Enabling diagrammatic de-abstraction and modelling of engineering problems

Abstract

This paper discusses efforts to enable de-abstraction of engineering problems from a representation suitable for engineers to that suitable for computer models and code. The key question is to what extent diagrammatic representations of problems can be used in order to provide modelling solutions. To achieve this, a source tree is created, this is translated to computer code, then represented as a result tree. This enables engineers to visualise problems such as representation of a product data structure in a way familiar to them, and this also gives a visual and colour coded representation of equations. This visualisation is easier to navigate and understand than that which can be provided by spreadsheets, and more maintainable. This research could also be used for business modelling, process modelling, and workflow.

Introduction

C.S. Peirce (1906) stated in 'Prolegomena to an Apology for Pragmaticism' "Come on, my Reader, and let us construct a diagram to illustrate the general course of thought; I mean a system of diagrammatization by means of which any course of thought can be represented with exactitude". That is the purpose of this research, but to limit the scope and so make application of this theory testable the research is restricted mainly to engineers (because they often think in terms of diagrams) and to the domain of modelling (which often requires diagrams). So the aim is to apply the research first where it can have the most use and encourage others to expand it for other domains and other users. This research is intended to simplify computing for computer literate non programmers, this includes many engineers. The main research area is enabling users such as engineers to model the problems they encounter in manufacturing and design. However, the wider aim is to prototype research for enabling a larger range of software users to model their problems. The intention is to create collaborative tools that allow users to develop software in a way they will be familiar with from their use of spreadsheets. This research brings together approaches of object orientation, the Semantic Web, relational databases, and Model Driven and Event Driven programming. Frankel et al. (2004) explain the opportunities for, and importance of this kind of research.

A way can be provided of applying this research and enabling more end user participation. This research is to test an approach of using digital technology, to make software and modelling development easier for computer literate end users. This would then enable them to solve problems to help their, and their teams' work to be more productive. The current problem is that the communication and translation steps required between users, and software developers, and help desk are too many, and too varied, this results in cost, uncertainty, delays, confusion, and confrontation. The solution is for software developers to develop more customisable software that can be customised by end users.

Figure 1 shows the way iterative development is used both in this research and in the implementation to ensure that changes can be made systematically as necessary and without disrupting the project.

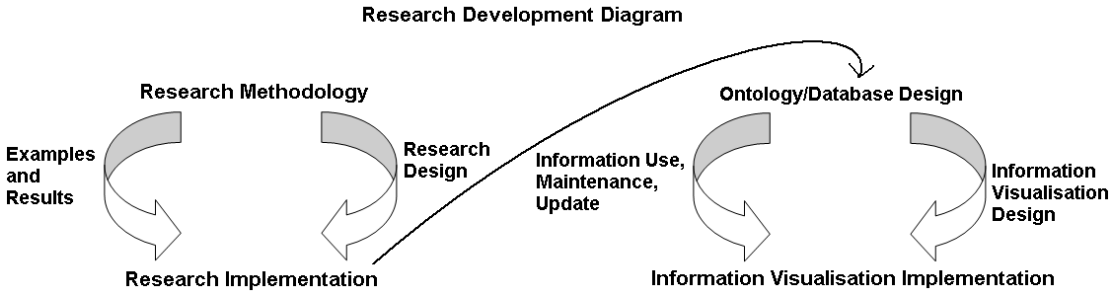

Figure 1. Research Development Iterations

Software engineering and modelling has much in common with engineering modelling, also the tools used for both, have much in common. Software process modelling, engineering process modelling, and business/workflow modelling share a common approach, and similar tools. Much of this commonality is in the need to transform requirements into design into code semi automatically. To achieve this, continuous consultation between potential users e.g. engineers for engineering modelling problems and developers for software problems is required.

An example of the need to apply such research is user's development of spreadsheets, this indicates user's willingness to use and develop software to meet their needs, but productivity in spreadsheet development can be poor, as they are difficult to track, and share. More advanced software tools are often not available to users, or take too long to learn. Semantic Web technologies can provide a solution to this by provision of free and customisable, shareable, and fully visualised applications for use in particular sectors. Development of such applications, and their use in industry would be proof of this.

