

Enabling Decision Support and Costing of Product Designs by using Visual Metaphors

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 within the aerospace sector.

This paper outlines the technique of User Driven Modelling. 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 ontology tool Protégé from Stanford University. This ontology can be translated into a Decision Support tool called DecisionPro (now renamed Vanguard System), 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. It is possible to search the information both in Protégé and on the web as it is represented using searchable Semantic Web languages.

Keywords

Visual Programming, Visualisation, Translation, Transformation, Meta Programming, Cost Modelling, Modelling, Decision Support, Design, Manufacture, User Driven Modelling, Semantic Web, End User Programming.

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. We use open standards wherever possible as agreed by organisations such as the World Wide Web consortium (W3C) [1], OASIS [2], and NIST [3]. 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 also 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 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. Functional programming involves the evaluation of mathematical functions. This is more natural to engineers who often model problems by using functional expressions on a spreadsheet, so use of functional expressions simplifies program statements and allows something closer to natural expression. The additional advantage is that of displaying the expressions in the appropriate context. An explanation of Functional Languages, and how these can be used with XML (eXtensible Markup Language) is available in Functional Programming and XML [4].

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 [5]. 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 (the new version is called Vanguard System) [6]. Vanguard are creating a modelling network where universities can share decision support models over a network. This application was used because it was selected during a project to evaluate, and then use software to solve costing problems as part of the DATUM (Design Analysis Tool for Unit-Cost Modeling) project [7].

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 Java, and Engineous' Java based costing tool Cost Estimator [8]. It is possible to search the information both in Protégé and on the Web as the information is represented using searchable Semantic Web languages. We are evaluating Semantic Web representation and search applications; these are explained later in the paper.

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 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.

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. 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 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. The aim of using this knowledge is 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 [9]. The Model Driven Semantic Web [10] explains the opportunities for and importance of this kind of research. The approach encourages much greater user involvement in software development. Tim Berners-Lee defined the Semantic Web as 'a web of data that can be processed directly or indirectly by machines' [11]. The research examines ways of structuring information, and enabling processing and searching of the information to provide a modelling capability. The advantages of increasing user involvement in software development are explained by Olsson [12].

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 construct or reuse ontologies - to represent the concepts used in a model. 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 be 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 has led to a spreadsheet crisis where incorrect and undocumented spreadsheets are produced by individuals who have little formal training DATUM [7] Erwig et al. [13]. Steps can be taken to correct this problem, 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) [13]. Despite such work creation of correct spreadsheets is difficult when the structure of the relationships in the spreadsheet is not clearly visible, this is why we are looking for alternative representations and visualisations.

Currently, to be able to model a complex problem, the users/domain experts 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. We call this 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.

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 mainly in the USA [14] and Network of Excellence on End User Development in Europe [15], and by the Institute for End User Computing [16]. Fabio Paternò has investigated this subject as part of the End User Development in Europe network Paternò [17].

This graph of programmers in the US, shows that users are by far the biggest group, many of these develop their own programs, there are two intermediate groups, and a small group of professional programmers.

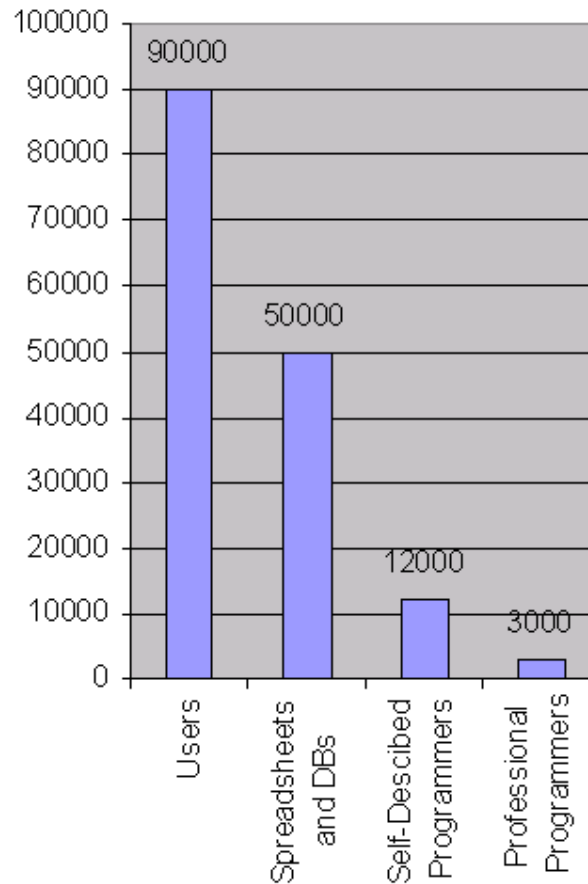


Fig. 1. US programmer numbers.

Based on data from US bureau of Labour Statistics.

Source [18] Scaffidi et al.

Research Approach

The ontology is made up of taxonomies for each domain such as parts, processed, 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). A representation of PSL as XML can be found at PSL-XML [19]. PSL-XML can be used with OWL, both OWL and PSL-XML can be based on RDF/XML (Resource Description Framework) implemented with XML (eXtensible Markup Language). The open standard OWL (Web Ontology Language) is explained by Bechhofer and Carroll [20]. This makes our translation process simpler and can enable interaction with other systems. This kind of solution is referred to in 'Ontologies and Semantics for Seamless Connectivity' [21].

Our research demonstrates how taxonomies 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 Development that could be part of a wider approach of User Driven Programming. 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 management systems such as Jena [22], and Protégé [23]. 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. This is so the user does not need 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 information.

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 DecisionPro (Vanguard System) 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. Standardised 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. Also the 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 a model must be represented differently from the taxonomy which represents it. So XML can be used both as the information processed by the application, and the application itself. This enables a recursive relationship between the model and the taxonomy. This recursion makes 'meta-programming' possible. Meta programming is the 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.

This research is influenced by dynamic software systems such as outlined by Huhns [24]. 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 [17] 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 an interface that consists of interaction objects. This approach can be used for automating the design of the 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

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 mis-understandings 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.

It may also prove possible to work in reverse from implementation to design, or design to conceptual model. Ontologies could be mapped to conceptual models, El-Ghalayini Et al. [25]. 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 that software is designed well or for 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 every day 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.

User Driven Modelling (UDM) could also 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 are much more able to attain the expertise to develop their own models than they are using current software languages. An example page has been created [26].

Criteria necessary for User Driven Model Development

This section explains the theory behind the User Driven Modelling approach, and the factors necessary to make the 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, and processing and searching this information to provide a modelling capability. Research by Aziz et al. [27] examines how open

standards software can assist in an organisations collaborative product development, and Wang et al. [28] outline an approach for integrating distributed relational database systems. Our automated production of software containing recursive SQL queries enables this. This approach is a type of very high level Meta-programming. Meta-programming, and structured language is explained by Dmitriev [29] and Mens et al. [30]. Techniques such as Model Driven Programming [10][31], Generative Programming [31][32], Aspect Oriented Programming [33][34] are being evaluated for creation of the modelling environment for End Users. 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 [35]. 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 by Tollis [36].

