and time constraints. Also it is necessary for users of a system to be able to gather information from various computer systems such as databases and spreadsheets. There is a conflict between the aim to develop an ideal representation of knowledge using an ontology editor, and the practical need to get the data from the database or application it is currently held in. The research undertaken in this thesis prototyped methodologies for creating, searching, editing, and interacting with information. Other researchers such as Aragonés et al (2006) and Crapo et al (2000) and (2002), and all the researchers mentioned in Chapter 4 and 5 have also investigated this problem. Software created for this thesis could be combined with software that others have created into an overall system, such as that below. This system would receive information from other software environments (left side) and from users (right side). This is shown in Figure 67.

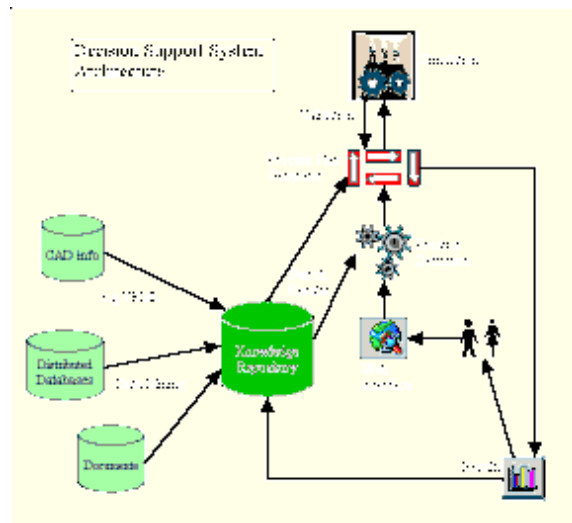


Figure 67 - Software System for Modelling

While chapter 4 and 5 explained the important research of others into knowledge representation, this applies mainly to the left side of this problem (although many also examined user interface problems, on the right side, including Aragonés et al (2006) and Crapo et al (2000) and (2002)). The right side of the diagram draws more on the work of researchers in Chapter 2 'History of End-user Programming' as it requires sophisticated modelling and interaction capabilities while still not expecting programming knowledge from the users. Shim et al (2006) also discuss user interface issues for this kind of problem, they investigate techniques for "powerful, yet simple user interface designs that enable interactive queries, reporting, and graphing functions". They also examine end-user computing history, and explain "The evolution of the human-computer interface is the evolution of computing. The graphical user interface (GUI) that was refined at Xerox, popularized by Macintosh, and later incorporated into Windows".

Visualisation and Interactivity

The next stage of research beyond the thesis will be to develop the example models towards use in public communities for application in industrial design and production problems such as efficient energy use and aircraft design and production, and also for use in education. Further research will broaden the end-user development techniques demonstrated in this thesis, to areas that are outside modelling, but link into this. The research could be applied to e-learning and this would enable the use of web-based distributed constructionist projects in teaching (Resnick, 1996) and (Stutt and Motta, 2004). Visualisation would be applied to the shaded area in Figure 68, where research in E-Learning would involve End-user programming enabled with Semantic Web technologies and with a visualisation and interactivity layer.

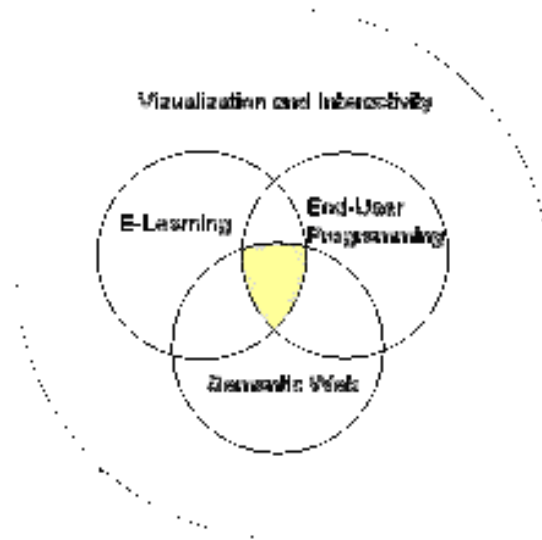


Figure 68 - E-Learning and End-user Programming

Recent developments in the use of Meta languages for platform independence should make the development of end-user programming quicker and easier. Bishop (2006) 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." The author investigates, evaluates and advocates the use of platform independent programming languages. Further research will involve enabling drag and drop from the previously created web-based interactive interface. Repenning (2007) argues that "Visual programming languages using drag and drop mechanisms as the programming approach make 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." Rosson (2007) also explains about creation of a web-based drag and drop interface. Further research after this thesis will involve investigating the use of Meta languages such as Simkin (Whiteside, 2006) for modelling and virtual reality, OpenLaszlo (2007) for web

applications, mentioned in (Bishop, 2006), and XForms (Bruchez, 2006) for the interface to modelling tools and other applications that require user interaction. . Jackiw and Finzer (1993) describe an example where a diagram is translated to a graph 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'. The thesis research 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.

User Driven Modelling Solutions

Figure 69 illustrates the wider solution to be created, now the problems that need solving for user driven programming have been investigated, and 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.

Method

- The Translator (Figure 69) will automatically create 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.
- 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.

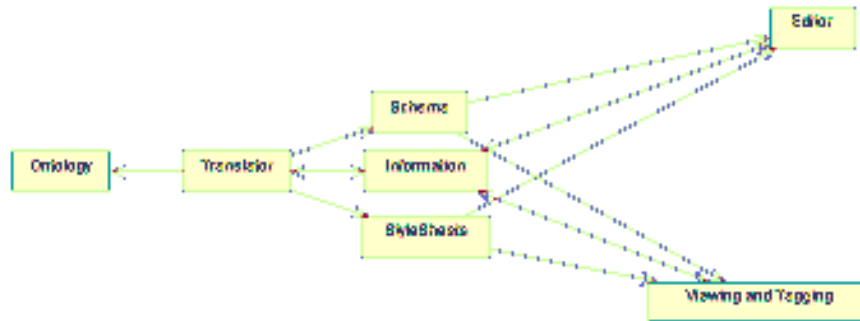


Figure 69 - Solution for User Driven Programming

The solution involves an ontology and a translator that communicates with the user through XML that is 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. 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. To complete the solution, a translator will be created using pure XML or RDF/XML programming so the entire solution would be in XML based languages. This involves programming with Semantic Web languages rather than just using them for information representation. This will make the translation easier and more reliable, and improve 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. 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 will provide an interactive programming environment for computer literate non-programmers to manipulate information and construct their own models.

Research Connectivity

Further research is needed into providing a linking mechanism for 'snippets' of information. People need answers to particular questions they are asking. In order to get the facts they need it is important for the returned information to contain this. Return of the information as factual snippets that can be pieced together into a report with links to the multiple sources would aid this. Work with semantic technologies and languages such as RDF and RSS can assist in this, Auer et al (2006) write about this. Experience in this research and developing the research website has enabled modelling of collaboration and connection in research. The aim of this is to connect all the research completed for the thesis with research of others that can be incorporated and combined. This idea of connecting research via web links fits in with this quote from Steve Jobs of Apple "Creativity is just connecting things" (Jobs, 1996). Berners-Lee et al (2006) 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 and Fischetti (1999) stated "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." He also writes "There is really little else to meaning. The structure is everything." 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. These are the main factors in this research and enabling end-user programming. This 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 for many years.

Usability Testing

Some of the examples created for the thesis have been used by Rolls-Royce, and there has been feedback from this. Other examples are being tested by putting them online for use by anyone interested. The popularity has been tracked and feedback obtained via comments through a Weblog (blog) and other Web 2.0 type mechanisms, and by email. The research has also been presented often in the last 2 years, and used for a student project. An environment was created where people can use example models and evaluate their usability and usefulness. For this thesis there has only been time to test the usability of systems by putting them online and assessing user feedback ad hoc. This follows a similar model to that used for the development of open source software or collaborations such as Wikipedia (2007f), and the Semantic Web Environmental directory SWED (2006). Fischer (2007) hypothesises that this emphasis on end-user development also changes the emphasis on testing "Software testing is conducted differently. Because domain expert developers themselves are the primary users, complete testing is not as important as in the case when the developers are not the users." Testing of usability for collaboration is complex and (Johnson et al, 2003) explain how this requires interdisciplinary expertise from several fields. Semantic Web research also requires an interdisciplinary approach as explained by Berners-Lee et al (2006). As the models created and explanations for this thesis are now online, and the source code made open this makes them available for others to research and test. A project such as this can bring together people with diverse backgrounds, interests and expertise. Cheung et al (2007) make the point that open source development can avoid vendor lock-in, eliminate unnecessary complexity, give freedom to modify applications, and provide platform and application independence. It is important to establish a way of prototyping the technologies created for this thesis, in different situations such as using the modelling and end-user programming systems for teaching, student projects, industry modelling, e-science, and finally branching out into the arts. Johnson (2004) has developed sophisticated ways of understanding and providing for complex human activity and testing the success of this.

Testing and Metrics

Price et al (1996) examined testing of applications developed by end-user programmers and of the development process. Nielsen (2001) examines usability testing "Although measuring usability can cost four times as much as conducting qualitative studies (which often generate better insight), metrics are sometimes worth the expense". He does continue that metrics are becoming less expensive to apply, and he explains some metrics which can be used. These metrics are mainly based on how long it takes (up to 5) users to perform specific tasks that the software is intended for, and how satisfied they are with the software. Ernst et al (2003) examine the issues of testing knowledge engineering applications, particularly issues about access to expert users, and domain knowledge confusing the results of surveys. Ernst et al decided to publish their work and seek feedback, the same approach was used in this thesis. An important factor in the research for this thesis, and that of Ernst et al has been to provide support for collaboration within the software developed and the visualisation. This enabling of collaboration is missing in spreadsheet based software. Alternative web-based spreadsheet tools enable better collaboration but do not yet visualise this well.

Some of the technologies involved in this research are new and not yet fully tested by long term use, but the essence of the research is enabling computer based information to be edited and shared more easily. This makes it possible for computer literate people to create models without writing code. This should be a step towards making more general user programming possible.

