

User Driven Programming Detailed Explanation

Html Version -

<http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/DetailedExplanation/UserDrivenModellingDetailedExplanation.htm>

Adobe Version -

<http://www.cems.uwe.ac.uk/amrc/seeds/PeterHale/DetailedExplanation/UserDrivenModellingDetailedExplanation.pdf>

Abstract

The Systems Engineering Estimation and Decision Support (SEEDS) team is part of the Aerospace Manufacturing Research Centre (AMRC) at the University of the West of England (UWE), Bristol. Our expertise is in working mainly with aerospace manufacturers to apply techniques in managing, categorising, and visualising information to support costing of products.

This paper outlines the technique of User Driven Modelling a technique for End User Programming. 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 re-engineer 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.

This paper 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. We are using the open standard Stanford University Ontology tool Protégé. This Ontology can be translated into a Decision Support tool called DecisionPro, which runs the model. Software we have created using DecisionPro allows calculations of the cost of a design to be made, and provides a colour-coded representation of the product tree. It is then possible to output this tree in the form of 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.

Keywords

Cost, Costing, Decision Support, Design, Manufacture, User Driven Modelling, End User Programming, Semantic Web, Taxonomy

Introduction

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 the World Wide Web consortium (W3C) [1]. The detail and accuracy of the information that can be provided to define the product varies along this chain. The best opportunities for cost reduction are early in the product life cycle, so it is important to gather together any information that is available on that component and any similar products. 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. 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.

This paper outlines techniques used, 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. Stanford University developed the Protégé open standard ontology tool [2]. Protégé is being used for our purposes, although there are other ontology tools that could have been used. This ontology can be translated into a Decision Support tool from Vanguard called DecisionPro. Vanguard are creating a modelling network where universities can share decision support models over a network [3]. This tool was used because it was selected during a project to evaluate, and then use software to solve costing problems. We are creating a modelling network that will link to that of Vanguard <http://www.cems.uwe.ac.uk/amrc/seeds/models.htm>.

Software created using DecisionPro 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 in the form of web pages, interactive diagrams and code in programming languages such as Engineous' Java based costing tool Cost Estimator [4]. This is explained with examples from our research. It is possible to search the information both in Protégé and on the Web as the information is represented using searchable semantic web languages.

This paper is based on research undertaken to establish a way of representing information relating to the design and production of a wing box, 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 to be generic, and thus re-usable for other modelling problems, and could be created 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.

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 much more effective development of models. Users can create their own programs from scratch using this technique. The paper begins with an explanation of the spreadsheet approach and explains the alternative researched since.

The sections 'Research Aim', 'Research Approach' and 'Why a different approach is needed' explain the problems engineers have with the commissioning of costing models, that lead to our conclusion that a different approach is necessary. 'User Driven Model Development' and 'Criteria necessary for User Driven Model Development' explain the alternative approach of User Driven Modelling, and how this approach would be used and tested for engineering costing problems. 'User Driven Modelling Techniques implemented with a simple example' introduces the approach and uses the example to explain this research. 'Early Approach' gives a critical evaluation of a wing box costing spreadsheet, and explains the difficulties that would be encountered if we were to try and adapt it to a different problem. The rest of the paper explains our implementation of a subset of the spreadsheet for testing

purposes, using the User Driven Modelling (UDM) approach. We conclude with results from the comparison of the two approaches.

Research Aim

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 paper identifies ways that problems arising from the model development process can be addressed, by a new way of providing for the creation of software. We have gained 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 unites approaches of object oriented design, the Semantic Web, and relational databases and event driven programming. The approach encourages much greater user involvement in software development. The advantages of increasing user involvement in software development are explained by Olsson [5]. The intention of this research is to allow users to create the whole model/program they develop without typing code.

Figure 1 shows that users are by far the biggest group of program developers, with two intermediate groups, and a small group of professional programmers.

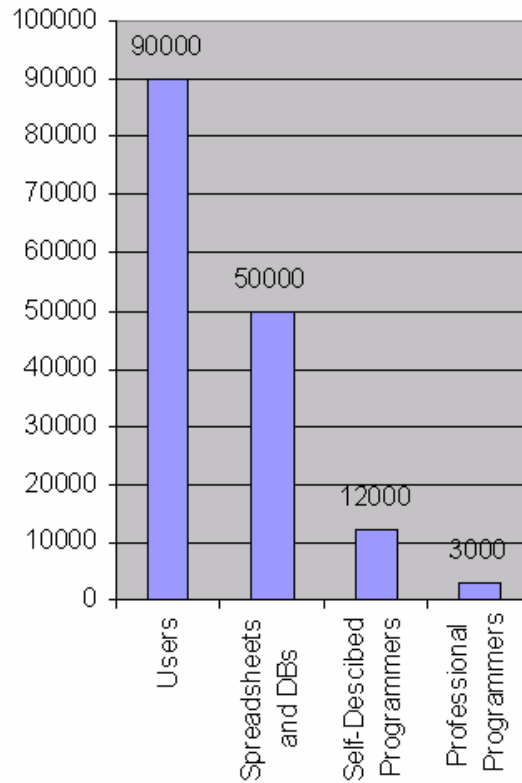


Figure 1 - Categories of User - United States at Work 2006

Source - Scaffidi et al. [6] based on data from US bureau of Labour Statistics.

These statistics were also explained and referred to in Myers et al. [7].

Categories of User

It is important to distinguish between the two different types of users for the system, as they would work on different parts of the overall system. However a person may be represented in either or both categories.

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, and some knowledge of the domain to which it is to be applied.

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 want to be able to use a model to evaluate a problem in order to help in decision making.

Expertise of Users

Within this paper 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 our research could be applied to other domains. The users/domain experts may well be computer literate and able to model certain problems using a software tool such as a spreadsheet. For reasons that will be explained later, this is only sufficient for simpler problems. The reasons that spreadsheets should not be used to represent complex models are connected with difficulties in maintaining, extending, and reusing spreadsheet models. This might not be such a problem in future if research such as that of Oregon State and Houston Universities can succeed in automatically generating correct spreadsheets, and solving errors of meaning (semantic errors) [8].

For now, to be able to model a complex problem, the users/domain experts currently must specify their requirements to other software experts, who may or may not have domain knowledge themselves. It is difficult to find and afford those who have sufficient expertise in both the software and the domain. Someone without the domain knowledge may not understand the requirements. Putting the right team together is a difficult balancing act, and sometimes may be difficult or impossible.

Software development is time consuming and error prone because of the need to learn computer languages. If people could instruct a computer without this requirement they could concentrate all their effort on the problem to be solved. This is termed User Driven Programming (UDP) within this paper, and for the examples demonstrated the term User Driven Modelling (UDM) is used to explain the application of User Driven Programming to model development. This research aims 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. The software developed as part of this research translates this human readable representation into computer languages. The tree also shows the flow of information. This technique is a kind of End User Programming, research in this area is undertaken by the EUSES (End Users Shaping Effective Software) research collaboration [9].