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 transformed. 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 visualisation 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 is usually insufficient indication as to how the results were calculated. So the relationship of results to inputs must be clearly shown.

Translation Mechanism

The diagram below illustrates the aim of having a two way translation between all levels in a hierarchy of translation between human and computer, and between different software environments. This definition used in this paper by Simons and Parmee [37] explains the aim - '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 in Watch What I Do: Programming by Demonstration [38]. 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.

Translation between the ontology and representations of it can enable the information to be visualised and edited, in different ways and using various software tools as appropriate. This means models can be created, edited, and viewed in a way most suitable to particular users. Figure 2 shows the translation mechanism.

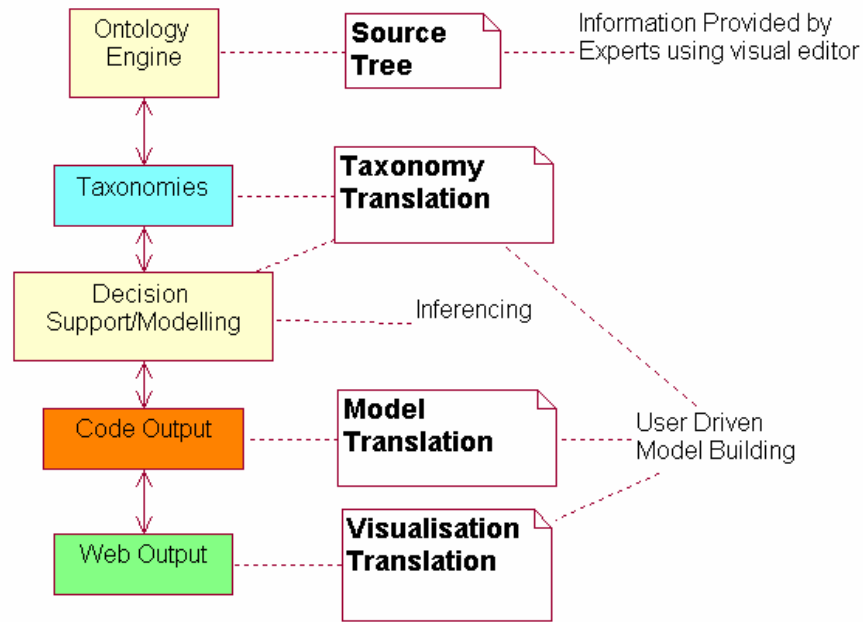


Fig. 2. - Translation Process.

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 [39], and links our research to that of ontology and Semantic web management and searching. 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 [40] evaluates engineering ontologies and gives examples. Issues involved in visualisation of light weight ontologies are examined by Fluit et al. [41]. An important reason for creation of an open standards central ontology, based on OWL [20], is that it can be accessed by many different applications. Research of others in this field has been investigated Corcho [42], Corcho and Gómez-Pérez [43] and Noy [44].

The approach involves adapting or creating software systems to provide the visual editor for the source tree, and model builders can 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 can 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 approach we advocate involves creation of an elaborator that can output code, in various computer languages or a Meta-programming syntax such as MetaL [45] or Simkin [46]. 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 a model builder 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) dependent on decisions or requests made by the user the last time the he or she 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.

Visualisation and Interaction Mechanism

Figure 3 shows the methodology behind the Semantic Web modelling. We have already prototyped all these stages, but have not yet developed a fully working modelling system to be used outside the university.

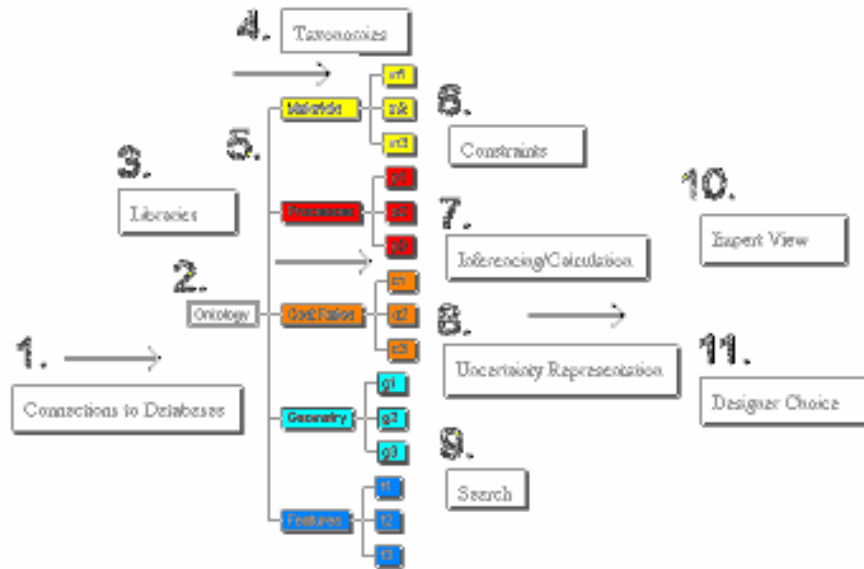


Fig. 3. Visualisation and Interaction Mechanism.

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 [41], and an interface built to allow domain experts to edit the ontology.
3. Libraries are created in a partnership between ourselves and 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, this part of the diagram was built with Vanguard system (explained below). We have created a link between the ontology tool and this decision support and calculation tool. Vanguard system 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 results in 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 we have prototyped including 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 can assist in this.
10. and 11. Experts such as designers can interact with the ontology, the model, and results, it's intended that there will be a two way feedback mechanism where the expert can make changes at any stage, and this filter into changed results. This can then support a cycle of results and rework.

Requirements of Web based Model

The background research has been on Semantic Web techniques that can be applied to this problem. The intention behind this research is to provide a tool that can be used by people who don't have access to CAD tools or other specialist software. It is to aid communication of product information throughout an organisation.

A web based model can be output through the translation mechanism and, with some further work, could allow interactive modelling collaboration. Evidence from related research demonstrates what is required of such a web based visual modelling system, and how it can be effective in aiding, design, manufacture and supply of components.