Chapter 13 - Findings and Conclusion

Findings from using the alternative approach

This chapter explains how the alternative approach used for this thesis to develop models and modelling capabilities compared to that of spreadsheet development, used within an Airbus project. The alternative approach was outlined in this thesis, of using open standards taxonomies and a web interface for developing decision support models for design and costing. This solved problems of the spreadsheet approach as indicated below -

Maintenance

The use of a centralised information source makes these models more reliable than the standalone spreadsheet. It was much harder to update multiple instances of the spreadsheets used by different people and ensure they all contain the same information. So the first step was to build a system for collaborative model building. As a piece of information can only belong to a unique location, the problems arising from duplicate pieces of information are eliminated. 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.

Extensibility

Creating the infrastructure for the collaborative model building system took much more time than it did for the spreadsheet system, but having done so it is quicker and easier to create further models. 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 was in 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, whatever changes there may be in the software market. This use of open standards also ensures that the system can link with most environments used by others.

Ease of Use

Most 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 make this relatively easy. The models contain only the functionality that is added by the model builder unlike the spreadsheets which had generic functionality that was not required and led to user's confusion. 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 according to the type of user, the situation, or the kind of information to be shown.

Sharing of Information

The use of open standards languages for representing information makes it much easier to represent information in a way that makes it accessible both to people and software. 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.

Conclusion

Within this thesis it is argued that there is a need for software developers to create programs that enable users to solve problems themselves. This is a reaction to the increased complexity of real world problems and software systems, which makes development of software solutions impractical without greater involvement from end-users. It is also argued that the research for this thesis has been a step towards making end-user programming possible. The research ideas look complex at first glance but this research is all about simplifying software development.

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 3 represent problems currently experienced throughout software and computer use.

The alternative approach for this thesis involves 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 also enables much more of the maintenance task to be left to users, who can also customise the system. Creating the infrastructure has taken much more time than it did for the spreadsheet system, but having done so it is much quicker and easier to create further models. This indicates that the extra research and development time taken though far exceeding what is required for a spreadsheet model 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 may contain 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 are eliminated. It is also much 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 pages is a huge advantage for creation of dynamic image views and decision support models over the web.

This research was a test case for a whole new approach that could be possible, 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 practical. Semantic Web languages are ideal for representing graphs and trees in an open standard way. The spatial, and tree/graph forms both have the same underlying semantics, and therefore can both be translated to computer languages. In fact it

would be much 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 in 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 enables an environment that could be made easy 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. 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 considering both increases in complexity of manufactured products, and of software systems themselves. The reason for scepticism about 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". 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.

Research in the use and visualisation of Semantic Web information provides the tools that end-user programmers have been lacking until recently. Cheung (2005) explains 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."

For proving the hypothesis that it is possible to create an end-user programming environment, usable by non programmers, it has been found that 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. If users can understand and navigate relationships, and add new relationships they can model any problem. It was important is designing 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. There is no need for 'black box' solutions in this research that hide information. There are no dead ends or blocks to expanding and improving this approach. To make the system easier to use it is only 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.

Web References

- Adobe Flex2, 2006. *Adobe Flex 2* [online]. Available from: <http://www.adobe.com/products/flex/> [Accessed].
- Alice, 2006. *Alice v2.0 - Learn to Program Interactive 3D Graphics* [online]. Available from: <http://www.alice.org/> [Accessed].
- Alloy, 2006. *The Alloy Analyzer - 3.0 Beta* [online]. Available from: <http://alloy.mit.edu/> [Accessed].
- Amaya, 2007. Welcome to Amaya - W3C's Editor/Browser [online]. Available from: <http://www.w3.org/Amaya/> [Accessed].
- Ambler, S. W., 2003. *The Object-Relational Impedance Mismatch* [online]. Available from: <http://www.agiledata.org/essays/impedanceMismatch.html> [Accessed].
- Aosd.net, 2007. *Welcome to aosd.net* [online]. Available from: <http://aosd.net/> [Accessed].
- Apple Lisa, 2006. *Apple Lisa The First Affordable GUI* [online]. Available from: <http://fp3.antelecom.net/gcifu/applemuseum/lisa2.html> [Accessed].
- AspectXML 2007. *Community open-source project* [online]. Available from: <http://www.aspectxml.org/>
- BBC News Technology, 2006. *Fifteen years of the web* [online]. Available from: <http://news.bbc.co.uk/1/hi/technology/5243862.stm> [Accessed].
- Bellis, M., 2006. *Inventors of the Modern Computer, The History of the IBM PC - International Business Machines* [online]. Available from: <http://inventors.about.com/library/weekly/aa031599.htm> [Accessed].
- Berners-Lee, T., 1999. *The future of the Web* [online]. Available from: <http://www.w3.org/Talks/1999/0414-LCS35-tbl/slide1-1.html> [Accessed].
- Berners-Lee, T., 2000. *Semantic Web on XML* [online]. Available from: <http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide1-0.html> [Accessed].
- Berners-Lee, T., 2002. *The World Wide Web - Past Present and Future* [online]. Available from: <http://www.w3.org/2002/04/Japan/Lecture.html>, Japan prize Commemorative Lecture [Accessed].
- Berners-Lee, T., 2006. *Tim Berners-Lee* [online]. Available from: <http://www.w3.org/People/Berners-Lee/> [Accessed].
- Bricklin, D., 2006. *VisiCalc: Information from its creators, Dan Bricklin and Bob Frankston* [online]. Available from: <http://www.bricklin.com/visicalc.htm> [Accessed].
- Cagle, K., 2006. Thoughts on Complexity [online]. Available from: http://www.oreillynet.com/xml/blog/2006/03/thoughts_on_complexity.html [Accessed].
- Cagle, K., Whats Ajax?, 2007. *A Primer on the Ajax Phenomenon* [online]. Available from: http://ajaxpatterns.org/wiki/index.php?title=Whats_Ajax%3f [Accessed].
- Cover Pages, 2001. *STEPml XML Specifications* [online]. Available from: <http://xml.coverpages.org/stepml.html> [Accessed].
- Cover Pages, 2002. *Project Management XML Schema (PMXML)* [online]. Available from: <http://xml.coverpages.org/projectManageSchema.html> [Accessed].
- Croquet, 2006. *A new open source software platform for creating deeply collaborative multi-user online applications* [online]. Available from: <http://www.opencroquet.org/> [Accessed].

- De Andreis, E., 2005. *2MXtree XML-based tree* [online]. Available from: <http://manudea.duemetri.net/manudea/xtree/default.asp> [Accessed].
- De Roure, D., Baker, M. A., Jennings, N. R., Shadbolt, N. R., 2003. *The Evolution of the Grid* [online]. Available from: <http://www.semanticgrid.org/documents/evolution/evolution.pdf>, Southampton University [Accessed].
- De Roure, D., Baker, M. A., Jennings, N. R., Shadbolt, N. R., 2003. *The Semantic Grid: A Future e-Science Infrastructure* [online]. Available from: <http://www.semanticgrid.org/documents/semgrid-journal/semgrid-journal.pdf>, Southampton University [Accessed].
- De Roure, D., Baker, M. A., Jennings, N. R., Shadbolt, N. R., 2004. *The Semantic Grid: Past, Present and Future* [online]. Available from: <http://www.semanticgrid.org/documents/semgrid2004/semgrid2004.html> [Accessed].
- Dmitriev, S., 2006. *Language Oriented Programming: The Next Programming Paradigm* [online]. Available from: <http://www.onboard.jetbrains.com/is1/articles/04/10/lop/> [Accessed].
- Dmoz Open Directory Project, 2006. *Visual Languages* [online]. Available from: <http://dmoz.org/Computers/Programming/Languages/Visual/> [Accessed].
- Dodds, L., 2007. *Introducing SPARQL: Querying the Semantic Web* [online]. Available from: <http://www.xml.com/lpt/a/2005/11/16/introducing-sparql-querying-semantic-web-tutorial.html> [Accessed].
- Dodds, L., 2007. *XML Army Knife* [online]. Available from: <http://xmlarmyknife.org/api/rdf/sparql/query> [Accessed].
- Eclipse, *Eclipse - an open development platform* [online]. Available from: <http://www.eclipse.org/> [Accessed].
- End-users Shaping Effective Software (EUSES), 2006. *Welcome to EUSES* [online]. Available from: <http://eusesconsortium.org/> [Accessed].
- Engineous Software, 2006. *Specification Version 1.6* [online]. Available from: http://www.engineous.com/product_FIPER_specifications.htm [Accessed].
- Exist, 2006. *Open Source Native XML Database* [online]. Available from: <http://exist.sourceforge.net/xquery.html> [Accessed 21st December 2006].
- Flock Browser, 2007. *the social web browser* [online]. Available from: <http://www.flock.com> [Accessed].
- Fluit, C., Marta, S., Harmelen, F. V., 2003. *Supporting User Tasks through Visualisation of Lightweight Ontologies* [online]. Available from: <http://www.cs.vu.nl/~frankh/abstracts/OntoHandbook03Viz.html> [Accessed].
- FOLD, 2006. *Students Online* [online]. Available from: <http://www.cems.uwe.ac.uk/exist/index.xql> [Accessed].
- Foster, I., Kesselman C., Nick, J. M., Tuecke, S., 2001. *The Physiology of The Grid* [online]. Argonne National Laboratory, University of Chicago, University of Southern California, Available from: <http://www.globus.org/alliance/publications/papers/ogsa.pdf> [Accessed].
- Google, 2007. *Create and share your work online* [online]. Available from: https://www.google.com/accounts/ServiceLogin?service=writely&passive=true&continue=http%3A%2F%2Fdocs.google.com%2F<mpl=WR_tmp_2_lfty&nui=1 [Accessed].

Hale, P., 2007. *AJAX/web 2.0* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/Ajax/ajax.htm> [Accessed].

Hale, P., 2007. *fetch rss feed* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/RDF/rssajax.htm> [Accessed].

Hale, P., 2007. *Flash Human Computer Interface Example Components and Materials* [online]. Available from: <http://www.cems.uwe.ac.uk/~phale/Flash/FlashHCI.htm> [Accessed].

Hale, P., 2007. *Interactive Examples* [online]. Available from: <http://www.cems.uwe.ac.uk/~phale/#InteractiveExamples> [Accessed].

Hale, P., 2007. *Interactive SVG Examples* [online]. Available from: <http://www.cems.uwe.ac.uk/~phale/InteractiveSVGExamples.htm> [Accessed].

Hale, P., 2007. *Modelling and Semantic Web Methodology* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/ModellingSemanticWeb.htm> [Accessed].