Research Approach

This research demonstrates how a taxonomy can be used as the information source, from which it is possible to automatically produce software. This technique is most suitable at present to modelling, visualisation, and searching for information. The paper 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 a visual environment 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 systems such as Protégé and Jena [10]. 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 (explained later).

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 that 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 this feedback.

A modelling environment needs to be created by software developers in order to allow users/model builders/domain experts to create their own models. This modelling environment could be created using an open standard language such as XML (eXtensible Markup Language). As the high level translation though this would depend on tools developed using lower level languages, this is why tools such as Protégé and DecisioPro are used. 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.

A requirement of this research is that open standard semantic languages are used to represent information, to be used both as input and output of the model. These languages are based on XML. These same open standard languages can be used for developing the program code of models. It is proposed that software and information represented by the software, be 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 writing of programs by other programs. The purpose of this 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 present day 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.

Dynamic software systems such as outlined by Huhns [11] have been examined. 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 [12] explains 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 methods used for this representation and translation will be explained in the rest of this document.

Why a different approach is needed

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 the users is to specify requirements rather than create programs, creating working programs conveys requirements much better than any other form of requirement specification.

It may also prove possible to work in reverse from implementation to design, or design to conceptual model. A UWE paper explains how ontologies could be mapped to conceptual models, El-Ghalayini Et al. [13]. This process can be made easier if the same open standard software representations, languages, and structures are used throughout this process. This would be useful for checking software is designed well or re-using software designs.

User involvement is important in the 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 as close as possible to that 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, the 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.

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 a user entering information visually in the form of a tree diagram. The research involves developing ways of automatically translating this information into program code in a variety of computer languages. This is very important and useful for many employees that have insufficient time to learn programming languages. To achieve this, visual editors are used to create and edit taxonomies to be translated into code. To make this possible, it is also important to examine visualisation, and visualisation techniques to create a human computer interface that allows non-experts to create software.

The research mainly concentrates on using the above technique for modelling, searching and sorting. The technique should be usable for other types of program development. Research relevant to User Driven Programming in general is covered, as this could be applied to the problem in future.

This research unites approaches of object oriented design, the Semantic Web, relational databases, and event driven programming. Tim Berners-Lee defined the semantic web as 'a web of data that can be processed directly or indirectly by machines' [14]. The research examines ways of structuring information, and enabling processing and searching of the information to provide a modelling capability.

UDM could help increase user involvement in software, by providing templates to enable non-programmers to develop modelling software for the purposes that interest them. If more users of software are involved in creation of software and the source of the code is open, this allows for the

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 are using current software languages.

Criteria necessary for User Driven Model Development

This section explains the theory behind the User Driven Modelling approach, and the factors necessary to make this approach possible. For this research 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. Research has examined ways of structuring information, processing and searching this information to provide a modelling capability. Research by Aziz et al. [15] examines how open standards software can assist in an organisations collaborative product development, and Wang et al. [16] outline an approach for integrating distributed relational database systems. Our automated production of software containing recursive Structured Query Language (SQL) queries enables this. This approach is a type of very high level Meta-programming. Meta-programming, and structured language is explained by Dmitriev [17] and Mens et al. [18]. The approach proposed 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. The creation of a web based visual representation of the information will allow people to examine and agree on information structures.

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 to use in the representation 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. This is explained in Tollis [19].

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.

This visualisation of results is essential to express clearly their meaning. Words in a report document can be ambiguous. So the relationship of results to inputs must be clearly shown.

Translation Mechanism

Figure 2 shows the translation mechanism.

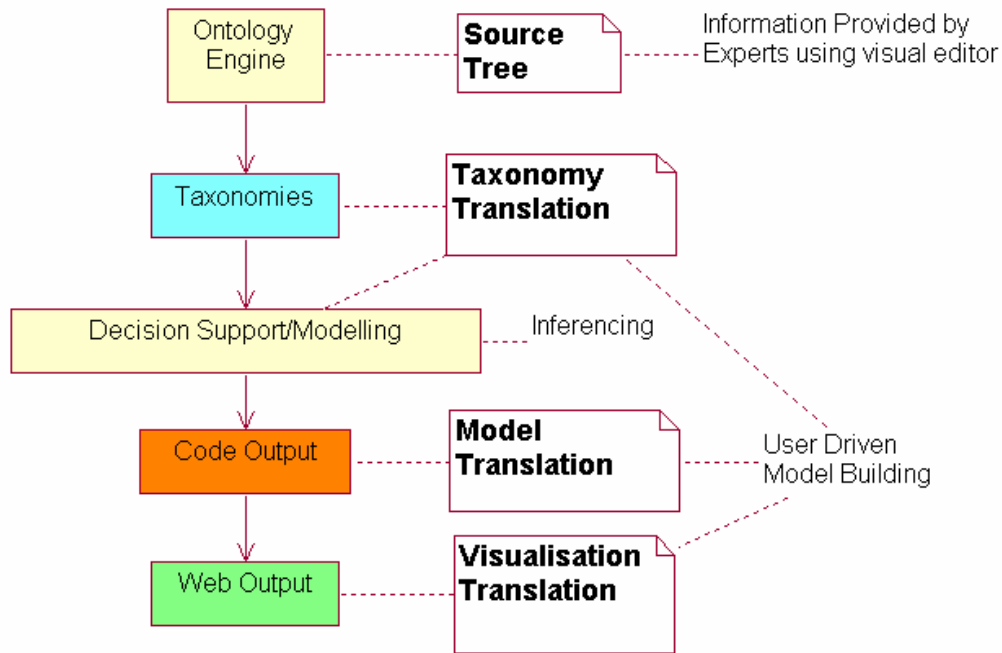


Figure 2 - Translation Process

For this research 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 [20]. The information is highly structured. Visualisation of this structure in order to represent the relationship between things clarifies the semantics. The meaning can be seen not just by the name of each item but also by the relationship of other items to it. It is envisaged that this taxonomy will provide a design costing capability, but the taxonomy and the techniques used to put it together could be re-used for other purposes. Eventually this taxonomy could become part of an overall ontology. At first this would be a light-weight ontology and this could be evaluated for usefulness before deciding on whether it would need to be more structured. Hunter [21] evaluates engineering ontologies and gives examples. Issues involved in visualisation of light weight ontologies are examined by Fluit et al. [22]. An important reason for creation of an open standards central ontology is that it can be accessed by many different applications. The open standard OWL (Web Ontology Language) is explained by Bechhofer and Carroll [23]. Research of others in this field has been investigated Corcho and Gómez-Pérez [24], Corcho et al. [25], and Noy [26].

The approach 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 taxonomy 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. Current research is on provision of a translation mechanism to convert information or models into other languages (primarily web based), and to visualise this information. This mechanism has been used in projects with two major aerospace companies. Examples of this are shown later in the paper.