An open standards web driven method of collaboration is required to make it possible for organisations and individuals to become more deeply involved in projects that are well coordinated using web technologies. Morris et al. [47] examine Interactivity and collaboration on the web. Aziz et al. [48] examine how open standards software can assist in an organisation's collaborative product development. This approach is outlined in Ciancarini et al. [49] that explains ways of designing a document-centric coordination application over the Internet. Nidamarthi et al. [50] explain how web based collaboration can aid the design process. Huang and Mak [51] evaluate issues in the development and implementation of web applications for product design and manufacture. Reed et al. [52] show how web based modelling and simulation can be used in the aircraft design process. Kim et al. [53] explain their approach to modelling and simulation. Zhang et al. [54] review Internet-based product information sharing and visualisation. Li [55] examines the role of web based services for distributed process planning optimization.

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 theory behind this approach is that of showing examples of a program in whatever way most puts across the information in an understandable way. Whenever necessary a translation should be performed from what way of viewing the information to another visualisation more suitable to that need. This must illustrate the concept that the information represents. This allows a user to manipulate the information and get immediate feedback on what has changed. This is related to Programming by Example [37], which is explained below.

Modelling and Simulation can help to ensure that the designers' model, system model and users' model are all the same. This subject is explored in Visualization and the process of modeling: a cognitive-theoretic view [56] and is the basis of the visualisation techniques used to enable the user to create and understand models that are subsequently translated into software representations. In the mid 1970s Smith [57] introduced the technique of Programming by Example with a program called Pygmalion. In Programming by Example[37] (Chapter 1) Smith explains how Pygmalion attempts to bridge the gap between the programmer's mental model of a subject and what the computer can accept. This demonstrated the need to describe algorithms through concrete examples rather than abstractly. Smith went on to develop office oriented icons as part of the Xerox's "Star" computer project. Guibert et al [58] 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. Including the context of the information allows a person to discover meanings or relationships in the information which would not always be obvious. 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 explained in this paper, leading to an improvement in information discovery. This is the theory behind our conversions to interactive SVG (Scalable Vector Graphics) and tree based representations of information and functions [59][60].

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 4 shows this.

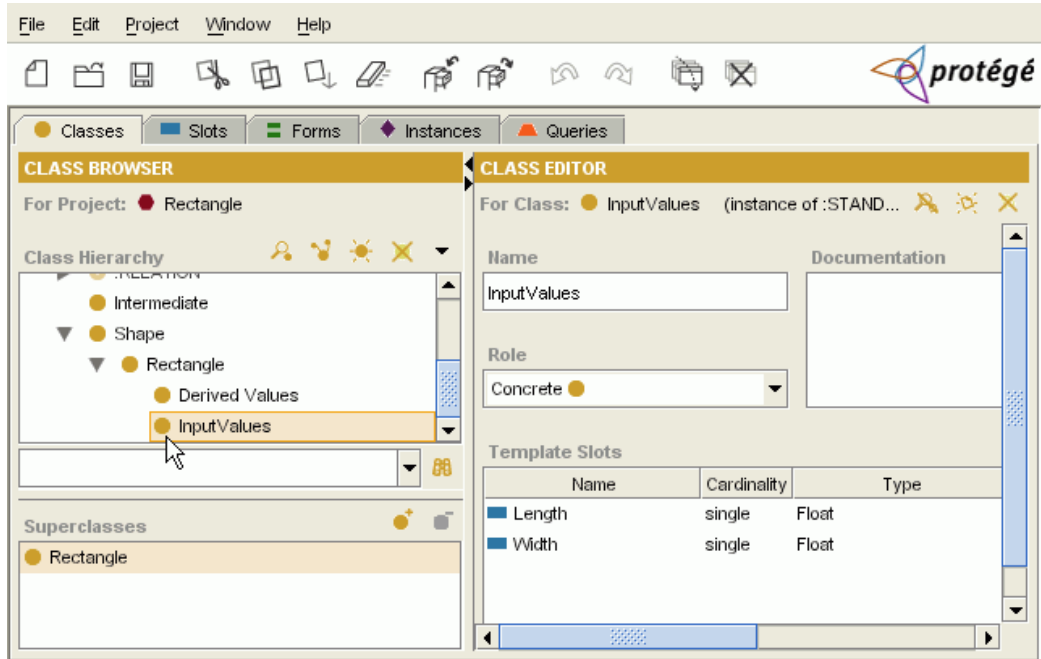


Fig. 4. Rectangle Explanation - Protégé.

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 5 shows this.

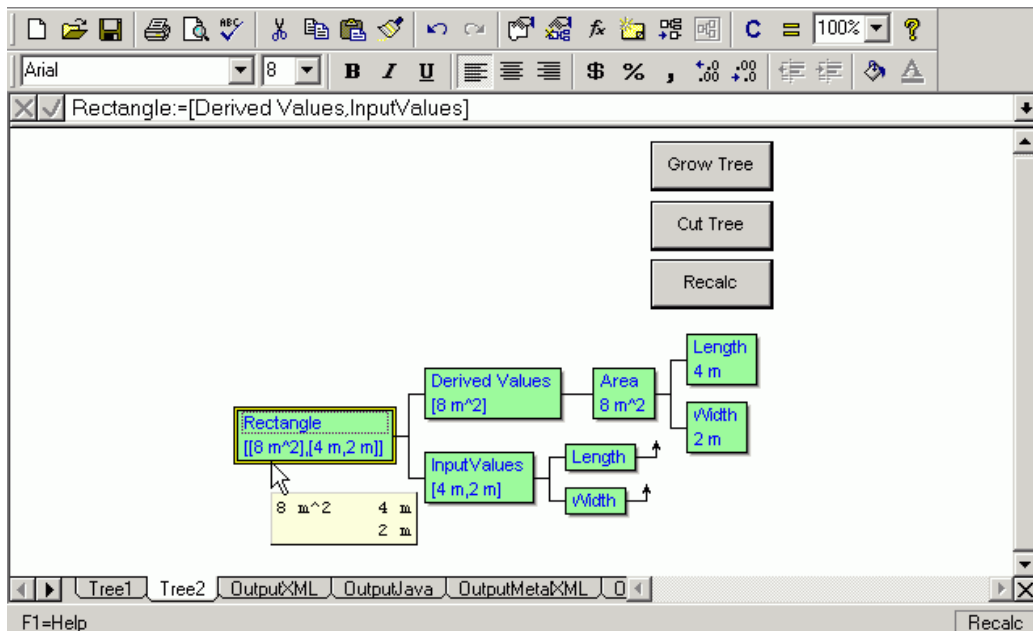


Fig. 5. - Rectangle Explanation - DecisionPro

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 6 illustrates this.