Hale, P., 2007. *Online Parametric Cost Models Examples* [online]. Available from: <http://www.cems.uwe.ac.uk/~phale/ParametricModelExamples/engine.xml> [Accessed].

Hale, P., 2007. *Process Specification Language Example* [online]. Available from: <http://www.cems.uwe.ac.uk/~phale/XMLDemonstrators/psl.xml> [Accessed].

Hale, P., 2007. *RDF/OWL - Resource Description Framework - and Semantic Web* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/RDF/RDF.htm> [Accessed].

Hale, P., 2007. *SEEDS RSS Feed* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/RDF/RDF.htm> [Accessed].

Hale, P., 2007. *Semantic Web* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/RDF/RDF.htm> [Accessed].

Hale, P., 2007. *Spar Menu* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/SparMenu.xml> [Accessed].

Hale, P., 2007. *Stepml* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/STEPml.htm> [Accessed].

Hale, P., 2007. *SVG - Scalable Vector Graphics* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/SVG/SVG.htm> [Accessed].

Hale, P., 2007. *User Driven Modelling Example* [online]. Available from: http://www.cems.uwe.ac.uk/~phale/RectangleDemo/RectangleDemo.viewlet/RectangleDemo_launcher.html [Accessed].

Hale, P., 2007. *User Driven Programming* [online]. Available from: http://www.cems.uwe.ac.uk/~phale/RectangleDemo/RectangleDemo.viewlet/RectangleDemo_launcher.html [Accessed].

Hale, P., 2007. *Wing - Adaptation of Christophe Bru's Cost Map to colour coding* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/WingMap/Wing.xml> [Accessed].

Hale, P., 2007. *XForms Articles/Tutorials* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/Ajax/ajax.htm#XForms> [Accessed].

Hale, P., 2007. *XML Extensible Markup Language* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/XML/XML.htm> [Accessed].

- HP Labs Semantic Web, 2006. *HP Labs Semantic Web Research* [online]. Available from: <http://www.hpl.hp.com/semweb/> [Accessed].
- Hunter, A., 2002. *Engineering Ontologies* [online]. Available from: <http://www.cs.ucl.ac.uk/staff/a.hunter/tradepress/eng.html> [Accessed].
- Institute for End-user Computing (IEUC), 2006. Institute for End-user Computing Inc. [online]. Available from: <http://www.ieuc.org/home.html> [Accessed].
- Institute for People-Centred Computation (IP-CC), 2006. *Supporting Discovery in Design and Innovative Decision-making* [online]. Available from: <http://www.ip-cc.org.uk/> [Accessed].
- Jena, 2006. *Jena - A Semantic Web Framework for Java* [online]. Available from: <http://jena.sourceforge.net/> [Accessed].
- KAON, 2006. *Welcome to KAON* [online]. Available from: <http://kaon.semanticweb.org/> [Accessed].
- Koala Publishers [online]. Available from: <http://www.koalapub.co.uk/> [Accessed].
- Kyoto Prize Laureates, 2004. *Dr. Alan Curtis Kay (U.S.A., b. 1940) - Computer Scientist, President, Viewpoints Research Institute* [online]. Available from: http://www.kyotoprize.com/commentary_kay.htm [Accessed].
- Lemos, M., 2006. *MetaL: An XML based Meta-Programming language* [online]. Available from: <http://www.meta-language.net> [Accessed].
- LPA VisiRule 1.0 [online]. Available from: <http://www.lpa.co.uk/vsr.htm> [Accessed].
- Lubell, J., 2006. *Representation of Process Descriptions* [online]. Available from: <http://ats.nist.gov/psl/xml/process-descriptions.html> Accessed 7th December 2006].
- Metatomix, 2007. *m3t4 Dashboard* [online]. Available from: <http://wiki.m3t4.com/homepage.action> [Accessed].
- Metatomix, 2007. *Metatomix Provides Free Semantic Toolkit for Eclipse* [online]. Available from: <http://www.metatomix.com/news/060307.html> [Accessed].
- Mikhaleenko, P., 2005. *Introducing SKOS* [online]. Available from: <http://www.xml.com/lpt/a/2005/06/22/skos.html> [Accessed].
- MIT Logo Foundation, 2006. *What is Logo?* [online]. Available from: <http://el.media.mit.edu/Logo-foundation/logo/index.html> [Accessed].
- National Institute of Standards and Technology NIST, 2006. *Manufacturing Systems Integration Division* [online]. Available from: <http://www.mel.nist.gov/msid/> [Accessed].
- National Institute of Standards and Technology NIST, 2006. *A Few PSL Basics...* [online]. Available from: <http://www.mel.nist.gov/psl/>. [Accessed 7th December 2006].
- National Institute of Standards and Technology NIST, 2006. *Rationale* [online]. Available from: <http://www.mel.nist.gov/psl/rationale.html>. Accessed 7th December 2006].
- National Institute of Standards and Technology NIST, 2006. *Working with industry to foster innovation, trade, security and jobs* [online]. Available from: <http://www.nist.gov/> [Accessed 7th December 2006].
- Network of Excellence on End-user Development in Europe (EUD.Net), 2006. *Network of Excellence on End-user Development* [online]. Available from: <http://giove.cnuce.cnr.it/eud-net.htm> [Accessed].

- Nielsen, J., 2001. Usability Metrics [online]. Available from: <http://www.useit.com/alertbox/20010121.html> [Accessed].
- OASIS, 2006. *Advancing E-Business Standards Since 1993* [online]. Available from: <http://www.oasis-open.org/home/index.php> [Accessed 7th December 2006].
- OASIS, 2007. *Universal Business Language (UBL)* [online]. Available from: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl [Accessed].
- Ontolingua, 2006. *Ontolingua* [online]. Available from: <http://www.ksl.stanford.edu/software/ontolingua/> [Accessed].
- OpenCyc, 2006. *General knowledge base and commonsense reasoning engine* [online]. Available from: <http://opencyc.org/> [Accessed].
- OpenLaszlo, 2007. *Solutions: Overview* [online]. Available from: <http://www.laszlosystems.com/developers/> [Accessed].
- Orbeon Forms, 2006. *Form-based web applications, done the right way* [online]. Available from: <http://www.orbeon.com/> [Accessed 7th December 2006].
- Palo Alto Research Center (PARC) - History, 2006. *PARC History* [online]. Available from: <http://www.parc.xerox.com/about/history/default.html> [Accessed].
- PDES Inc, 2007. Why is STEP Important? [online]. Available from: http://pdesinc.aticorp.org/step_overview.html [Accessed].
- Peterson, M. D., 2005. *O'Reilly XML.com - [Part 3] Assets, Atom Feeds, and AspectXML - The Triple Threat of Web Development?* [online]. Available from: http://www.oreillynet.com/xml/blog/2005/09/part_3_assets_atom_feeds_and_a.html [Accessed].
- Power, D. J., 2006. *A Brief History of Spreadsheets* [online]. Available from: <http://dssresources.com/history/sshistory.html> [Accessed].
- Power, D. J., 2006. Free Decision Support Systems Glossary [online]. Available from: <http://www.DSSResources.COM/glossary/> [Accessed].
- Program-Transformation.Org, 2007. *Model Transformation* [online]. Available from: <http://www.program-transformation.org/Transform/ModelTransformation> [Accessed].
- Program-Transformation.Org, 2007. *Program Transformation* [online]. Available from: <http://www.program-transformation.org/Transform/ProgramTransformation> [Accessed].
- Protégé Community Wiki, 2006. *User Driven Programming* [online]. Available from: <http://protege.cim3.net/cgi-bin/wiki.pl?UserDrivenProgramming> [Accessed].
- Rauterberg, M., 2006. *History of HCI - Sketchpad (1963) Ivan Sutherland - Presentation by MIT Lab* [online]. Available from: <http://www.idemployee.id.tue.nl/g.w.m.rauterberg/presentations/HCI-history/sld020.htm> [Accessed].
- RDF:about, 2006. *Quick Intro to RDF* [online]. Available from: <http://rdfabout.com/quickintro.xpd> [Accessed].
- Ruby on Rails, 2007. *Web development that doesn't hurt* [online]. Available from: <http://www.rubyonrails.org/> [Accessed].
- RuleML, 2007. *The Rule Markup Initiative* [online]. Available from: <http://www.ruleml.org/> [Accessed].

Schrage, M., 1991. *Spreadsheets: Bulking Up On Data* [online]. Available from: <http://www.systems-thinking.org/buod/buod.htm> [Accessed].

Semantic Web Environmental directory SWED, 2006. *Summary* [online]. Available from: <http://www.swed.org.uk/swed/about/> [Accessed].

Simons, C. L. Parmee, I. C., 2006. *A manifesto for cooperative human / machine interaction* [online]. Available from: <http://www.cems.uwe.ac.uk/~clsimons/Publications/CooperativeInteraction.pdf> [Accessed].

Sims, R., 2007. *AND AND AND AND (PLE)* [online]. Available from: <http://blog.simslearningconnections.com/?p=50> [Accessed 4th June 2007].

Softartisans, 2007. *ExcelWriter* [online]. Available from: <http://officewriter.softartisans.com/officewriter-37.aspx> [Accessed].

Stanford University, 2006. *The Demo* [online]. Available from: <http://sloan.stanford.edu/mousesite/1968Demo.html> [Accessed].

Stanford University, 2006. *Welcome to protégé* [online]. Available from: <http://protege.stanford.edu/> [Accessed].

STEPtools, 2007. *ST-Developer Tools Reference* [online]. Available from: http://www.steptools.com/support/stdev_docs/devtools/devtools.html [Accessed 3rd January 2007].

Sun Developer Network, 2007. *Code Samples and Apps - Applets* [online]. Available from: <http://java.sun.com/applets/index.html> [Accessed].

TopBraid Composer, 2006. *The Complete Semantic Modeling Toolset* [online]. Available from: <http://www.topbraidcomposer.com/> [Accessed].

T-Systems, 2007. *T-Systems Business Flexibility* [online]. Available from: <http://www.t-systems.co.uk/> [Accessed].

University of Victoria, 2006. *Model Driven Visualization (MDV)* [online]. Available from: <http://www.thechiselgroup.org/?q=mdv> [Accessed].

University of the West of England - SEEDS Group, 2006. *Semantic Web Modelling Centre Proposal - Examples Page* [online]. Available from: <http://www.cems.uwe.ac.uk/amrc/seeds/models.htm> [Accessed].