The alternative approach involves creation of an elaborator that can output code, in various computer languages or a Meta-programming syntax such as metaL [27] or Simkin [28]. 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.

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 this. 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.

The translation mechanism is illustrated in the next section using a simple example.

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.

The approach explained in this section 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 decision support software that is used as a calculation engine and translator.

The component is created, Input Values are defined and these can be used in a calculation. First the length of the rectangle is defined. This is given a value of 4 metres. Next the width of the rectangle is defined. This is given a value of 2 metres. 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. Area is assigned a value of Length * Width. This is a simple equation that will be used to calculate the result. Calculations are all defined by equations that relate attributes of the taxonomy. Figure 3 shows this.

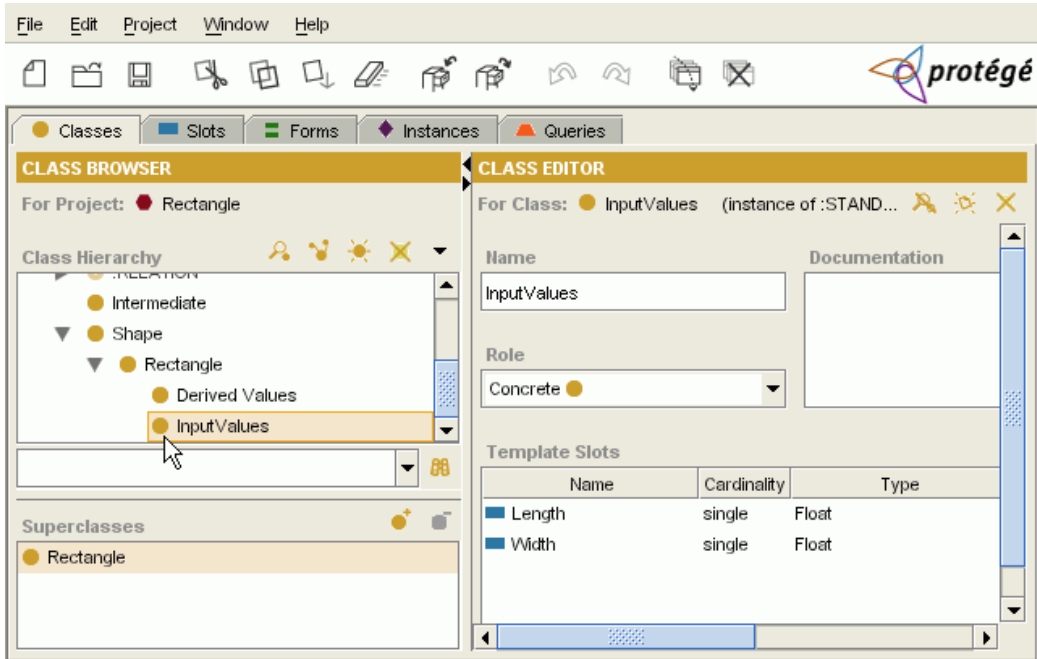


Figure 3 - Rectangle Explanation - Protégé

The taxonomy is translated and stored in a relational database that maps the structure. The taxonomy can be read from the database by the decision support system DecisionPro. Generic code written using the DecisionPro editor reads from this database and recreates the tree. This represents the task of the software developer in providing the infrastructure for the users/model developers. 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. The calculation uses the equation(s) defined in the Protégé editor, which relate the nodes in the taxonomy tree. Figure 4 shows this.

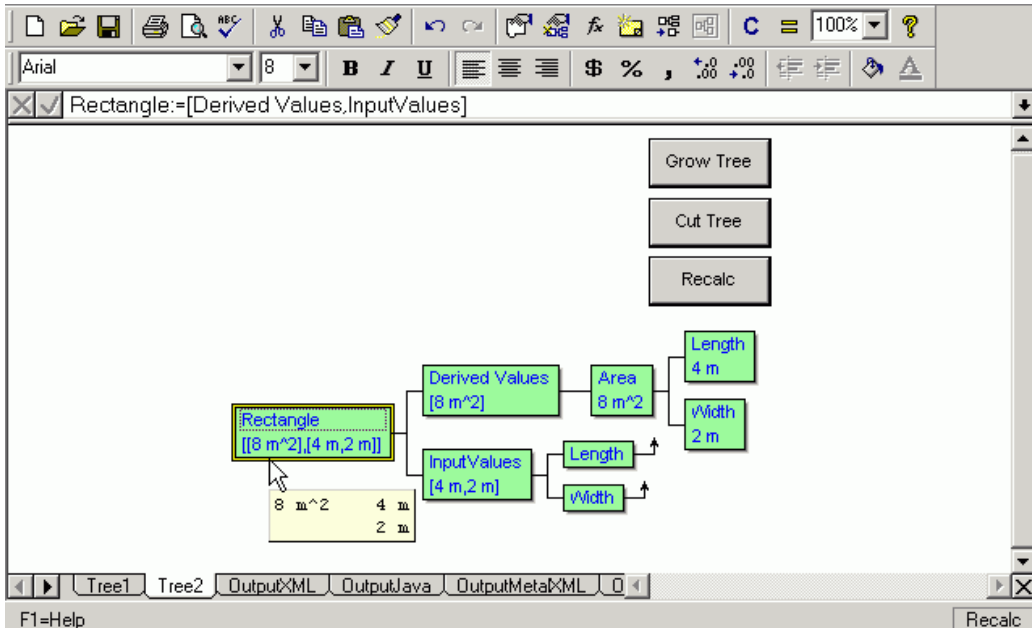


Figure 4 - Rectangle Explanation - DecisionPro

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. The tree is then displayed in a browser, and it's possible to click on individual items to view the item, and details such as the equation and result. Figure 5 illustrates this.