Methodology

A common factor in these various types of modelling is the need to transform between a high level abstraction, to a lower level such as a computer model and then code. This is illustrated by examples of semi automatically produced programs/models (Hale, 2008). The translation process involves translating from a tree/graph representation, and for each node this is translated into a code representation of the equation that relates this node to any others, and this code is then presented in the interface as a result tree/graph. This can be achieved for programs and/or web pages. Kraus et al. (2007) examine and implement this transformation problem and also produce code and/or web pages. Uschold (2003) defines the Semantic Web as being machine usable and associated with more meaning. So this is a good way to convey the abstractions represented in a source and result tree to the end user.

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

Solutions to this transformation problem can be found by adapting current tools and techniques using a systematic approach. Such tools and techniques involve use of modelling tools, spreadsheets, ontology management tools, and Semantic Web/Web 2.0 tools. These possible solutions are not mutually exclusive and their combination could be the best way of providing usable collaborative modelling tools for computer literate end users and domain experts. The link between these alternative ways of advancing current research is translation and User Driven Modelling/Programming.

Software tools to improve and combine UML (Unified Modeling Language) and other modelling, spreadsheet, Web 2.0/Semantic Web, and software solutions are required to allow this kind of end user participatory development and a sustained collaborative organised push towards achieving this. To achieve this, end users and teams need to be able to visualise their solutions both literally as diagrams, and metaphorically via problem solving collaboration. This would realise the intentions that were expressed decades ago in the form of PC based computing and spreadsheets, but add better information and modelling organisation, and collaboration capabilities.

The essence of the problem is that organisations and individuals have many limitations resulting from the lack of facilities to allow users to program. For example the use of 'out of the box' modelling tools, which are hard to customise or introduce collaborative capabilities, because a project deadline is so urgent that nothing else is practical. The problem with this is that when organisations have urgent needs to produce models for a particular deadline, all their effort goes into this. Once a deadline is met or missed, there will be a new deadline for something else. So all the effort goes into meeting these deadlines, and there are little or no resources for providing a better solution that makes it easier in the long run to meet the deadlines. This causes many issues, which need to be urgently tackled by organisations in the short term. This creates an ironic dilemma, that the problems caused by lack of end user programming facilities cause there to be a need to tackle these problems via quick fixes because they are so urgent. This makes it difficult to secure funding and time for longer term end user programming research.

As an example of the problems that occur; many people use Excel for financial modelling, and engineering, business, and science models. As these models become large, the short term solution is to use Excel Visual Basic to aid navigation through the model. However, though this helps with the immediate problem, in the longer term it can be difficult to reuse the model, as the Excel spreadsheet creator has to go through his or her code and remember what they have done, this problem is worse if a different user is trying to reuse the spreadsheet. Experience of dealing with projects that create new software systems is that they are very ambitious, but not very modular or customisable. Under management pressure for meeting of timescales, they are often released too early with insufficient consultation with users. Then software developer's time is reallocated to a help desk, in order to deal with the problems caused by the release of software too early, too unchangeable and with too little user involvement.

A longer term solution would be to provide a visual programming environment where the structure of the code is visualised, to make it easier to track the formulae, and so navigation is a central part of the program design, this also minimises the need for code. A solution is to produce highly customisable software, so that the software team do not have to anticipate every problem the users will want to solve. Then many users with more advanced needs and computing abilities could customise software for their needs.

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

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

Figure 2 shows the solutions, and how these make User Driven Modelling/Programming possible :-