Vanguard System, 2006. *Global Knowledge Portal* [online]. Available from: <http://wiki.vanguardsw.com/> [Accessed].

Vanguard System, 2006. *Vanguard System* [online]. Available from: <http://www.vanguardsw.com/products/vanguard-studio/> [Accessed].

Veryard, R., 2001. *Data Mappings* [online]. Available from: <http://www.users.globalnet.co.uk/~rxv/infomgt/datamapping.pdf> [Accessed].

Virtual Projects, 2002. *PMXML - a XML standard for project management* [online]. Available from: http://www.vrtpri.com/content/istandards/pmxml_en.html [Accessed].

Visual Knowledge, 2006. *Visual Knowledge Semantic Wiki* [online]. Available from: <http://www.visualknowledge.com> [Accessed 20th December 2006].

Volz, R., Staab S., Oberle D., Motik B., 2003. *KAON SERVER - A Semantic Web Management System* [online]. Available from: <http://www.aifb.uni-karlsruhe.de/WBS/dob/pubs/www2003.pdf> [Accessed].

Walker, D. W., 2003. *Emerging Distributed Computing Technologies* [online]. Available from: <http://www.cs.cf.ac.uk/User/David.W.Walker>, Cardiff University [Accessed].

Ward, M., 2007. *Web users driving change in 2007* [online]. Available from: <http://news.bbc.co.uk/1/hi/technology/6198125.stm> BBC Technology [Accessed].

WebProtege, 2007. *Wiki Page* [online]. Available from: <http://protege.cim3.net/cgi-bin/wiki.pl?WebProtege> [Accessed].

Whiteside, S., 2006. *Simkin the embeddable scripting language* [online]. Available from: <http://www.simkin.co.uk/> [Accessed].

Wikipedia, 2007. *Cross-cutting concern* [online]. Available from: http://en.wikipedia.org/wiki/Cross-cutting_concern [Accessed].

Wikipedia, 2007. *Dartmouth BASIC* [online]. Available from: http://en.wikipedia.org/wiki/Dartmouth_BASIC [Accessed].

Wikipedia, 2007. *Interaction* [online]. Available from: <http://en.wikipedia.org/wiki/Interaction> [Accessed].

Wikipedia, 2007. *Metaprogramming* [online]. Available from: <http://en.wikipedia.org/wiki/Metaprogramming> [Accessed].

Wikipedia, 2007. *Simula* [online]. Available from: <http://en.wikipedia.org/wiki/Simula> [Accessed].

Wikipedia, 2007. *Welcome to Wikipedia* [online]. Available from: http://en.wikipedia.org/wiki/Main_Page [Accessed].

World Wide Web Consortium (W3C), 2005. *XML Pipeline Language (XPL) Version 1.0 (Draft)* [online]. Available from: <http://www.w3.org/Submission/xpl/> [Accessed].

World Wide Web Consortium (W3C), 2006. *Extensible Markup Language (XML)* [online]. Available from: <http://www.w3.org/XML/> [Accessed].

World Wide Web Consortium (W3C), 2006. *Leading the Web to its Full Potential* [online]. Available from: <http://www.w3.org/> [Accessed].

World Wide Web Consortium (W3C), 2006. *OWL Web Ontology Language Guide* [online]. Available from: <http://www.w3.org/TR/owl-guide/> [Accessed].

World Wide Web Consortium (W3C), 2006. *OWL Web Ontology Language Overview* [online]. Available from: <http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.2> [Accessed].

World Wide Web Consortium (W3C), 2006. *RDF Vocabulary Description Language 1.0: RDF Schema* [online]. Available from: <http://www.w3.org/TR/rdf-schema/> [Accessed].

World Wide Web Consortium (W3C), 2006. *Resource Description Framework (RDF)* [online]. Available from: <http://www.w3.org/RDF/> [Accessed].

World Wide Web Consortium (W3C), 2006. *Scalable Vector Graphics (SVG) XML Graphics for the Web* [online]. Available from: <http://www.w3.org/Graphics/SVG/> [Accessed].

World Wide Web Consortium (W3C), 2006. *SPARQL Query Language for RDF* [online]. Available from: <http://www.w3.org/TR/rdf-sparql-query/> [Accessed].

World Wide Web Consortium (W3C), 2006. *XQuery 1.0: An XML Query Language* [online]. Available from: <http://www.w3.org/TR/xquery/> [Accessed].

World Wide Web Consortium (W3C) Math Home, 2007. *What is MathML?* [online]. Available from: <http://www.w3.org/Math/Overview.html> [Accessed].

References

- 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)*.
- Abraham, R., Erwig, M., 2007. Exploiting Domain-Specific Structures For End-User Programming Support Tools. In: End-User Software Engineering Dagstuhl Seminar.
- Allan, A., Taylor, T., 2003. eSTAR: Telescopes and Databases as a Single Information Grid, Toward an International Virtual Observatory. In: *Proceedings of the ESO/ESA/NASA/NSF Conference, Garching, Germany, 10-14 June 2002, ESO Astrophysics Symposia*. ISBN 3-540-21001-6. Springer-Verlag Berlin/Heidelberg, pp 167.
- 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*.
- Auer, S., Riechert, T., Dietzold, S., 2006. OntoWiki - A Tool for Social, Semantic Collaboration. In: *The 5th International Semantic Web Conference, Athens, GA, USA*.
- Aziz, H., Gao, J., Maropoulos, P., Cheung, W. M., 2005. Open standard, open source and peer-to-peer tools and methods for collaborative product development. *Computers in Industry*, 56, pp 260-271.
- 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.
- Bergin, T. J., Gibson, R. G., 1996. History of Programming Languages, Volume 2, ISBN-10: 0-201-89502-1; ISBN-13: 978-0-201-89502-5.
- 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.
- Bernstein, A., Kaufmann, E., Kaiser, C., Kiefer, C., 2006. Ginseng: A Guided Input Natural Language Search Engine for Querying Ontologies. In: *Jena User Conference, Bristol, UK*.
- 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.
- Blackwell, A., 2007. Interdisciplinary Design Research for End-User Software Engineering. In: End-User Software Engineering Dagstuhl Seminar.
- Bloodsworth, P., Greenwood, S., 2005. COSMOA: An Ontology-Centric Multi-Agent System For Coordinating Medical Responses To Large-Scale Disasters. *AI Communications Vol 18 (3) Agents Applied in Health Care* pp 229-240, ISSN:0921-7126.
- Booch, G., 2007. The beauty of software. In: *British Computer Society Turing Lecture March 2007*.

- Borthick, A. F., Bowen, P. L., Donald, R. J., Micauel, H. K. T., 2001. The effects of information request ambiguity and construct incongruence on query development. *Decision Support Systems* Vol 32 pp 3-25.
- 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.
- Bruchez, E., 2006. XForms: an Alternative to Ajax?. In: *XTech 2006: Building Web 2.0* 16-19 May 2006, Amsterdam, The Netherlands.
- Brundwick, B., (ed.) US Dept. of Defense, 1995. *Parametric Cost Estimating Handbook* Joint Government/Industry Initiative
- Buehlmann, U., Ragsdale, C. T., Gfeller, B., 2000. A spreadsheet-based decision support system for wood panel manufacturing. *Decision Support Systems*. 29 pp 207-227.
- Burnett, M. M., Engels, G, Myers, B. A., Rothermel, G., 2007. End-User Software Engineering Dagstuhl Seminar.
- Cagle, K., 2006. AJAX on the Enterprise. In: *AJAXWorld conference & Expo, October 4, 2006*.
- Camarinha-Matos, L. M., Afsarmanesh, H., Osoro, A. L., 2001. Flexibility and safety in a web-based infrastructure for virtual enterprises *Computer Integrated Manufacturing* Vol 14 (1) pp 66-82.
- Carroll, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., and Wilkinson, K. 2004. Jena: implementing the semantic web recommendations. In: *Proceedings of the 13th international World Wide Web Conference* New York. WWW Alt. '04. ACM Press, New York, NY, pp 74-83.
- 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.
- Chen, C.-H., Yücesan, E., 2001. Distributed Web-Based Simulation Experiments For Optimization. *Journal of Simulation Practice and Theory*, 9, pp 73-90.
- Cheng, K., Pan, P. Y., Harrison D. K., 2001, Web-based design and manufacturing support systems: implementation perspectives. *International Journal of Computer Integrated Manufacturing*, Vol 14 (1) pp 14-27.
- 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.
- Ciancarini, P., Rossi, D., Vitali, F. 2001. Designing a document-centric coordination application over the Internet. *Interacting with Computers*, 13, pp 677-693.