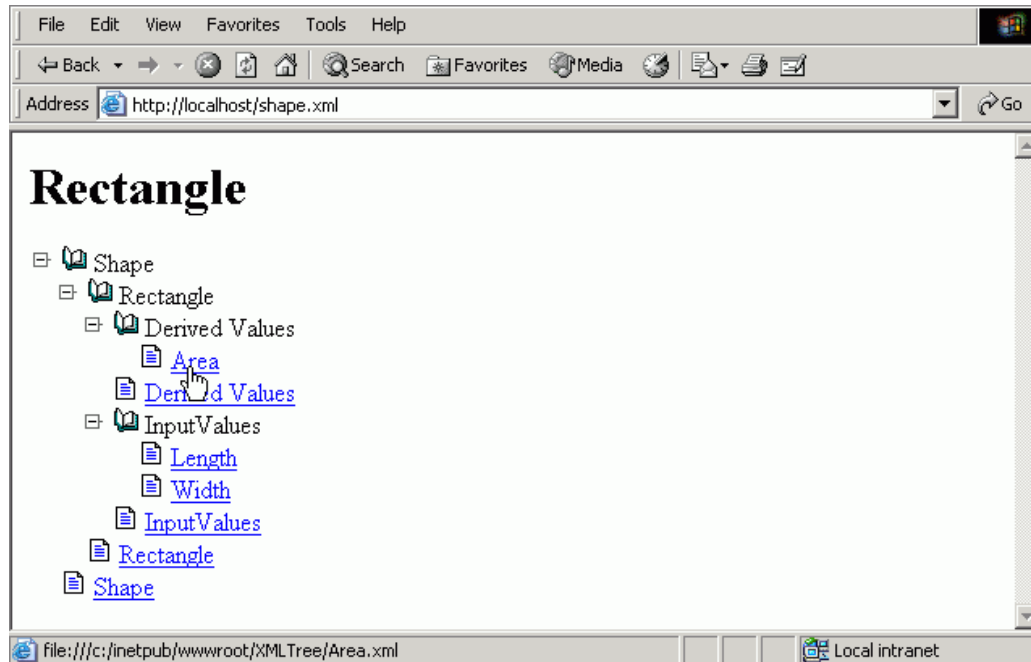


Figure 5 - Rectangle Explanation - Web View Tree

Sometimes an alternative is needed to the tree view of a taxonomy such as a diagrammatic representation of the object. This alternative can be displayed on the Web page using SVG (Scalable Vector Graphics). SVG is an XML format. Interactivity is added to help visualisation, and allow for inputs to be changed dynamically.

Figure 6 shows an output SVG rectangle diagram that includes interactivity. This has been translated from the tree-based representation. The input values used for the calculation and the diagram itself can be changed via an automatically produced user interface that is related to the taxonomy structure.

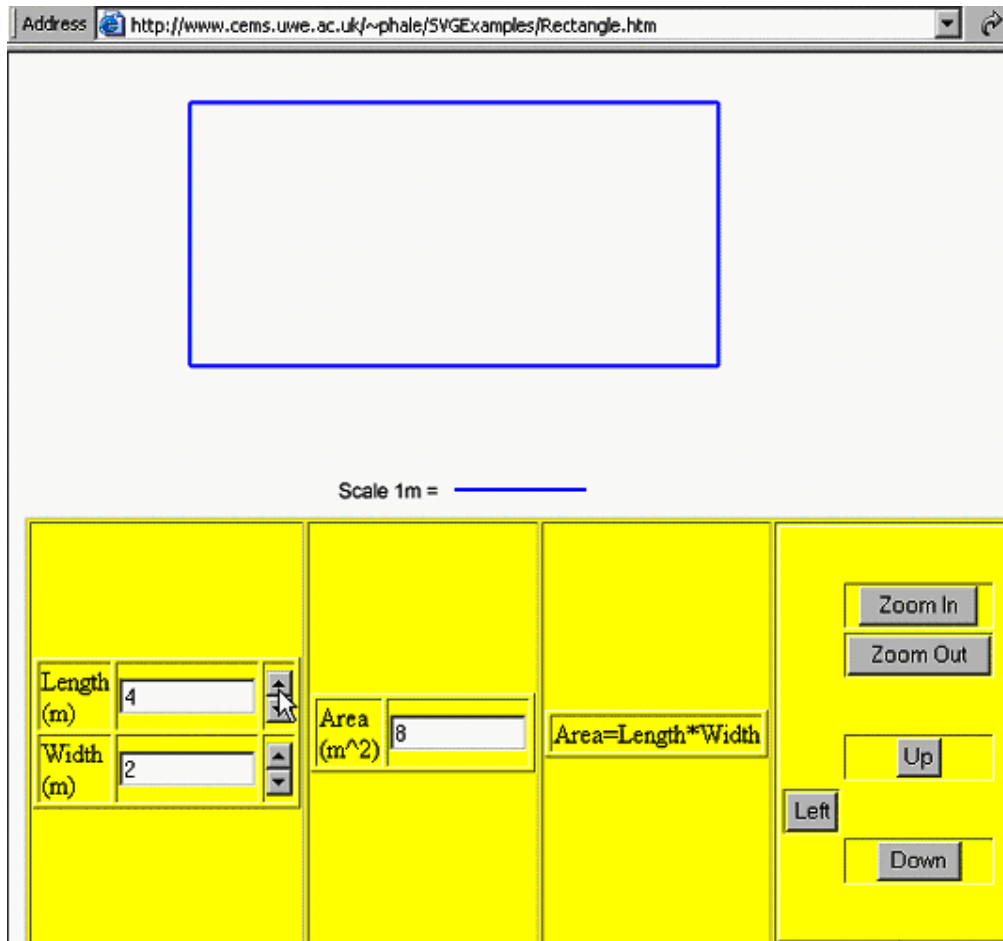


Figure 6 - Rectangle Explanation - SVG Diagram View

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, and there are other controls for moving and resizing the diagram.

A flash movie illustrating this example is available at http://www.cems.uwe.ac.uk/~phale/RectangleDemo/RectangleDemo.viewlet/RectangleDemo_launcher.html.

Summary

This example shows that it is possible to create models visually without the user needing to type code. Calculations and translations can be 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. The ontology editor can be used to produce anything from a simple taxonomy like this, to a large and complex ontology.

Problems with spreadsheet costing models

This section is based on work undertaken 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 will be argued that this approach is insufficient for providing generic and reusable models. This is contrasted with the alternative investigated of User Driven Modelling, this work is explained using examples.

ACCS (Aerospace Composite Costing System)

The ACCS spreadsheet estimation tool was created for a project to provide a large aerospace customer with a comparative cost for the manufacture of the various wing box parts using carbon-fibre composite. The example illustrates a design that is difficult to cost because of a change in process i.e. use of composites rather than metal for the manufacture of a wing. In the 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. The same is true for new processes such as superplastic forming and high speed machining, whose characteristics are significantly different from previously used processes. Therefore parametric models based on processes used already cannot be re-applied for the new processes, as no historical information is available.

The project was to create software for the purpose of costing a product where parametric costing was not viable. The customer specified that this should be a short project to allow the costing of manufacture of a composite wing box, and that a spreadsheet must be used for this due to the 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. This spreadsheet proved to be successful, but not re-usable for other problems.

The composite wing spreadsheet allowed engineers to evaluate the options for the design and manufacture of composite wing box components. It covered four main components - Skins, Spars, Ribs, Stringers and possible manufacturing techniques for each. Visual Basic code was used in the spreadsheet to provide navigation and constrain the navigation order to ensure decisions were made in the correct order. However there were problems that this did not solve that will emerge in the longer term. These problems relate to maintenance, extensibility, ease of use, and sharing of information.

Maintenance

If 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 are not editable. This protection is advanced and responsive. 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. The spreadsheet is also very large, and complex auditing, or updating all the formulae would be a major task.

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 the relevant values and formulae to change.

Ease of Use

Users can find their way around the sheets as the navigation paths are made clear using software that guides them, but the overall structure of the information in the spreadsheet is not clear.