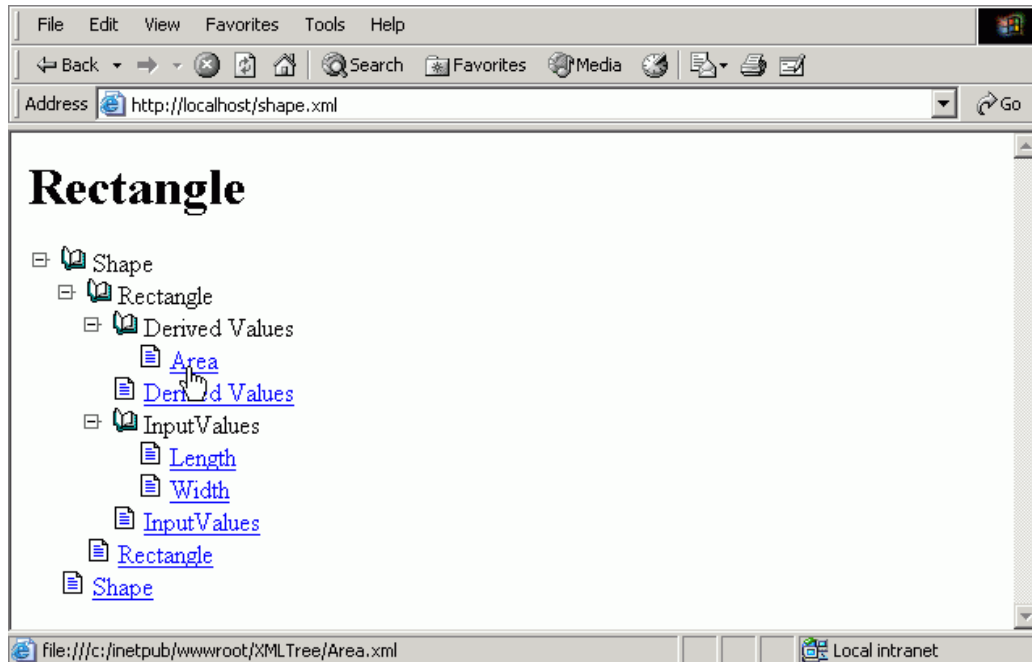


Fig. 6. Rectangle Explanation - Web View Tree.

Sometimes an alternative is needed to the tree view of a taxonomy such as a diagrammatic representation of the object. 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). SVG is an XML format. Interactivity is added to help visualisation, and allow for inputs to be changed dynamically.

Figure 7 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.

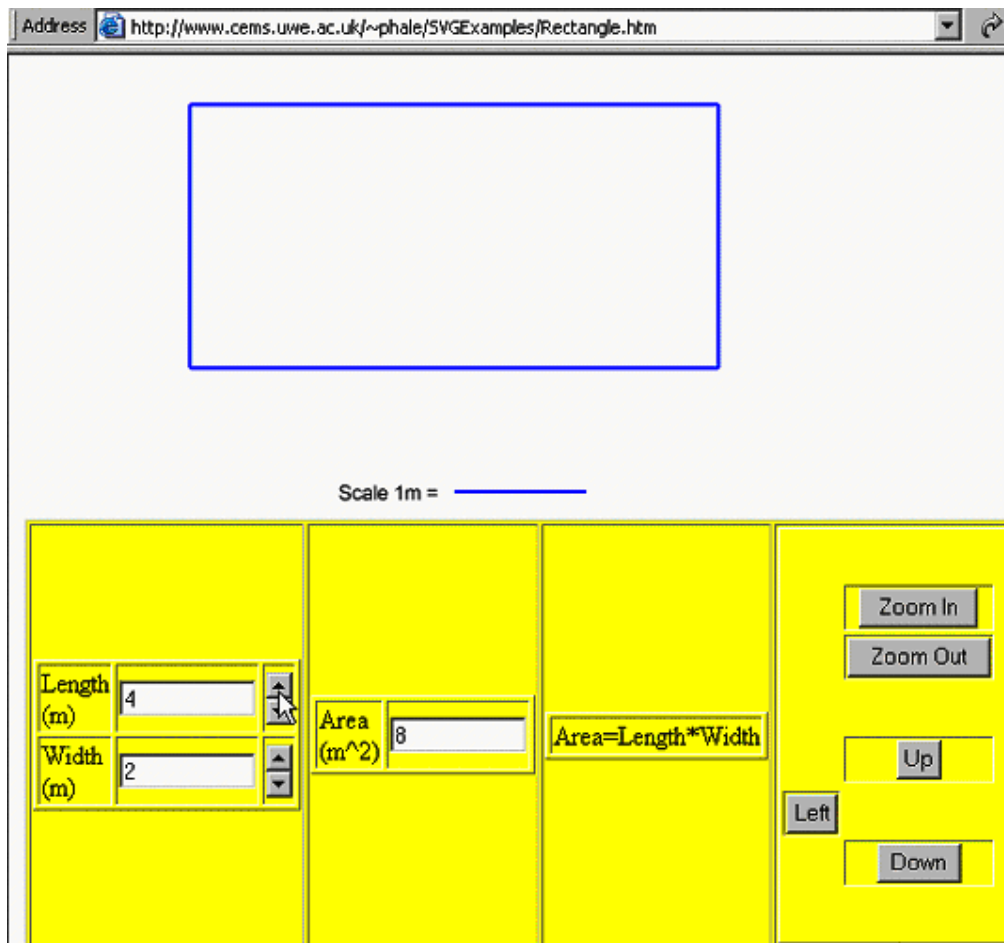


Fig. 7. Rectangle Explanation - SVG Diagram View.

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 [61].

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 visual 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 ACCS example described below, 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 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 widespread availability of this application. 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. The spreadsheet we produced proved to be successful, but not re-usable.

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 to provide navigation between the many sheets. However there were problems that this did not solve. 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. 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. Spreadsheets can export data as XML but spreadsheet models are rarely constructed with this in mind. Applications for modelling complex problems should make it easy and natural to build and visualise trees and graphs that represent the relationship between nodes e.g. [6]. This makes exporting of structured information much more practical.

Schrage [62] 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 [63] explain the necessity for collaborative information systems for designers and the need for an object-oriented approach to this.

These problems can be solved by using the meta-programming and transformation approach that we explained earlier, using the simple rectangle example. This approach is now explained using a complex example.

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 8 illustrates how code is produced from the semantic relationships.