- 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.
- Clarke, S., 2007. What is an End-user Software Engineer?. In: End-User Software Engineering Dagstuhl Seminar.
- 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*.
- 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.
- Correa da Silva F.S., Vasconcelos, W.W., Robertson, D.S., Brilhante, V., de Melo, A.C.V., Finger, M., Agusti, J., 2002. On the insufficiency of ontologies: problems in knowledge sharing and alternative solutions. *Knowledge-Based Systems*, Vol 15 (3) pp 147-167.
- Costabile, M. F., Piccinno, A., 2007. Software environments for supporting End-User Development. In: End-User Software Engineering Dagstuhl Seminar.
- Coutaz, J., 2007. Meta-User Interfaces for Ambient Spaces: Can Model-Driven-Engineering Help?. In: End-User Software Engineering Dagstuhl Seminar.
- 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 pp 53-85.
- Crapo, A. W., Waisel, L. B., Wallace, W. A., Willemain, T. R., 2000. Visualization and the process of modeling: a cognitive-theoretic view. In: *Conference on Knowledge Discovery in Data - Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* pp 218-226.
- Cypher, 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] 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).
- Department of Defense, 1999. *Parametric Estimating Handbook*, Second Edition (Spring 1999).
- De Souza, C., 2007. Designers Need End-User Software Engineering. In: End-User Software Engineering Dagstuhl Seminar.
- Dittrich, Y., 2007. Rethinking the Software Life Cycle: About the Interlace of Different Design and Development Activities. In: End-User Software Engineering Dagstuhl Seminar.
- Dubinko, M., 2005. *XForms Essentials*. O'Reilly.
- 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. In: *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.*
- El-Ghalayini, H., Odeh, M., McClatchey, R., 2005. Engineering Conceptual Data Models from Domain Ontologies: A Critical Evaluation. *In: IASTED International Conference on Databases and Applications, part of the 23rd Multi-Conference on Applied Informatics, Innsbruck, Austria, pp 222-227.*
- Elrad, T., Filman, R. E., Bader, A., 2001. Aspect-oriented programming: Introduction. *Communications of the ACM*, 44(10), pp 28-32.
- 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.*
- 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 KCAP [online]. Available from: <http://www.neilernst.net/docs/pubs/ernst-kcap03.pdf> [Accessed].*
- 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 [online]. Available from: http://web.engr.oregonstate.edu/~erwig/papers/Gencel_ICSE05.pdf [Accessed].*
- Feng, C., Kusiak, A., Huang, C., 1996. Cost evaluation in design form features. *Computer Aided Design*, Vol 28(11), pp 879-885.
- Feng, S., Zhang, Y., 1999. Conceptual Process Planning - a definition and functional decomposition. *In: Proceedings of the 1999 International Mechanical Engineering Congress and Exposition, Vol. 10 pp. 97-106.*
- Feng, S., Zhang, Y., 2000. Information Modeling of Conceptual Design Integrated with Process Planning. *In: Symposia on Design For Manufacturability, International Mechanical Engineering Congress and Exposition, Vol. 10 pp 97-106.*
- Fensel, D., Angele, J., Decker, S., Erdmann, M., Schnurr, H., Studer, R., Witt, A., 2000. On2broker: Lessons Learned From Applying AI to the Web *International Journal of Cooperative Information Systems*.
- 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.
- Ferreira, D. & Ferreira, J., 2001. Designing Workflow-Enabled Business-to-Business Infrastructures. *In: 7th International Conference on Concurrent Enterprising pp 81-89.*
- Fischer, G., 2007. Meta-Design: A Conceptual Framework for End-User Software Engineering. *In: End-User Software Engineering Dagstuhl Seminar.*
- Fishwick, P. A., Miller, J. A., 2004. Ontologies for Modeling and Simulation: Issues and Approaches. *In: Proceedings of the 2004 Winter Simulation Conference, Orlando, Fla, pp 259-264.*

- Fluit, C., Marta, S., Harmelen, F. V., Staab, S., Studer, R., 2003. *Handbook on Ontologies in Information Systems*. Springer-Verlag.
- Foster, I. Kesselman, C. Tuecke, S., 2001. The Anatomy of The Grid. *International Journal of Supercomputer Applications*, Argonne National Laboratory, University of Chicago, University of Southern California
- 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*.
- Goble C., De Roure D., 2002, The Grid: an application of the semantic web. *ACM SIGMOD Record* Vol 31 (4) Special Issue: Special section on semantic web and data management table of contents pp 65-70.
- Grant, D., Ngwenyama, O., 2003. A report on the use of action research to evaluate a manufacturing information systems development methodology in a company, *Information Systems Journal* 13 (1), 21-35.
- 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.
- Gropp, E., 2003. Accelerating SVG Transformations with Pipelines. In: *SVG Open 2003 - Conference and Exhibition 2nd Annual Conference on Scalable Vector Graphics - Vancouver, Canada*.
- Gross, M. D., 2007. Designers Need End-User Software Engineering. In: *End-User Software Engineering Dagstuhl Seminar*.
- Grove, R., 2000. Internet-based expert systems. *Expert Systems*, 17(3).
- 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.
- Gruber, T.R. 1993. A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, vol 5 pp 199-220.
- 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., Hill, T., Nour, M., Bru, C., Jocelyn, A., Round, M., Dunkley, M., 2001. Provision of a Web Based Decision Support System for Wing Box Tooling. In: *7th International Conference on Concurrent Enterprising*, Bremen, Germany, Italy 27-29 June 2001.

- Hale, P., Scanlan, J., Bru, C., Dunkley, M., 2002, Preliminary Findings from the DECIDE project. *In: ISPE/CE2002 Concurrent Engineering Conference* Cranfield University pp 839.
- 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*.
- 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.
- Hendler, J., 2001. Agents and the Semantic Web. *IEEE Intelligent Systems Journal*.
- 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.
- Huang, G. Q., Mak, K. L., 2001. Issues in the development and implementation of web applications for product design and manufacture. *Computer Integrated Manufacturing*, Vol 14(1), pp 125-135.
- Huang, G. Q., Mak, K. L., 2001. Web-integrated manufacturing: recent developments and emerging issues. *Computer Integrated Manufacturing*, Vol 14(1), pp 3-13.
- 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.
- Ing-Xiang, C., Chun-Lin, F., Pang-Hsiang, L., Li-Chia, K., Cheng-Zen, Y., 2005. Integrated Visualization for Semantic Web. *In: 19th International Conference on Advanced Information Networking and Applications*, 2, pp 701-706.
- Isenhour, P. L., Rosson, M. B., Carroll, J. M., 2001. Supporting interactive collaboration on the Web with CORK. *Interacting with Computers*, 13 pp 655-676.
- Jakiw, 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] ISBN:0262032139.
- Jiang, P., Fukuda, S., 2001. TeleRP-an Internet web-based solution for remote rapid prototyping service and maintenance. *Computer Integrated Manufacturing* Vol 14 (1) pp 83-94.
- JISC (Joint Information Systems Committee) Anderson, P. Technology and Standards Watch. 2007. *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].
- Jobs, S., 1996. The NeXT Insanely Great Thing. *Wired Magazine* Edited by Gary Wolf.
- 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.