Sharing of Information

It might become necessary to export the information from the spreadsheet to a process-planning tool for example. The lack of structure in the information makes it difficult to export it to other software systems. This makes it hard to make use of tree based W3C (World Wide Web Consortium) standard languages, such as XML (eXtensible Markup Language) or RDF (Resource Description Framework).

Schrage [29] 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 harder to integrate the model with other software applications. Knowledge sharing is essential

for collaboration. Merlo and Girard [30] explain the necessity for collaborative information systems for designers.

User Driven Modelling Implementation

The example used to illustrate the new approach uses information taken from the composite wing box spreadsheet.

Implementation Example – Spar Hand Lay-Up Process

The implementation creates decision support programs automatically in response to user choices. The translation then creates programs in other computer languages, which allow presentation of results. The basis of this is that elaborators are nodes in the tree, which are automatically created and dynamically write objects. This allows our 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 or visualisation.

Taxonomies are created in Protégé for Parts, Materials, Consumables, Processes, Rates, and Tooling for a prototype costing system. 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 7 illustrates how code is produced from the semantic relationships.

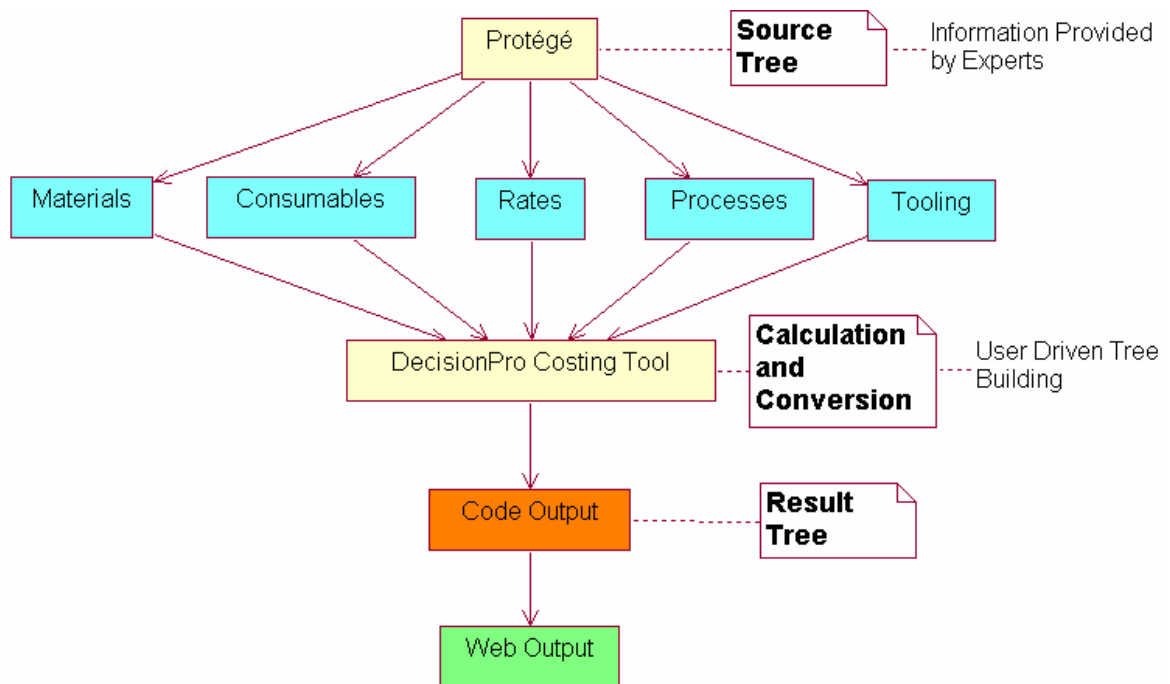
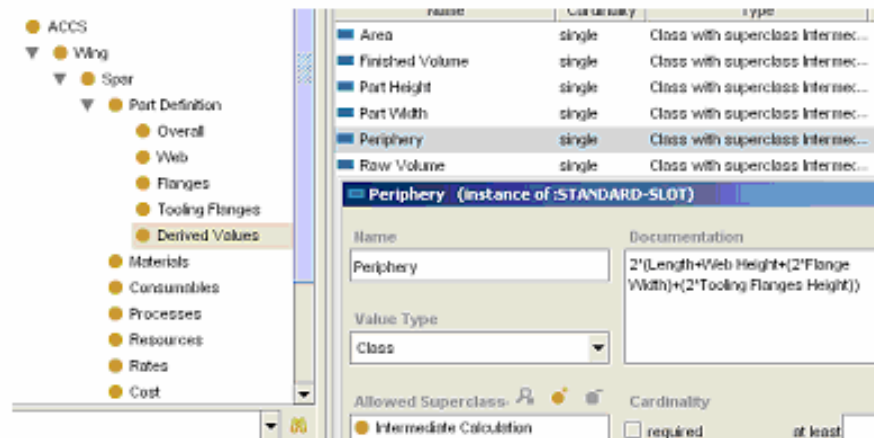


Figure 7 - Rectangle Explanation - SVG Diagram View

Figure 8 shows the equation as text in the Documentation field of the Periphery attribute of Derived Values. This equation text is translated into the DecisionPro visualisation.



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

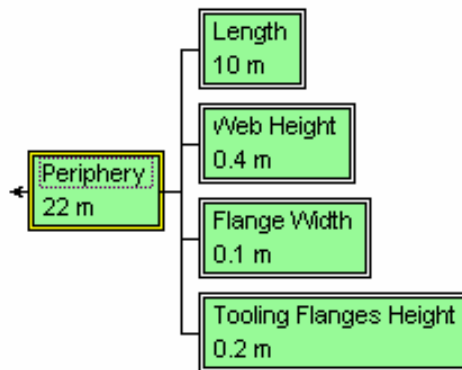


Figure 8 - Spar Periphery Calculation translated from Protégé.

For the prototype to be extended and applied for practical use, each taxonomy would be filled with a structured tree representation of experts' knowledge in the form of classes, values and equations. A costing tree can be automatically produced from these taxonomies. Equations created by the expert, together with choices made by the user of the decision support software, determine how these taxonomies are linked for a particular costing. The costing tool user would then determine which costing equations are used, by answering questions on dialogue forms. These questions would be asked whenever multiple solutions were 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. Figure 9 shows the top level of the user interface created automatically by our DecisionPro software from the Protégé taxonomies.