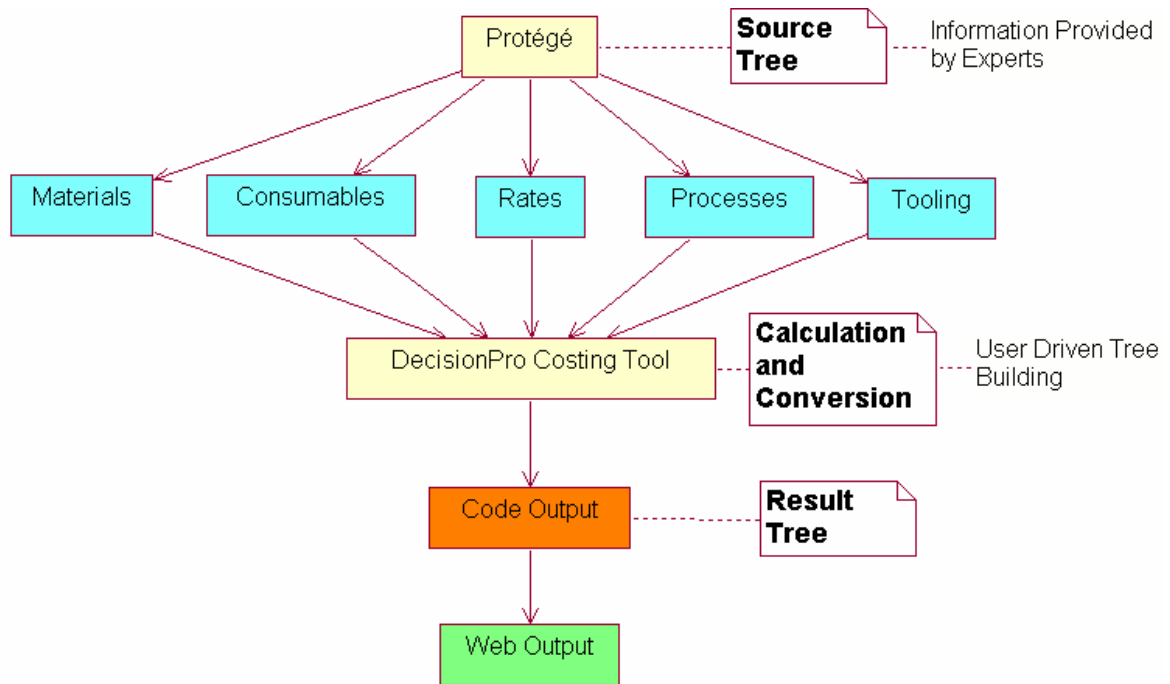
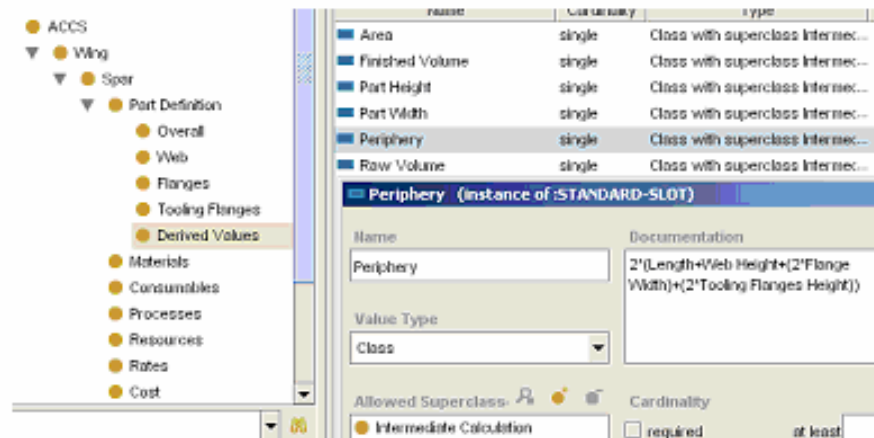


Fig. 8. Rectangle Explanation - SVG Diagram View.

Figure 9 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.



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

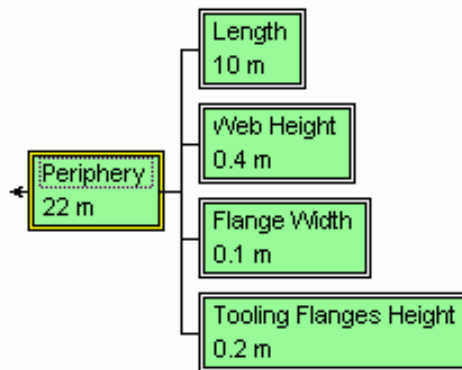


Fig. 9. Spar Periphery Calculation translated from Protégé.

For the prototype to be extended and applied for external 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 produced automatically 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 are 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, and not the user interface or calculation engine. Figure 10 shows the top level of the user interface created automatically by our DecisionPro software from the Protégé taxonomies.

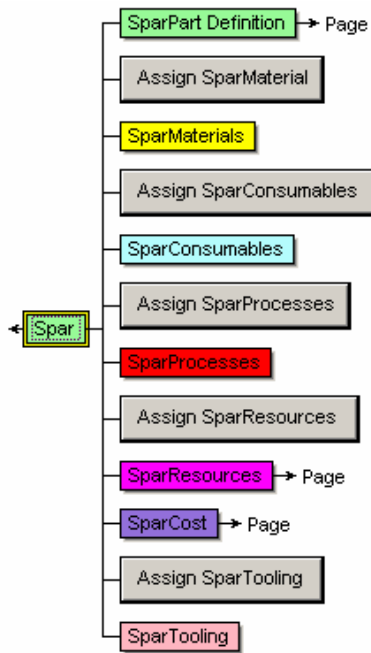


Fig. 10. Interface automatically generated from Protégé.

The user will make choices so the decision support result 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 11 illustrates this.