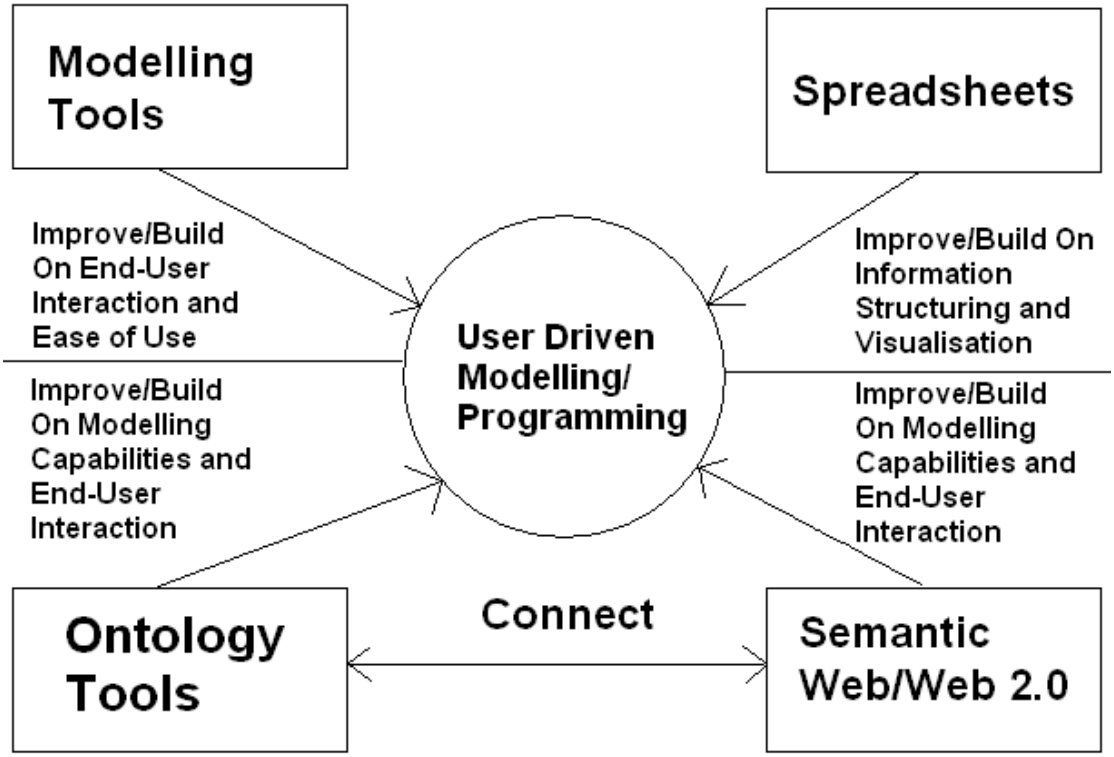

Figure 2. Methodology Diagram - Enabling User Driven Modelling/Programming

It is possible to create an extra layer of visualised semantics to enable users to specify commands in structured language. This approach of adding extra layers is the way this visual programming works. Users provide the information the program needs at the visual interface layer, and program code is created automatically. The layers provide the bridge between

abstract ideas and computer code. If this approach is taken to its logical conclusion, it would be possible to allow the user to specify what the computer should do. Then each layer would communicate this to the layer below until the computer performs the action required. A simple example of this approach is the use of spreadsheets. Users can specify a calculation in mathematical terms using a formula. The spreadsheet then calculates the result of the formula. Users can change the formula if it is incorrect without any need to write code or re-compile. This accounts for the popularity of spreadsheets. However, spreadsheets do not provide the centralised and structured data-store required for a distributed collaborative system. Therefore, this research concentrates on combining the wide applicability of generic spreadsheet modelling with structured and adaptable modelling and visualisation.

It is important to enable changes to the design of the information source and its structure as necessary, even when it contains information. This makes possible continuous improvement of the information and its representation together. Clear visualisation of the structure makes out of date and duplicate information obvious, so it can be changed by the end users of the information. This provides for maintenance of information quality without necessitating end users to understand relational database design; though relational databases can still be accessed by software specialists for more in depth and less frequent structural changes. Engineering modelling can be performed using a high level diagrammatic view of the problem and conveyed to the computer via transformation. Program/model transformation allows for writing in one representation or language, and translating to another. This is particularly useful for language independent programming, or for high level and end user programming that can then be translated to a language more easily interpreted by computer systems.

A taxonomy representation is translated into a computer model. Relationships can be conveyed to a software model that evaluates them. Information is translated from the taxonomy and is visualised in tree form in a decision support tool with the example of wingbox manufacture information. The visualisation of the information in a tree can be further translated into visualisation as an interactive diagram. The representation can be translated into different languages, to allow for language independence.

Figure 3 explains the transformation process.

Figure 3. Translation Process

Implementation

Figure 4 illustrates the implementation of the translation stages. Step 1 is creation of the ontology, which is then translated to the decision support and modelling tool (Vanguard System) for Step 2. Step 2 is illustrated to the right, and this shows colour coding of the taxonomies (sub ontologies) that make up the ontology e.g. parts, processes, and materials. Step 3 involves translations to visualisations for the web (using Semantic Web formats) and alternative representations. Step 3 can also produce program and/or meta-program code.