- Johnson, P., May, J., Johnson, H., 2003. Introduction to Multiple Collaborative Tasks. *In: ACM Transactions on Computer-Human Interaction (TOCHI)*, Volume 10 (4) December 2003 pp 277-280.
- Jones, C.V., 1996. *Visualization and Optimization*, Kluwer Academic Publishers, Boston, MA.
- Kay, A., 2003. ETech 2003 Presentation. *In: ETech O'Reilly Emerging Technology Conference, Westin Santa Clara April 22-25 2003* [online]. Available from: <http://www.lisarein.com/alankay/tour.html>
Lisa Rein's Tour Of Alan Kay's presentation [Accessed].
- Kelly, B., Guy, M., Dunning, A., 2007. Addressing The Limitations Of Open Standards *In: Museums and the Web 2007*. San Francisco 11-13th April 2007.
- Kemp, B., Buckner, K., 1999. A taxonomy of design guidance for hypermedia design. *Interacting With Computers*, 12(2), pp 143-160.
- Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W. G., 2001. Getting Started with AspectJ. *Communications of the ACM*, 44(10) pp 59-65.
- 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.
- Kim, Y., Choi, Y., Bong Yoo, S., 2001. Brokering and 3D collaborative viewing of mechanical part models on the Web. *Computer Integrated Manufacturing*, 14(1), pp 28-41.
- Kimberly, G., 1995. *Learning in MarketPlace: Economic Objects to Think With and Talk About*. Masters thesis. Cambridge, MA: MIT Media Laboratory.
- Ko, A. J., 2007. Barriers to Successful End-User Programming. *In: End-User Software Engineering Dagstuhl Seminar*.
- 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.
- Koonce, D., Judd, R., Keyser, T., Bailey, M. A., 2000. A Cost Estimation Tool Integrated into FIPER. American Institute of Aeronautics and Astronautics. [online]. Available from: <http://www.engineous.com/resources.htm> [Accessed].
- Kuljis, J., Paul, R. J., 2001. An appraisal of web-based simulation: whither we wander?. *Simulation Practice and Theory*, 9, pp 37-54.
- Lacy, L., Gerber, W., 2004, Potential Modeling and Simulation Applications of the Web Ontology Language - OWL. *Proceedings of the 2004 Winter Simulation Conference* pp 265-270.
- Lau, H. C. W., Bing, J., Lee, W. B., Kau, K. H., 2001. Development of an intelligent data-mining system for a dispersed manufacturing network. *Expert Systems* Vol 18(4).
- 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.
- Lee, C., Chen, Y. T., 2000. Distributed visual reasoning for intelligent information retrieval on the web. *Interacting with Computers*, 12, pp 445-467.
- Li, W. D., Fuh, J. Y. H., Wong, Y. S., 2004. An Internet-enabled integrated system for co-design and concurrent engineering. *Computers in Industry* 55 pp 87-103.
- Li, W. D., 2005. A Web-based service for distributed process planning optimization. *Computers in Industry*, 56, pp 272-288.
- 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.
- Liu, L., Calton, P., Wei, H., 2001. An XML-enabled data extraction toolkit for web sources. *Information Systems* Vol 26 pp 563-583.
- 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/CADUI'2004)*. Funchal, Madeira Island, Portugal, 13-16 January.
- Macías, J. A., Castells, P., 2004. Finding Interaction Patterns in Dynamic Web Page Authoring. In: *Proceedings of the 9th IFIP Working Conference on Engineering for Human-Computer Interaction with The 11th International Workshop on Design, Specification and Verification of Interactive Systems (EHCI-DSVIS'2004)*. Hamburg, Germany.
- Marsh, R., Hill, T., Scanlan, J., Dunkley, M., Cleevely, P., 2001. Probabilistic Pseudo-generative Cost Modelling Through Virtual Template Propagation. In: *CEAS Conference on Multidisciplinary Aircraft Design and Optimisation*, pp 49-55.
- Marsh, R., Hill, T., Scanlan, J., Dunkley, M., Cleevely, P., 2002. Modelling manufacturing cost uncertainty with input distributions and exemplars. In: *9th ISPE International Conference on Concurrent Engineering: Research and Applications*, Cranfield University pp 881-890.
- Masters, J., Güngördü, Z., 2003. Structured Knowledge Source Integration: A Progress Report. In: *Integration of Knowledge Intensive Multiagent Systems*, Cambridge, Massachusetts, USA, 2003 [online]. Available from: http://www.cyc.com/doc/white_papers/kimas2003.pdf [Accessed].
- Mayo, E., Steinberg, T., 2007. The Power of Information. In: *UK Government Cabinet Office Report* [online]. Available from: http://www.cabinetoffice.gov.uk/publications/reports/power_information/power_information.pdf?id=3965.
- Mecham, M., 1999. Tools Making Cyberspace A Real Place for Design, *Aviation Week & Space Technology* December 6 1999.
- McGovern, J., Cagle, K., Bothner, P., Nagarajan, V., Linn, J., 2003. *XQuery Kick Start*. Sams 1st edition.
- McGuinness, D. L., 2000. Conceptual Modeling for Distributed Ontology Environments. In: *Proceedings of the Eighth International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues (ICCS 2000)*, Darmstadt, Germany. August 14-18, 2000.
- 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.
- McKeown, J., Grimson, J., 2000. SVG: putting XML in the picture. In: *XML Europe 2000 Paris France*.
- Mens, K., Michiels, I., Wuyts, R., 2002. Supporting Software Development through Declaratively Codified Programming Patterns. *Expert Systems with Applications* 23, pp 405-413.
- Merlo, C., Girard, P., 2004. Information system modelling for engineering design co-ordination. *Computers in Industry*, 55, pp 317-334.
- 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.
- Mohamed, A., Celik, T., 2002. Knowledge based-system for alternative design, cost estimating and scheduling. *Knowledge-Based Systems Vol 15 (3)*, pp 177-188.
- Morris, S., Neilson, I., Charlton, C., Little, J., 2001. Interactivity and collaboration on the WWW - is the 'WWW shell' sufficient?. *Interacting with Computers*, 13, pp 717-730.
- 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.
- Myers, B., Ko, A., Burnett, M., 2006. Invited Research Overview: End-User Programming Extended Abstracts. *In: CHI2006*, Montreal, Canada, pp 75-80.
- Naylor, T., Steele, I., Carter, D., Allan, A., Etherton, J., Mottram, C., 2003. eSTAR Building an Observational GRID. *In: Astronomical Data Analysis Software & Systems (ADASS) Conference ASP Conference Series, 2003, Exeter University, Liverpool John Moores University, 295.*
- Nidamarthi, S., Allen, R. H., Ram, D. S., 2001. Observations from supplementing the traditional design process via Internet-based collaboration tools. *Computer Integrated Manufacturing*, 14(1), pp 95-107.
- Nilsson, M., Palmér, M., Naeve, A., 2002. Semantic Web Metadata for e-Learning - Some Architectural Guidelines. *In: WWW2002 Hawaii USA.*
- Noy, N.F., 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.
- Olive, M., Rahmouni, H., Solomonides, T., 2007. From HealthGrid to SHARE: A Selective Review of Projects. *In: Proceedings of Healthgrid 2007 Geneva, Switzerland. April 2007*
- Olsson, E., 2004. What active users and designers contribute in the design process. *Interacting with Computers* 16, pp 377-401.
- Oren, E., Breslin, J. G., Decker, S., 2006. How Semantics Make Better Wikis. *In: WWW 2006, May 23-26, 2006, Edinburgh, Scotland.*
- Otter, M., Johnson, H., 2001. Lost in hyperspace: metrics and mental models. *Interacting with Computers* 13, pp 1-40.
- Page, E. H., 1998. The Rise of Web-Based Simulation: Implications for the High Level Architecture. *In: Proceedings of the 1998 Winter Simulation Conference* Washington DC p 1663-1668
- Page, E. H., Buss, A., Fishwick, P. A., Healy, K. J., Nance, R. E., Paul, R. J., 2000. Web-Based Simulation: Revolution or Evolution?. *ACM Transactions on Modeling and Computer Simulation*, 10(1), pp 3-17.
- Page, E. H., Opper, J. M., 2000. Investigating the application of web-based simulation principles within the architecture for a next-generation computer generated forces model. *Future Generation Computer Systems* Volume 17(2) pp 159-169.
- 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., 1999. What is Logo? And Who Needs it? An essay. In: LCSl's book, *Logo Philosophy and Implementation*. [online]. Available from: <http://www.microworlds.com/company/philosophy.pdf> [Accessed].
- 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].
- Park, M., Fishwick P. A., 2005. Ontology-based Customizable 3D Modeling for Simulation. Ph.D thesis, University of Florida. [online]. Available from: http://etd.fcla.edu/UF/UFE0010095/park_m.pdf [Accessed].
- Paternò, F., 2005. Model-based tools for pervasive usability. *Interacting with Computers*, 17(3), pp 291-315.
- Phillips, P., Rodden, T., 2001. Multi-authoring virtual worlds via the World Wide Web. *Interacting with Computers*. 13 (3), February 2001, pp. 401-426(26).
- Price, S., Cheah, L. L., Hobbs, P., 1996. Software Quality Management in End-User Programming and Object Based Rapid Application Development (RAD). In: *British Computer Society, Quality Specialist Group 4th International Conference, SQM 96*.
- Qu-Yang, C., Lin, T. S., 1997, Developing an Integrated Framework for Feature-Based Early Manufacturing Cost Estimation, *The International Journal of Advanced Manufacturing Technology*, Vol 13 pp 618-629.
- Quigley, A., 1999. Automated Tool Support for a large scale diagramming Tool. In: *2nd Australian Workshop on Constructing Software Engineering Tools (AWCSET'99)*, Macquarie University Sydney, Australia.
- Quint, V., Vatton, I., 2004. Techniques for Authoring Complex XML Documents, In: *DocEng 2004 - ACM Symposium on Document Engineering* Milwaukee October 28-30 [online]. Available from: <http://wam.inrialpes.fr/publications/2004/DocEng2004VQIV.html> [Accessed].
- Quint, V., Vatton, I., 2005. Towards Active Web Clients, In: *DocEng 2005 - ACM Symposium on Document Engineering* Bristol United Kingdom 2-4 November 2005 [online]. Available from: <http://wam.inrialpes.fr/publications/2004/DocEng2004VQIV.html> [Accessed].
- 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.
- 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://ilk.media.mit.edu/papers/Distrib-Construct.html> [Accessed].
- 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].
- Rich, C., Waters, R. C., 1990. *The Programmer's Apprentice*. The ACM Digital Library - ISBN:0-201-52425-2 .
- Rodgers, P. A., Caldwell, N. H., M., Clarkson, P.J., Huxor, A. P., 2001. *The management of concept design knowledge in modern product development organizations*. International Journal of Computer Integrated Manufacturing, 14(1), pp 108-115.
- Rosson, M. B., 2007. Position paper for EUSE 2007 at Dagstuhl. In: End-User Software Engineering Dagstuhl Seminar.
- Sampson, G., 2000, The role of taxonomy in language engineering. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*. ISSN: 1364-503X Issue: Volume 358, Number 1769 / April 15, 2000 pp 1339 - 1355.
- Sasse, M. A., 1997. Eliciting and Describing Users' Models of Computer Systems. Ph.D thesis, University of Birmingham.[online]. Available from: <http://www.cs.ucl.ac.uk/staff/a.sasse/thesis> [Accessed].
- Savage, S. L., 1996. Innovative use of spreadsheets in teaching, OR/MS Today, 23(5).
- 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/HCC'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).
- Segal, J., 2007. End-User Software Engineering and Professional End-User Developers. In: End-User Software Engineering Dagstuhl Seminar.
- 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.
- Schmitz, P., 2006. Inducing ontology from Flickr tags. In: *WWW2006 Conference, Edinburgh, UK*. May 22-26, 2006.
- Smith, D. C., 1977. *A Computer Program to Model and Stimulate Creative Thought*. Basel: Birkhauser.
- Smith, D. C., 1993. Pygmalion: An Executable Electronic Blackboard. In: A. Cypher, ed. *Watch What I Do: Programming by Demonstration*. MIT Press, Chapter 1 [online]. Available from: <http://www.acypher.com/wwid/Chapters/01Pygmalion.html> [Accessed] ISBN:0262032139.
- Smith, R., 2001. What's Required in Knowledge Technologies: A Practical View. In *Proceedings of Knowledge Technologies 2001*.
- Spahn, M., Scheidl, S., Stoitsev, T., 2007. End-User Development Techniques for Enterprise Resource Planning Software Systems. In: End-User Software Engineering Dagstuhl Seminar.
- 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*.

- Stutt, A., Motta, E., 2004. Semantic Learning Webs. *Journal of Interactive Media in Education, 2004 (10)*. Special Issue on the Educational Semantic Web. ISSN:1365-893X [online]. Available from: <http://www-jime.open.ac.uk/2004/10> [Accessed].
- Su, D., Amin, N., 2001. A CGI-based approach for remotely executing a large program for integration of design and manufacture over the Internet, *Computer Integrated Manufacturing* Vol 14 (1) pp 55-65.
- Sutherland, I., 1963. GUIdebook - Graphical User Interface Gallery - Sketchpad: A man-machine graphical communication system. Reprinted from proceedings of the AFIPS Spring Joint Computer Conference, Detroit, Michigan, May 21-23, 1963, pp. 329-346 [online]. Available from: <http://www.guidebookgallery.org/articles/sketchpadamanmachinegraphicalcommunicationsystem> [Accessed].
- Sutton, D. C., 2001. What is knowledge and can it be managed?. *European Journal of Information Systems*, Vol 10 pp 72-79.
- Sweeting, J., 2000. Risk Analysis Without Monte Carlo Simulation, *The Cost Engineer* May 2000 pp 17-20.
- Sycara, K. P., 1998. The many faces of agents. *AI Magazine*, Journal Paper
- Tao, Y., Hong, T., Sun S., 2004. An XML implementation process model for enterprise applications. *Computers in Industry* 55 pp 181-196.
- Thomson, J. R., Greer, J., Cooke, J., 2001. Automatic generation of instructional hypermedia with APHID. *Interacting with Computers* vol 13 pp 631-654.
- Tollis, I. G., 1996. Graph Drawing and Information Visualization. *ACM Computing Surveys*, 28A(4).
- Tufte, E., R., 1990. *Envisioning Information*. Graphics Press.
- Uschold, M., 2000. Creating, integrating and maintaining local and global ontologies. In: *Proceedings of ECAI-2000: The European Conference on Artificial Intelligence* Amsterdam.
- 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).
- Varian, H. R., 1997. How to Build an Economic Model in Your Spare Time. In: *Passion and Craft: Economists at Work*, edited by Michael Szenberg, University of Michigan Press.
- 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.
- Voller, V. R., Porté-Agel, F., 2002. Moore's Law and Numerical Modeling. *Journal of Computational Physics*, 179, pp 698-703.
- Volz, R., Oberle, D., Staab, S., Motik, B., 2003. KAON SERVER - A Semantic Web Management System. *Alternate Track Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, 20-24 May 2003. ACM*. [Online] Available from: <http://www.aifb.uni-karlsruhe.de/WBS/dob/pubs/www2003.pdf> [Accessed].
- 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.