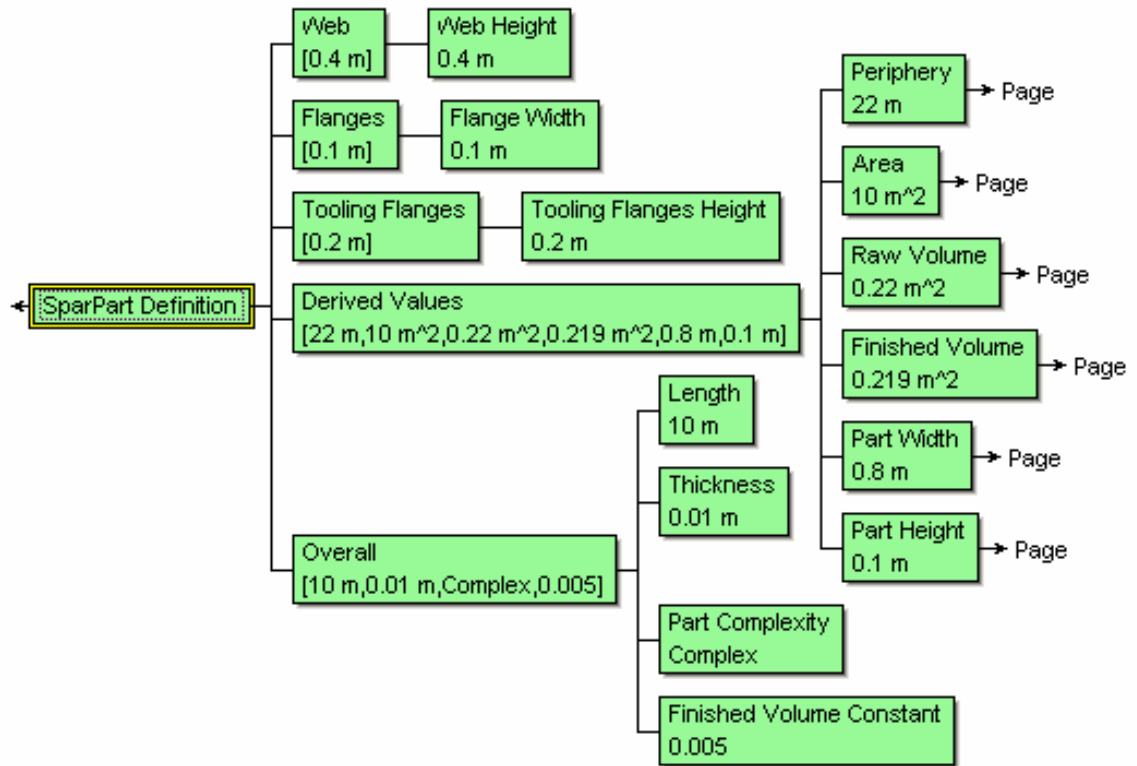


Fig. 11. Interface automatically generated from Protégé - spar definition.

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 [64].

Name	Raw Volume
Value	Raw Volume = Periphery * Thickness
Value	0.22 m^2
Parent	• Derived Values

Fig. 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 automatically translated from the part definition described in the part taxonomy.

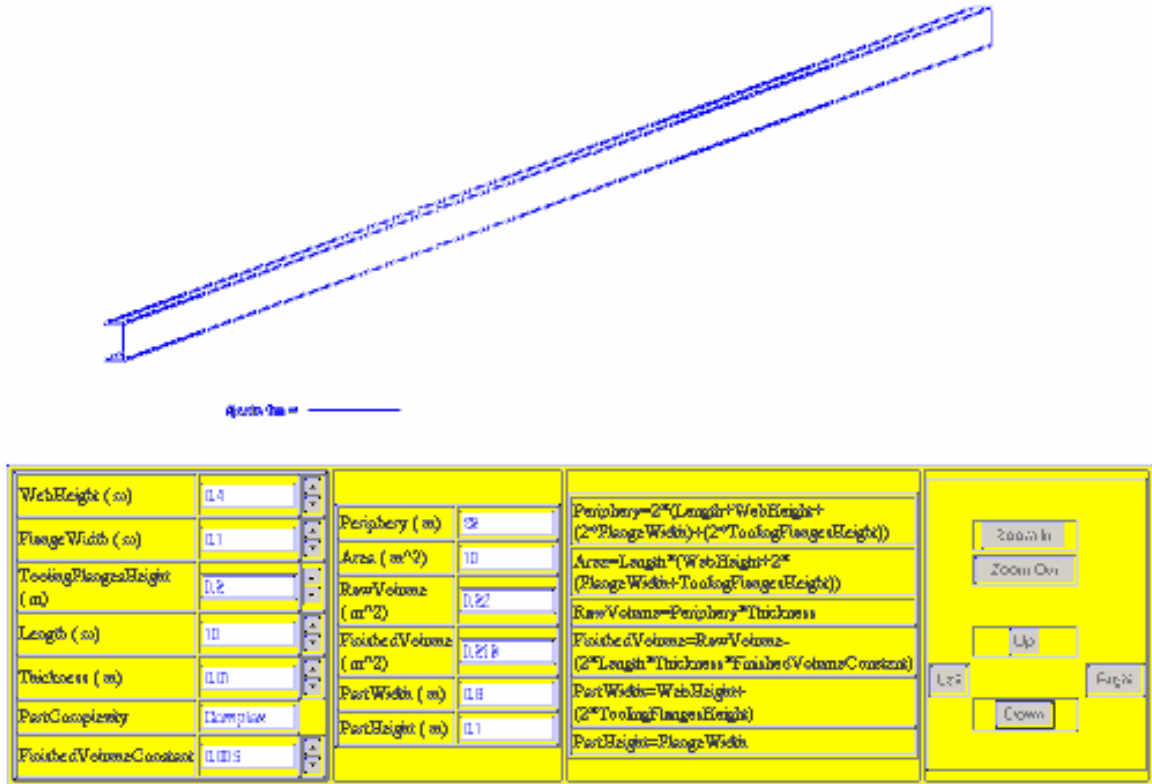


Fig. 13. Interactive Spar Diagram (SVG).

The way the user can interact with the diagram can be seen by viewing these examples on our Interactive SVG links page demonstrated in [59].

The XML can also be displayed on the web using a Flash program created by Rhodes et al. [60]. 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.

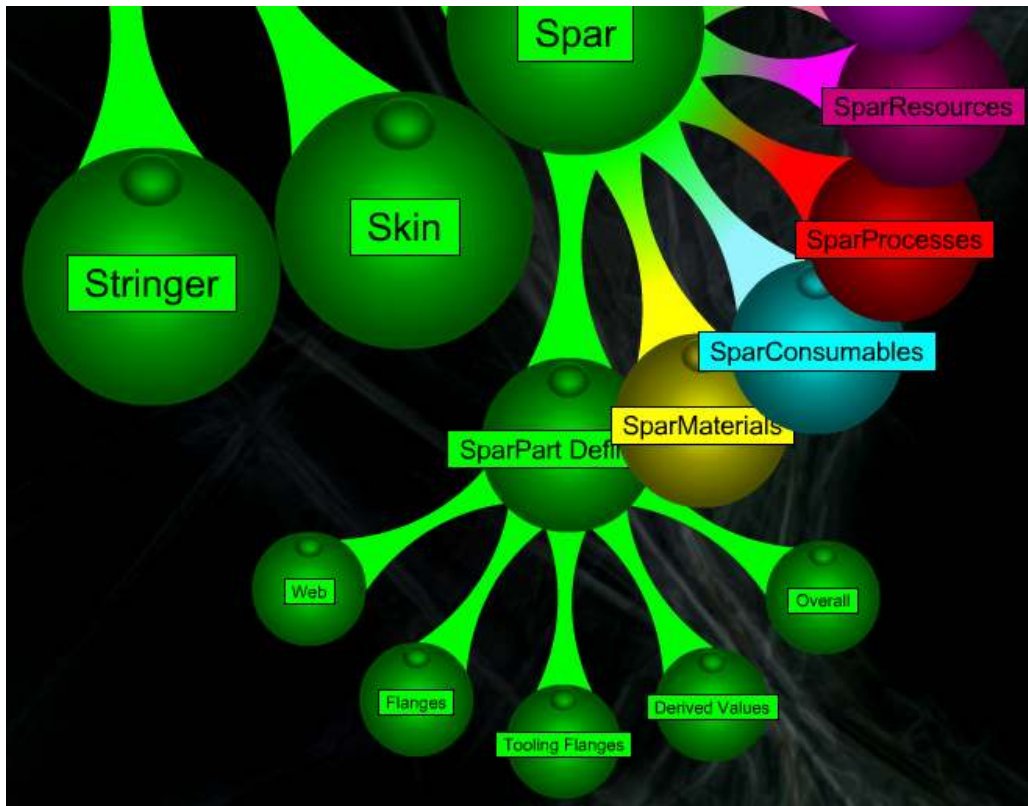


Fig. 14. Flash interface for navigating exported XML tree.