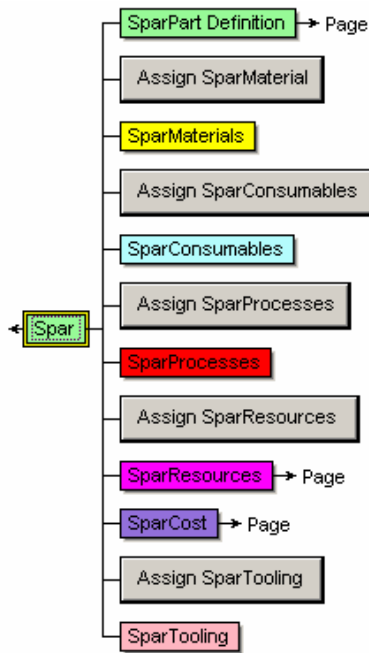


Figure 9 - Interface automatically generated from Protégé

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

DecisionPro visualises large trees by breaking them into individual pages, and indicating with a right arrow where there are further pages that can be viewed. Clicking the 'Part Definition' right arrow will display the corresponding information. The 'Derived Values' branch contains parameters of the spar that are calculated from the spar dimensions. Figure 10 shows this.

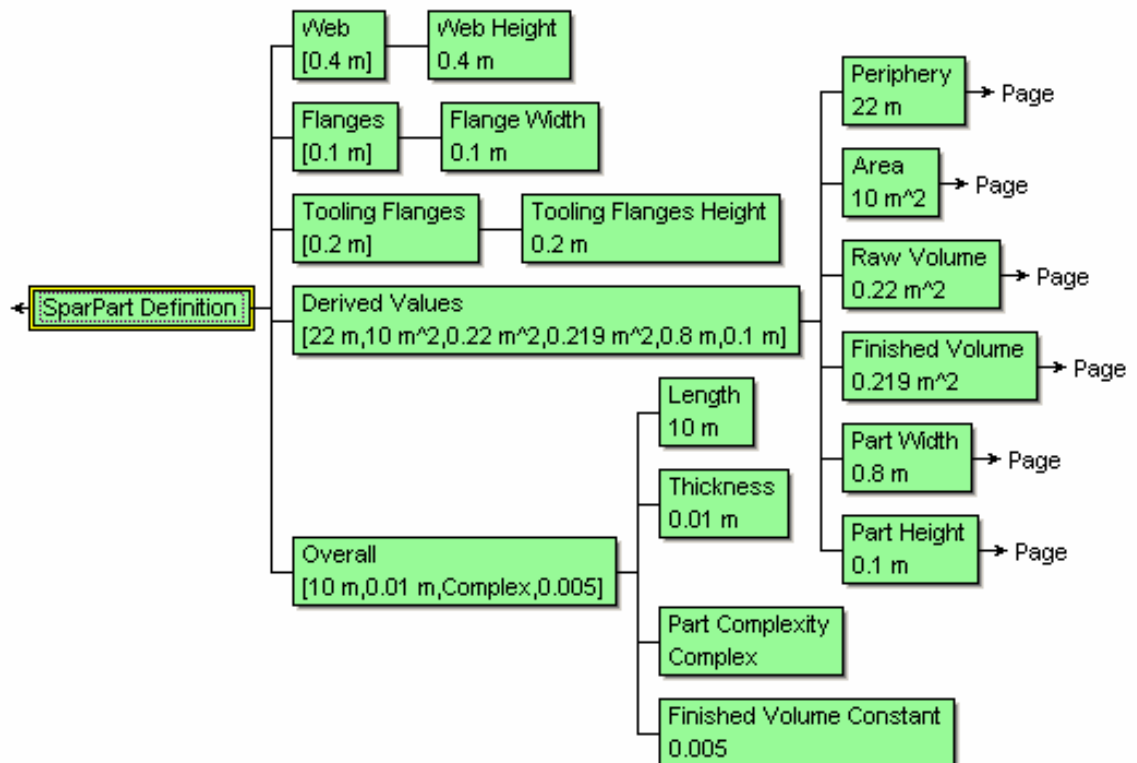


Figure 10 - Interface automatically generated from Protégé

Different types of information indicated by colour coding may be combined in a calculation. This is illustrated in Figure 11. 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.

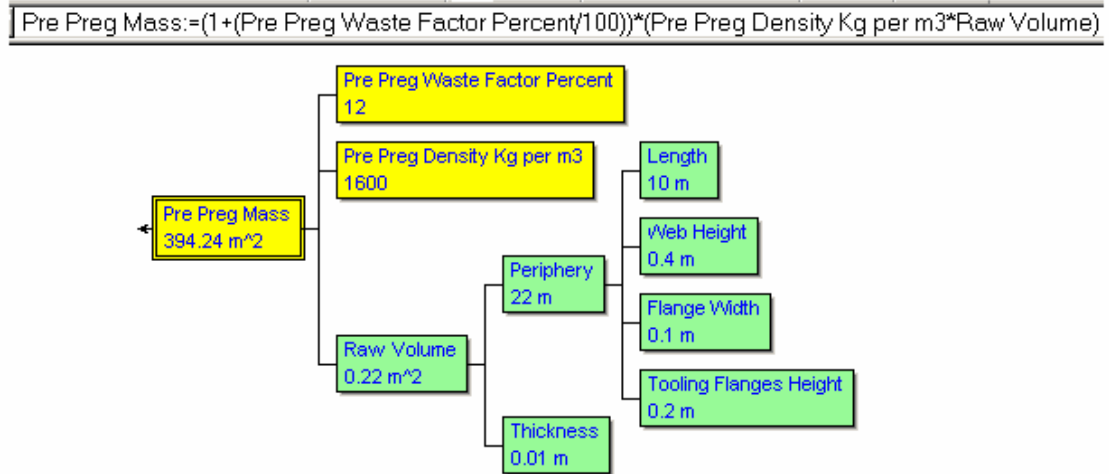


Figure 11 - Use of Colour Coding

Figure 12 shows the DecisionPro tree translated into XML and displayed as a web tree using a stylesheet. The menu uses a stylesheet created by De Andreis [31]. The examples below are available from our page at <http://www.cems.uwe.ac.uk/~phale/>.