Wang, C.-B., Chen, T.-Y., Chen, Y.-M., Chu, H.-C., 2005. Design of a Meta Model for integrating enterprise systems. *Computers in Industry*, 56, pp 205-322.

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, [online]. Available from: <http://www.palgrave-journals.com/jors/journal/vaop/ncurrent/full/2602279a.html>.

Wujek, B. A., Koch, P. N., McMillan, M., Chiang, W. A., 2002. Distributed, Component-Based Integration Environment For Multidisciplinary Optimal and Quality Design. *American Institute of Aeronautics and Astronautics*, [online]. Available from: <http://www.engineous.com/resources.htm> [Accessed].

Zhang, S., Weimen, S., Hamada, G., 2004. A review of Internet-based product information sharing and visualization. *Computers in Industry* Vol 54, pp 1-15.

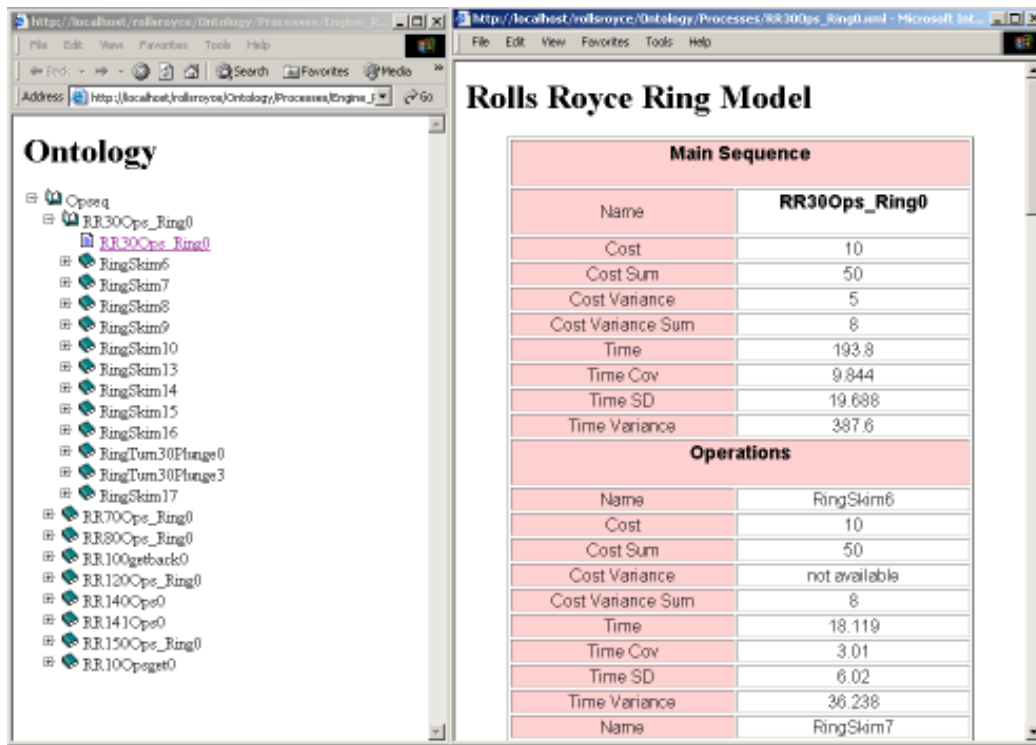
Zhang, Z., 2005. Ontology Query Languages for the Semantic Web: A Performance Evaluation. MSc Thesis, (Under the Direction of John.A.Miller).

| | |
|---|-----|
| Figure 1 - Research Area | 8 |
| Figure 2 - Adding User Layer to Model-Driven Programming | 13 |
| Figure 3 - User Generated Change | 14 |
| Figure 4 - User Generated Change, Alternative Interfaces | 15 |
| Figure 5 - Model Generated Change | 15 |
| Figure 6 - ACCS Navigation..... | 26 |
| Figure 7 - Categories of User - United States at Work 2006 | 31 |
| Figure 8 - End-user Programming Diagram..... | 32 |
| Figure 9 - Interconnection of Research and Communities..... | 34 |
| Figure 10 - Translation Process | 39 |
| Figure 11 - Architecture, sourced from Berners-Lee (2000)..... | 44 |
| Figure 12 - Language and Tool Mapping | 49 |
| Figure 13 - RDF Graph Example, Description of Aircraft | 53 |
| Figure 14 - Early End-user Modelling Example | 60 |
| Figure 15 - Early End-user Modelling Example 2 | 61 |
| Figure 16 - Interactive Applet..... | 63 |
| Figure 17 - Java Graph Layout example, possible user interface for user driven program..... | 64 |
| Figure 18 - Distributed Spreadsheet Spar Definition | 64 |
| Figure 19 - DAML Oil Web Program Step 1 of Wizard..... | 65 |
| Figure 20 - Ontology of Engine Components and related information..... | 66 |
| Figure 21 - Selection of Material for components | 67 |
| Figure 22 - Translation and Aspect-Oriented Programming..... | 72 |
| Figure 23 - Translation Process | 73 |
| Figure 24 - Translation Process Chain..... | 74 |
| Figure 25 - Visualisation of Equation | 79 |
| Figure 26 - XML Based Software for capturing information | 81 |
| Figure 27 - Automated Generation of Web Forms | 82 |
| Figure 28 - Semantic Web Modelling system..... | 84 |
| Figure 29 - Rectangle Explanation - Protégé 1 | 86 |
| Figure 30 - Rectangle Explanation - Protégé 2 | 87 |
| Figure 31 - Rectangle Explanation - Protégé 3 | 87 |
| Figure 32 - Rectangle Explanation - Protégé 4 | 88 |
| Figure 33 - Rectangle Explanation - Protégé 5 | 88 |
| Figure 34 - Rectangle Explanation - Protégé 6 | 89 |
| Figure 35 - Rectangle Explanation - Vanguard System 1 | 89 |
| Figure 36 - Rectangle Explanation - Vanguard System 2 | 90 |
| Figure 37 - Rectangle Explanation - Vanguard System 3 | 91 |
| Figure 38 - Rectangle Explanation - Vanguard System 4 | 91 |
| Figure 39 - Rectangle Explanation - Web View Tree 2 | 92 |
| Figure 40 - Rectangle Explanation - Web View Diagram 1 | 93 |
| Figure 41 - Rectangle Explanation - Web View Diagram 2 | 93 |
| Figure 42 - Rectangle Explanation - Web View Diagram 3 | 94 |
| Figure 43 - Rectangle Explanation - Web View Diagram 4 | 94 |
| Figure 44 - Rectangle Explanation - Web View Diagram 5 | 95 |
| Figure 45 - Translation Process Implementation | 97 |
| Figure 46 - Ontology to Model Conversion..... | 98 |
| Figure 47 - Spar Diagram | 99 |
| Figure 48 - Spar branch automatically created from information source..... | 100 |
| Figure 49 - Part Definition Branch | 101 |
| Figure 50 - Translation of Equation Representation from Protégé to Vanguard System..... | 102 |
| Figure 51 - Pre Preg Mass Calculation | 103 |
| Figure 52 - Hand Layup Calculation..... | 103 |
| Figure 53 - Using the tree view for cost drill down | 103 |
| Figure 54 - Web page showing translated XML displayed as interactive web application | 104 |
| Figure 55 - Parametric Cost Estimation | 105 |
| Figure 56 - Interactive Spar Diagram (SVG)..... | 106 |
| Figure 57 - XML web translation from Decision tree..... | 107 |
| Figure 58 - Flash interface for navigating exported XML tree | 108 |
| Figure 59 - Flash viewing of Spar Part Definition node | 109 |

| | |
|---|-----|
| Figure 60 - Translation from decision tree into Java | 109 |
| Figure 61 - Translation from decision tree into Java Applet..... | 109 |
| Figure 62 - Translation from decision tree to Cost Estimator..... | 110 |
| Figure 63 - Semantic Search interface | 110 |
| Figure 64 - Results from semantic search | 111 |
| Figure 65 - WebProtege Searching - Cure Cycle..... | 112 |
| Figure 66 - Future Research Area..... | 116 |
| Figure 67 - Software System for Modelling | 117 |
| Figure 68 - E-Learning and End-user Programming..... | 118 |
| Figure 69 - Solution for User Driven Programming | 120 |

Appendix

Open Standards



RDF Process sequence display

RDF

```
<rdf:Description about="http://uwe/SequenceDefinition/Sequence">
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:s="http://localhost/schema/">
<s:Sequence>
  <rdf:Sequence>
    <rdf:Description about="Sequence">
      <s>Name>RR30Ops_Ring0</s>Name>
      <s:Cost>10</s:Cost>
      <s:CostSum>50</s:CostSum>
      <s:CostVariance>5</s:CostVariance>
      <s:CostVarianceSum>8</s:CostVarianceSum>
      <s:Time>193.8</s:Time>
      <s:TimeCov>9.844</s:TimeCov>
      <s:TimeSD>19.688</s:TimeSD>
      <s:TimeVariance>387.6</s:TimeVariance>
    </s:Sequence>
    <rdf:Sequence>
      <rdf:Description about="Operation">
        <s>Name>RingSkim6</s>Name>
        <s:Cost>10</s:Cost>
        <s:CostSum>50</s:CostSum>
        <s:CostVariance>not available</s:CostVariance>
        <s:CostVarianceSum>8</s:CostVarianceSum>
```

```

<s:Time>18.119</s:Time>
<s:TimeCov>3.01</s:TimeCov>
<s:TimeSD>6.02</s:TimeSD>
<s:TimeVariance>36.238</s:TimeVariance>
</rdf:Description>
<rdf:Description about="Operation">
  <s:Name>RingSkim7</s:Name>
  ...
</rdf:Description>
</rdf:Sequence>
</s:Sequence>
</rdf:Description>
</rdf:Sequence>
</s:Sequence>
</rdf:RDF>

```

DAML/OIL

In the example below, the class spar has properties of 'Area', 'Clean Area', 'Depth', 'Length', 'Periphery', 'Root Width', 'Shot Blast Contact Area Adjustment', 'Support Per Metre' and 'Tip Width'. It has a subclass of 'CentreSparII' and other subclasses not listed here. The line `<a:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#Double"/>` illustrates another facility of DAML+OIL that of data typing.