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 that 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. An alternative is required so modellers can continue their work without having to rely on software developers. This approach has the advantage that modellers can interact and change a generic model visually, as the full structure of the model is displayed in an interactive tree. Our alternative approach involves development of a system, where a model builder, via visual editing of library taxonomies can undertake creation, 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 edit, 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 and transformable image views of the decision support models over the web.

Future Research

Since these examples were created open standard tools have been released that should enable improvement of 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 [65], TopBraid Composer [66], General Electric's ACUIY

enterprise modelling tool [67], and Visual Knowledge's Semantic Wiki visualiser [68]. These tools include Java Eclipse extensions for high level programming. We have also investigated transformations that can translate the ontology into representations in other languages and tools. The result documents could be searched using XQuery within Exist [69] and SPARQL (Simple Protocol and RDF Query Language) [70] and edited using XForm editors such as Orbean XForms [71].

The next stage of research 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 [72] and aircraft Design and production

References

- 1 W3C, <http://www.w3.org/>, W3C World Wide Web Consortium, (2006).
- 2 OASIS, <http://www.oasis-open.org/home/index.php>.
- 3 NIST, <http://www.nist.gov/>, National Institute of Standards and Technology.
- 4 Parsia B, <http://www.xml.com/pub/a/2001/02/14/functional.html>, Functional Programming and XML, O'Reilly XML.com.
- 5 T. Redmond, T. Tudorache, J. Vendetti, Building Applications with Protégé: An Overview, 9th Intl. Protégé Conference, July 23-26, 2006 - Stanford, California.
- 6 Vanguard, <http://wiki.vanguardsw.com>, Global Knowledge Portal, (2006).
- 7 J. Scanlan, A. Rao, C. Bru, P. Hale, R. Marsh, DATUM Project: Cost Estimating Environment for Support of Aerospace Design Decision Making, Journal of Aircraft - (2006) 43(4).
- 8 Engineous Software, http://www.engineous.com/product_FIPER_specifications.htm, Specification Version 1.6, (2006).
- 9 K. B. Bruce, A. P. Danyluk, T. P. Murtagh, Event-driven programming is simple enough for CS1, Proceedings of the 6th annual conference on Innovation and technology in computer science education, 2001.
- 10 D. Frankel, P. Hayes, E. Kendall, D. McGuinness, The Model Driven Semantic Web - 1st International Workshop on the Model-Driven Semantic Web (MDSW2004) Enabling Knowledge Representation and MDA® Technologies to Work Together (2004).
- 11 T. Berners-Lee, M. Fischetti, M. L. Dertouzos, Weaving the Web, Harper San Francisco, 1999.
- 12 E. Olsson, What active users and designers contribute in the design process, Interacting with Computers 16 (2004) 377-401.
- 13 M. Erwig, R. Abraham, I. Cooperstein, S. Kollmansberger, Automatic Generation and Maintenance of Correct Spreadsheets?, Proceedings of the 27th international conference on Software engineering, St. Louis, MO, USA (2006) 136-145
- 14 EUSES (End Users Shaping Effective Software), <http://eusesconsortium.org/>, (2006).
- 15 Network of Excellence on End User Development in Europe, <http://giove.cnuce.cnr.it/eud-net.htm>, (2006).
- 16 Institute for End User Computing, <http://www.ieuc.org/home.html>, (2006).
- 17 F. Paternò, Model-based tools for pervasive usability, Interacting with Computers 17(3) (2005) 291-315.
- 18 C. Scaffidi, M. Shaw, B. Myers, Estimating the Numbers of End Users and End User Programmers, IEEE Symposium on Visual Languages and Human-Centric Computing, (VL/HCC'05) (2005) 207-214 Dallas, Texas.
- 19 Lubell J, 2006 <http://ats.nist.gov/psl/xml/process-descriptions.html> - XML Representation of Process Descriptions
- 20 S. Bechhofer, J. Carrol, Parsing owl dl: trees or triples?, Proceedings of the 13th international conference on World Wide Web, NY, USA (2004) 266 - 275.

- 21 M. Uschold, M. Gruninger, Ontologies and Semantics for Seamless Connectivity, Association for Computer Machinery - Special Interest Group on Management of Data - SIGMOD Record December 33(4)(2004).
- 22 Jena, <http://jena.sourceforge.net/>, Jena - A Semantic Web Framework for Java (2006).
- 23 Protégé, <http://protege.stanford.edu/>, welcome to protégé (2006).
- 24 M. Huhns, Interaction-Oriented Software Development, International Journal of Software Engineering and Knowledge Engineering 11 (2001) 259-279.
- 25 H. El-Ghalayini, M. Odeh, R. McClatchey, Engineering Conceptual Data Models from Domain Ontologies: A Critical Evaluation. IASTED International Conference on Databases and Applications, 23rd Multi-Conference on Applied Informatics, Innsbruck, Austria (2005) 222-227.
- 26 P. Hale, <http://www.cems.uwe.ac.uk/amrc/seeds/models.htm>, Models Page, (2006).
- 27 H. Aziz, J. Gao, P. Maropoulos, W. M. Cheung, Open standard, open source and peer-to-peer tools and methods for collaborative product development, Computers in Industry 56 (2005) 260-271.
- 28 C-B. Wang, T-Y. Chen, Y-M. Chen, H-C. Chu, Design of a Meta Model for integrating enterprise systems, Computers in Industry 56 (2005) 205-322.
- 29 S. Dmitriev, Language Oriented Programming: The Next Programming Paradigm, <http://www.onboard.jetbrains.com/is1/articles/04/10/lop/>, (2006).
- 30 K. Mens, I. Michiels, R. Wuyts, Supporting Software Development through Declaratively Codified Programming Patterns. Expert Systems with Applications 23 (2002) 405-413.
- 31 J. Gray, J. Zhang, Y. Li, S. Roychoudhury, H. Wu, R. Sudarsan, A. Gokhale, S. Neema, F. Shi, T. Bapty, Model-Driven Program Transformation of a Large Avionics Framework, 2004, Third International Conference on Generative Programming and Component Engineering GPCE.
- 32 R. Marsh, T. Hill, J. Scanlan, M. Dunkley, P. Cleevely, Probabilistic Pseudo-generative Cost Modelling Through Virtual Template Propagation CEAS Conference on Multidisciplinary Aircraft Design and Optimization June (2001) Maternushaus Koln, Germany.
- 33 G. C. Murphy, R. J. Walker, E. L. A. Baniassa, M. P. Robillard, A. Lai, M. A. Kersten, Does aspect-oriented programming work?, 2001, Communications of the ACM, 44 (10) 75 - 77 (2001) ISSN:0001-0782
- 34 T. Elrad, R. E. Filman, A. Bader, Aspect-oriented programming: Introduction, Communications of the ACM 44 (10) (2001) 28-32.
- 35 A. Bernstein, E. Kaufmann, C. Kaiser, C. Kiefer, Ginseng: A Guided Input Natural Language Search Engine for Querying Ontologies, Jena User Conference, Bristol, UK (2006)
- 36 I. G. Tollis, Graph Drawing and Information Visualization, ACM Computing Surveys, 28A (4) (1996).
- 37 C. L. Simons, I. C., Parmee, <http://www.cems.uwe.ac.uk/~clsimons/Publications/CooperativeInteraction.pdf>, A manifesto for cooperative human / machine interaction (2006).
- 38 A. Cypher, 1993, Watch What I Do Programming by Demonstration, MIT Press, ISBN:0262032139.
- 39 S. W. Ambler, <http://www.agiledata.org/essays/impedanceMismatch.html>, The Object-Relational Impedance Mismatch, (2003)..