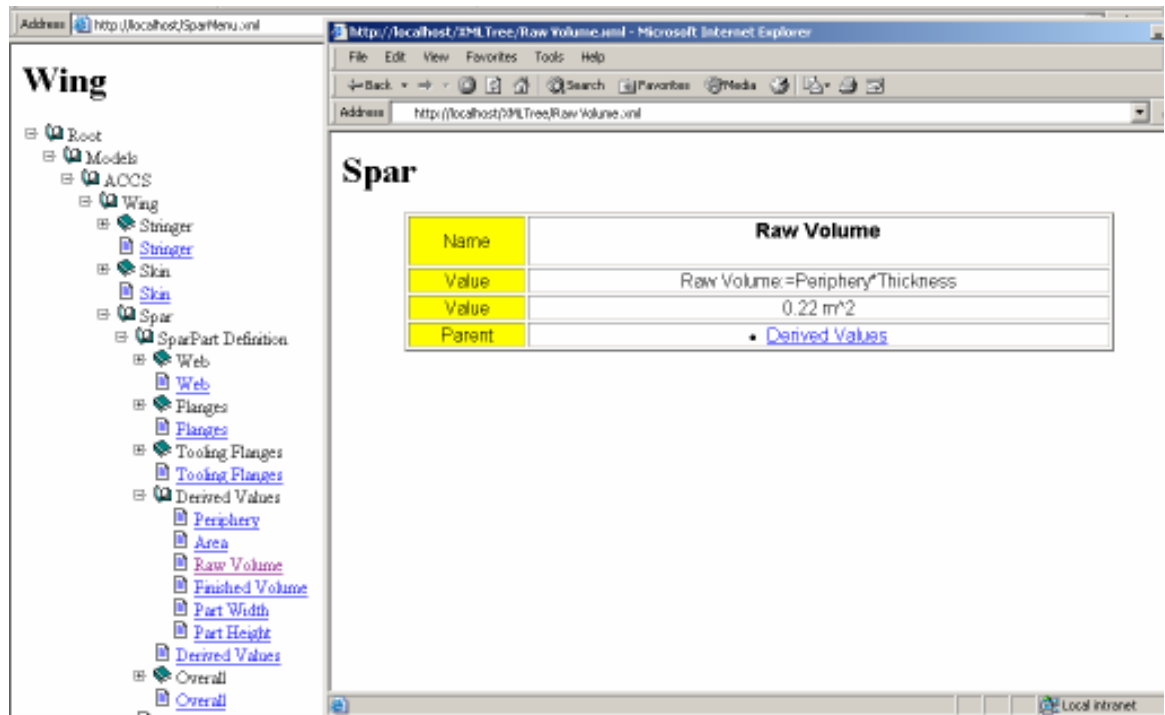


Figure 12 - XML web translation from Decision tree

The production of part diagrams using SVG can be automated in a similar manner to that used for the automated production of DecisionPro costing models. Figure 13 shows an example of such an

interactive visualisation of a Spar. This interactive diagram was output automatically from the part definition described in the part taxonomy.

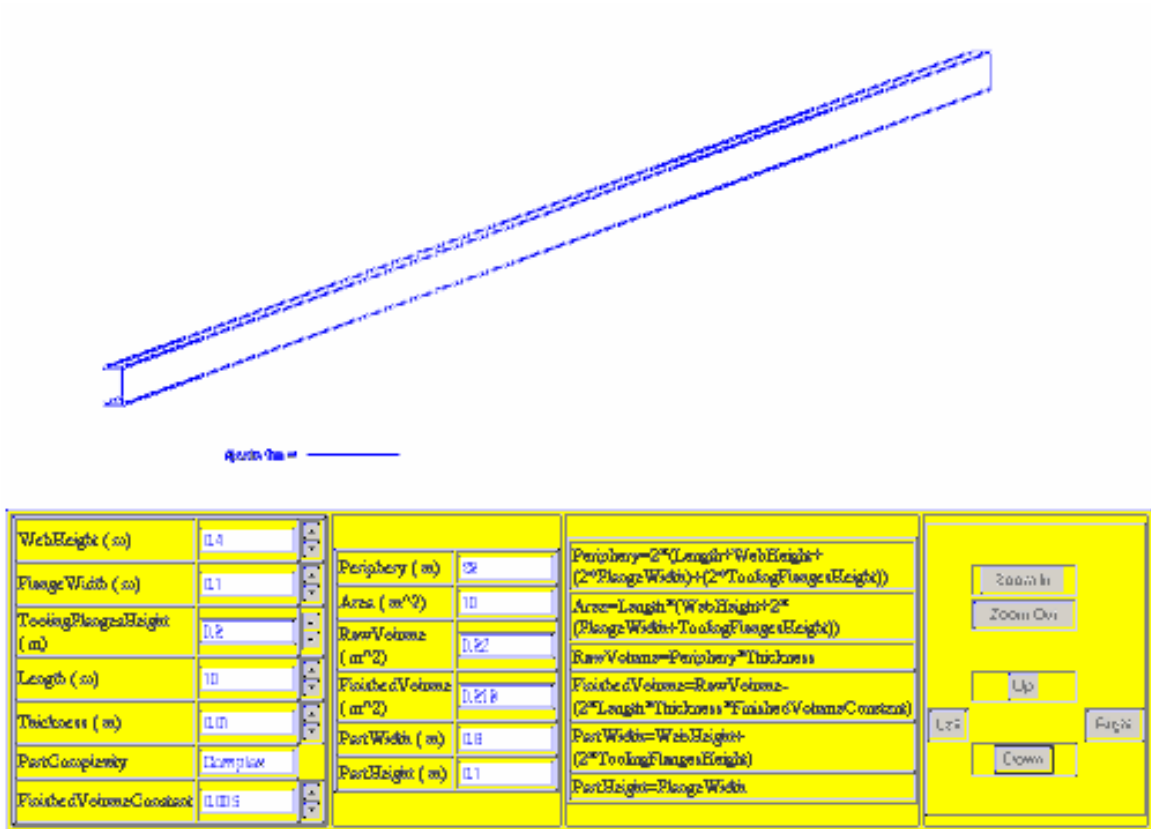


Figure 13 - Interactive Spar Diagram (SVG)

The way the user can interact with the diagram can be seen by viewing these examples on our Interactive SVG links page at <http://www.cems.uwe.ac.uk/~phale/InteractiveSVGExamples.htm>. These provide examples of wing box components.

The XML can also be displayed on the web using a Flash program created by Rhodes et al. [32]. This creates a tree with a three dimensional look and a use of colour, shading, and movement of the nodes that makes it an intuitive user interface that is easy to navigate. 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 14.