DAML+OIL Representation of Spar displayed using Stylesheet

```

<?xml version='1.0' encoding='ISO-8859-1'?>
<?xml-stylesheet type="text/xsl" href="http://localhost/daml/wingboxrdfindividual.xsl"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!ENTITY a 'http://www.daml.org/2001/03/daml+oil#'>
]>

```

```

<rdf:RDF xmlns:a="&a;"
  xmlns:rdf="&rdf;">
<a:Ontology rdf:about="Spar"
  a:versionInfo=""/>

<a:DatatypeProperty rdf:about="Spar">
  <a:domain rdf:resource="Area"/>
</a:DatatypeProperty>
<a:DatatypeProperty rdf:about="Spar">
  <a:domain rdf:resource="Clean Area"/>
</a:DatatypeProperty>
<a:DatatypeProperty rdf:about="Spar">
  <a:domain rdf:resource="Depth"/>
</a:DatatypeProperty>
<a:DatatypeProperty rdf:about="Spar">
  <a:domain rdf:resource="Length" rdf:about="Length in metres"/>
</a:DatatypeProperty>
<a:DatatypeProperty rdf:about="Spar">
  <a:domain rdf:resource="Periphery "/>
</a:DatatypeProperty>
<a:DatatypeProperty rdf:about="Spar">
  <a:domain rdf:resource="Root Width"/>
</a:DatatypeProperty>
<a:DatatypeProperty rdf:about="Spar">
  <a:domain rdf:resource="Shot Blast Contact Area Adjustment"/>
</a:DatatypeProperty>
<a:DatatypeProperty rdf:about="Spar">
  <a:domain rdf:resource="Support Per Metre"/>
</a:DatatypeProperty>
<a:DatatypeProperty rdf:about="Spar">
  <a:domain rdf:resource="Tip Width"/>
  <a:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#Double"/>
</a:DatatypeProperty>

<a:Class rdf:about="CentreSparII">
  <a:subClassOf rdf:resource="Spar"/>
</a:Class>

...

</rdf:RDF>

```

UML design translated into XMI and rendered with a stylesheet. A user can click on an associated term to view that term.

Product Data Taxonomy

| | | |
|-------------|-----------------|------|
| Class | Adjust | |
| Supertypes: | <u>Handling</u> | |
| Attributes: | | |
| visibility | type | name |
| Operations: | | |
| visibility | return | name |

| | | |
|-------------|--|------|
| Class | AllCells | |
| Supertypes: | <u>V</u> | |
| Subtypes: | <u>AssemblyCell, ATLN01, ATL_Cell, AutoclaveHoldCell, Autoclave_exit, CleanOpArea, ConsolidationCell, CureCell, MachiningCell, NDTCell, PaintingCell, PreformExtraction, ToolCleaningCell, VacuumForming</u> | |
| Attributes: | | |
| visibility | type | name |
| Operations: | | |
| visibility | return | name |

XMI Taxonomy

Process Specification Language - PSL

The first step is to declare the resources; this is shown below.

Process Specification Language Example

| Resource Classes | |
|------------------|------------|
| Class | |
| Paint | |
| Class | |
| PaintBrush | |
| Class | |
| PaintMixer | |
| Class | |
| PaintThinner | |
| Class | |
| SandPaper | |
| Class | |
| Grit | |
| #grit | grit |
| #Grit | #Grit |
| #SandPaper | #SandPaper |

PSL Class Definition

Instances of this class can then be declared. This is illustrated next.

| Resource Instances | | |
|--------------------|--|--------------|
| bustaron- | | |
| cat | | 100 |
| bustaron- | | |
| cat | | 200 |
| bustaron- | | |
| Paint | | paint-primer |
| bustaron- | | |
| Paint | | paint-blue |
| bustaron- | | |
| Paintbrush | | brush |
| bustaron- | | |
| Paint | | mixer |
| bustaron- | | |
| Paintbrush | | thinner |
| bustaron- | | |
| Sandpaper | | s1 |
| silt | | #100 |

PSL Instance Creation

Time points are then created to make it possible to create a sequence of activities. This is shown below.

| 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 |

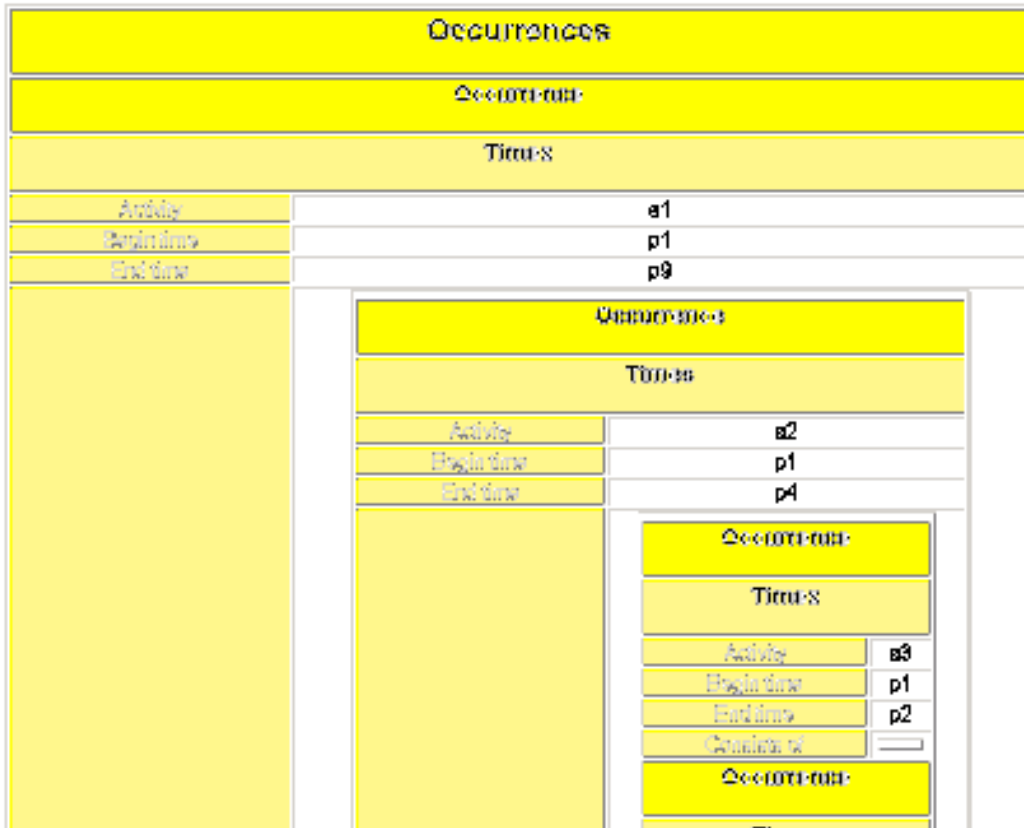
PSL Time Point Creation

After this activities are specified. This is shown below.

| Activity Specifications | | |
|-------------------------|----------------|---------------|
| Activity | a1 | |
| a276 | Finish product | |
| Division of | Sub Activity | a2 |
| | a276 | Paint |
| | Sub Activity | a3 |
| | Name | Mix paint |
| | Resources | #Paint |
| | Resources | #PaintMixer |
| | Sub Activity | a4 |
| | a276 | Apply paint |
| | Resources | #Paint |
| | Resources | #PaintBrush |
| | Sub Activity | a5 |
| | Name | Clean brush |
| | Resources | #PaintBrush |
| | Resources | #PaintThinner |
| Sub Activity | a6 | |
| Name | Sand | |

PSL Activity Specification

The activities are then assigned occurrence times that allow each activity to be related in a sequence of sequences. The figure below demonstrates this.



Activity occurrence times

The next figure shows a step example, implemented with a stylesheet, based on an example from the STEPml website. Step Tools (2007) provide tools for building STEPml applications.

| STEPML Example | | | | | | | | | | | | | |
|--|--|------------------|--|-----------|------------------------------|------|----------------|-------------|---|--------------|--|----|--|
| application protocol definition | | | | | | | | | | | | | |
| version | version 1.1 | | | | | | | | | | | | |
| application identifier file extension | pdm_schema | | | | | | | | | | | | |
| application protocol year | 2000 | | | | | | | | | | | | |
| application | <table border="1"> <tr> <td colspan="2">application root</td> </tr> <tr> <td>mechanics</td> <td>mechanical design</td> </tr> </table> | application root | | mechanics | mechanical design | | | | | | | | |
| application root | | | | | | | | | | | | | |
| mechanics | mechanical design | | | | | | | | | | | | |
| application organization assignment | | | | | | | | | | | | | |
| assigned organization | <table border="1"> <tr> <td colspan="2">Organization</td> </tr> <tr> <td>name</td> <td>Modern Design Services Corp.</td> </tr> <tr> <td>id</td> <td>MDSC</td> </tr> <tr> <td>description</td> <td>Engineering Design & Services firm</td> </tr> </table> | Organization | | name | Modern Design Services Corp. | id | MDSC | description | Engineering Design & Services firm | | | | |
| Organization | | | | | | | | | | | | | |
| name | Modern Design Services Corp. | | | | | | | | | | | | |
| id | MDSC | | | | | | | | | | | | |
| description | Engineering Design & Services firm | | | | | | | | | | | | |
| assigned organization role | <table border="1"> <tr> <td colspan="2">Role</td> </tr> <tr> <td>id</td> <td>id owner</td> </tr> </table> | Role | | id | id owner | | | | | | | | |
| Role | | | | | | | | | | | | | |
| id | id owner | | | | | | | | | | | | |
| Item | <table border="1"> <tr> <td colspan="2">product</td> </tr> <tr> <td>id</td> <td>vs1</td> </tr> <tr> <td>name</td> <td>valve assembly</td> </tr> <tr> <td></td> <td> <table border="1"> <tr> <td colspan="2">product root</td> </tr> <tr> <td>id</td> <td></td> </tr> </table> </td> </tr> </table> | product | | id | vs1 | name | valve assembly | | <table border="1"> <tr> <td colspan="2">product root</td> </tr> <tr> <td>id</td> <td></td> </tr> </table> | product root | | id | |
| product | | | | | | | | | | | | | |
| id | vs1 | | | | | | | | | | | | |
| name | valve assembly | | | | | | | | | | | | |
| | <table border="1"> <tr> <td colspan="2">product root</td> </tr> <tr> <td>id</td> <td></td> </tr> </table> | product root | | id | | | | | | | | | |
| product root | | | | | | | | | | | | | |
| id | | | | | | | | | | | | | |

STEPml example