- 40 A. Hunter, <http://www.cs.ucl.ac.uk/staff/a.hunter/tradepress/eng.html>, Engineering Ontologies, (2006).
- 41 C. Fluit, S. Marta, F. V. Harmelen, Supporting User Tasks through Visualisation of Light-weight Ontologies. Handbook on Ontologies, Springer-Verlag, (2003).
- 42 O. Corcho, A. Gómez-Pérez, A Roadmap to Ontology Specification Languages. Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management. Chicago, USA (2000).
- 43 O. Corcho, M. Fernández-López A. Gómez-Pérez., Methodologies, Tools and Languages For Building Ontologies. Where is their Meeting Point?. Data and Knowledge Engineering, 46 (2003) 41-64.
- 44 N.F. Noy, Semantic Integration: A Survey Of Ontology-Based Approaches. SIGMOD Record, Special Issue on Semantic Integration, 33 (4) (2004).
- 45 M. Lemos, <http://www.meta-language.net>, MetaL: An XML based Meta-Programming language, (2006).
- 46 Simkin, <http://www.simkin.co.uk/>, A high-level lightweight embeddable scripting language which works with Java or C++ and XML, (2006).
- 47 S. Morris, I. Neilson, C. Charlton, J. Little, Interactivity and collaboration on the WWW - is the 'WWW shell' sufficient?. Interacting with Computers, 13, (2001) 717-730.
- 48 H. Aziz, J. Gao, P. Maropoulos, W. M. Chewing, Open standard, open source and peer-to-peer tools and methods for collaborative product development, Computers in Industry, 56 (2005) 260-271.
- 49 P. Ciancarini, D. Rossi, F. Vitali, Designing a document-centric coordination application over the Internet, Interacting with Computers 13 (2001) 677-693.
- 50 S. Nidamarthi, R. H. Allen, D. S. Ramm, Observations from supplementing the traditional design process via Internet-based collaboration tools, Computer Integrated Manufacturing 14 (1), (2001) 95-107.
- 51 G. Q. Huang, K. L. Mak, Issues in the development and implementation of web applications for product design and manufacture, Computer Integrated Manufacturing, 14 (1), (2001) 125-135.
- 52 J. A. Reed, G. J. Follen, A. A. Afjeh, Improving the Aircraft Design Process Using Web-Based Modeling and Simulation, ACM Transactions on Modeling and Computer Simulation, 10 (1), (2000) 58-83.
- 53 Y. Kim, Y. Choi, S. Bong Yoo, Brokering and 3D collaborative viewing of mechanical part models on the Web, Computer Integrated Manufacturing, 14 (1), (2001) 28-41.
- 54 S. Zhang, S. Weimen, G. Hamada A review of Internet-based product information sharing and visualization, Computers in Industry, 54, (2004) 1-15.
- 55 W. D. Li A Web-based service for distributed process planning optimization. Computers in Industry 56 (2005) 272-288.
- 56 A. W. Crapo, L. B. Waisel, W. A. Wallace, T. R. Willemain, Visualization and the process of modeling: a cognitive-theoretic view - Conference on Knowledge Discovery in Data - Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (2000) 218-226.
- 57 D. C. Smith, A Computer Program to Model and Stimulate Creative Thought. Basel: Birkhauser (1977).

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

59 P. Hale, <http://www.cems.uwe.ac.uk/~phale/InteractiveSVGExamples.htm>, Interactive SVG Examples, (2006)

60 G. Rhodes, J. Macdonald, K. Jokol, P. Prudence, P. Aylward, R. Shepherd, T. Yard, A Flash Family Tree, Flash MX Application and Interface Design, 1st October 2002 ISBN:1590591585

61 P. Hale, http://www.cems.uwe.ac.uk/~phale/RectangleDemo/RectangleDemo.viewlet/RectangleDemo_launcher.html, User Driven Modelling Example, (2006).

62 M. Schrage, Spreadsheets: Bulking Up On Data, Los Angeles Times, written (1991), <http://www.systems-thinking.org/buod/buod.htm>, (accessed 2006).

63 C. Merlo P. Girard, Information system modelling for engineering design co-ordination, Computers in Industry 55 (2004) 317-334.

64 E. De Andreis, <http://manudea.duemetri.net/xtree/>, 2MXtree XML-based tree, (2006).

65 Metatomix m3t4, <http://www.metatomix.com/news/060307.html>, Metatomix Provides Free Semantic Toolkit for Eclipse Developers Worldwide (2006).

66 TopBraid Composer, <http://www.topbraidcomposer.com/>, The Complete Semantic Modeling Toolset (2006).

67 A. Aragonés, J. Bruno, A. Crapo, M. Garbiras, An Ontology-Based Architecture for Adaptive Work-Centered User Interface Technology, Jena User Conference (2006), Bristol, UK.

68 Visual Knowledge, <http://www.visualknowledge.com>, Semantic Wiki.

69 Exist, <http://exist.sourceforge.net/xquery.html>, Open Source Native XML Database (2006).

70 SPARQL, <http://dret.net/glossary/sparql>, Simple Protocol and RDF Query Language (2006).

71 Orbeon Xforms, <http://www.orbeon.com/>, Form-based web applications, done the right way (2006).

72 Vanguard Software Modelling Wiki - <http://wiki.vanguardsw.com/bin/browse.dsb?dir/Engineering/Aerospace/> - Engineering - Aerospace.