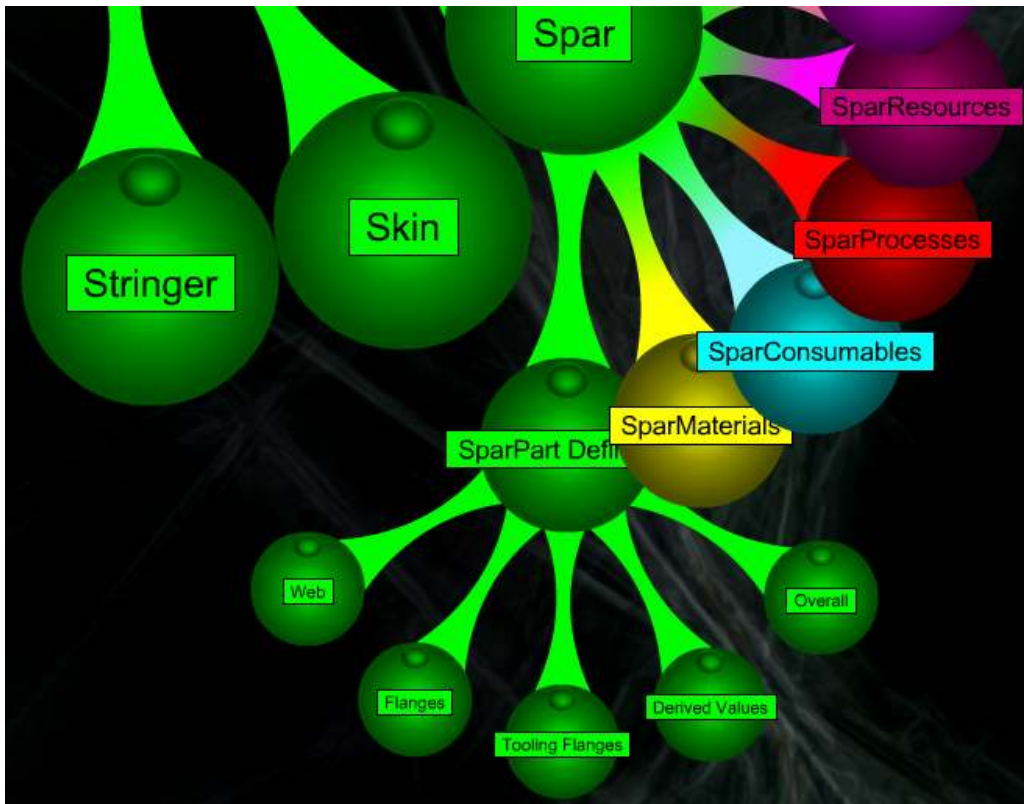


Figure 14 - Flash interface for navigating exported XML tree

The representation can also be translated into Java, and Cost Estimator

Conclusion

We compared the approach of developing decision support models for design and costing using a spreadsheet, to the approach of User Driven Modelling, with open standards languages, and a web interface for this purpose. We conclude that the use of spreadsheets for this purpose is beset by problems. These relate to Maintenance, Extensibility, Ease of Use, and Sharing of Information. But an alternative is required so modellers can continue their work without having to rely on software developers. The alternative approach involves development of a system, where a model builder, via visual editing of library taxonomies can undertake maintenance and extension of information. As yet a prototype only has been created for this, and a great deal more work is required to finalise the structure for the holding of this information, and agree standards for terminology. Also we need to provide more validation of user input. However dealing with this proof of concept has indicated to us that it is far easier to maintain, search and share information using this approach than it is using spreadsheets. The ability to visualise, search and share the information using structured languages and web-pages, is a huge advantage, and allows us to create dynamic image views and decision support models over the web.

References

- 1 W3C. (2006). W3C World Wide Web Consortium, <http://www.w3.org/>.
- 2 Stanford University. (2006). Stanford University - Welcome to protégé, <http://Protégé.stanford.edu/>.
- 3 Vanguard. (2006). Vanguard Home Page, <http://wiki.vanguardsw.com/>.
- 4 Engineous Software. (2006). Specification Version 1.6, http://www.engineous.com/product_FIPER_specifications.htm, Engineous Software.
- 5 Olsson, E. (2004). What active users and designers contribute in the design process, *Interacting with Computers*, 16: 377-401.
- 6 Scaffidi, C., Shaw, M., Myers, B. (2005). Estimating the Numbers of End Users and End User Programmers, *IEEE Symposium on Visual Languages and Human-Centric Computing. (VL/HCC'05): 207-214 Dallas, Texas.*
- 7 Myers, B., Ko, A., Burnett M. (2006). Invited Research Overview: End-User Programming Extended Abstracts, *CHI'2006: pp. 75-80, Montreal, Canada.*
- 8 Erwig, M., Abraham, R., Cooperstein, I., Kollmansberger, S. (2006). Automatic Generation and Maintenance of Correct Spreadsheets?, http://web.engr.oregonstate.edu/~erwig/papers/Gencel_ICSE05.pdf.
- 9 EUSES. (2006). End Users Shaping Effective Software. (2006). <http://eusesconsortium.org/>.
- 10 Jena. (2006). A Semantic Web Framework for Java, and Protégé, <http://jena.sourceforge.net/>.
- 11 Huhns, M. (2001). Interaction-Oriented Software Development. *International Journal of Software Engineering and Knowledge Engineering*, 11: 259-279.
- 12 Paternò, F. (2005). Model-based tools for pervasive usability. *Interacting with Computers*, 17(3): 291-315.
- 13 El-Ghalayini, H., Odeh, M., McClatchey, R., Solomonides, T. (2005). Reverse engineering ontology to conceptual data models, <http://www.uwe.ac.uk/cems/graduateschool/news/posters/conference/conference2005.pdf>, Graduate School Conference, 114-119.
- 14 Berners-Lee, T. (1999). Weaving the Web, Orion business - now Texere, <http://www.w3.org/People/Berners-Lee/Weaving/Overview.html>.
- 15 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: 260-271.
- 16 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: 205-322.
- 17 Dmitriev, S. (2004). Language Oriented Programming: The Next Programming Paradigm, <http://www.onboard.jetbrains.com/is1/articles/04/10/lop/>.
- 18 Mens, K., Michiels, I., Wuyts, R. (2002). Supporting Software Development through Declaratively Codified Programming Patterns, *Expert Systems with Applications*, 23: 405-413.
- 19 Tollis, I. G. (1996). Graph Drawing and Information Visualization, *ACM Computing Surveys*, 28A(4).

- 20 Ambler, S. W. (2003). The Object-Relational Impedance Mismatch, <http://www.agiledata.org/essays/impedanceMismatch.html>.
- 21 Hunter, A. (2002). Engineering Ontologies, <http://www.cs.ucl.ac.uk/staff/a.hunter/tradepress/eng.html>.
- 22 Fluit, C., Marta S., Harmelen F. V. (2003). Supporting User Tasks through Visualisation of Lightweight Ontologies, <http://www.cs.vu.nl/~frankh/abstracts/OntoHandbook03Viz.html>.
- 23 Bechhofer, S. and Carroll, J. (2004). Parsing owl dl: trees or triples?, Proceedings of the 13th international conference on World Wide Web, NY, USA: 266 - 275.
- 24 Corcho, O. and Gómez-Pérez, A. (2000). A Roadmap to Ontology Specification Languages, Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management, Chicago, USA.
- 25 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: 41-64.
- 26 Noy, N.F. (2004). Semantic Integration: A Survey Of Ontology-Based Approaches. SIGMOD Record, Special Issue on Semantic Integration, 33 (4).
- 27 Lemos, M. (2006). MetaL: An XML based Meta-Programming language, <http://www.meta-language.net>.
- 28 Simkin. (2006). A high-level lightweight embeddable scripting language which works with Java or C++ and XML, <http://www.simkin.co.uk/>.
- 29 Schrage, M. (1991). Spreadsheets: Bulking Up On Data. <http://www.systems-thinking.org/buod/buod.htm>, Los Angeles Times.
- 30 Merlo, C. and Girard, P. (2004). Information system modelling for engineering design coordination, Computers in Industry, 55: 317-334.
- 31 De Andreis, E. (2005). 2MXtree XML-based tree, <http://manudea.duemetri.net/xtree/>.
- 32 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, <http://www.friendsofed.com/books/1590591585/>.