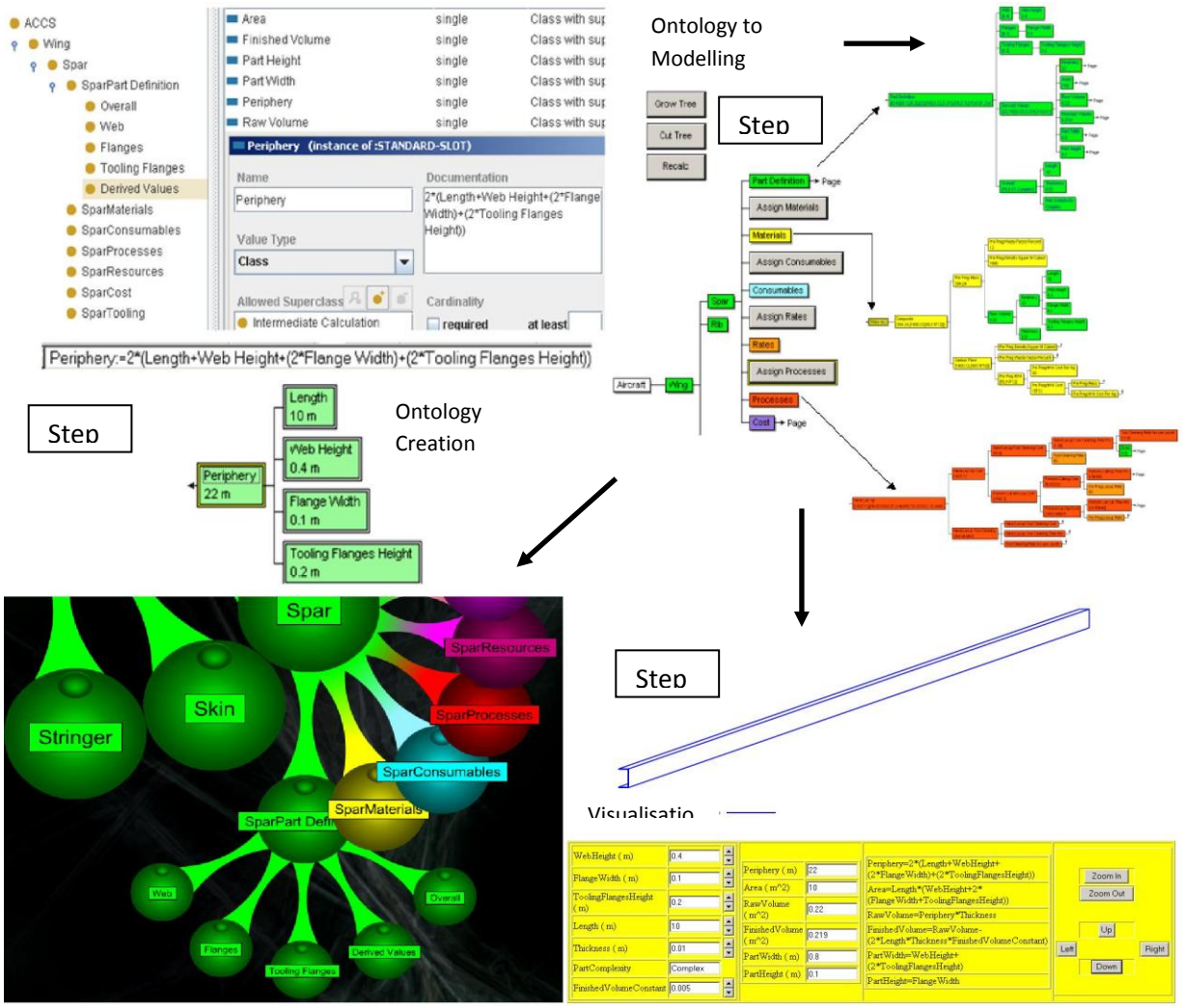

Figure 4. Stepped Translation and Visualisation

Related Research

To make the above practical, sustained research is needed in the areas of visualisation, modelling, end user programming, and transformation as well as the links between these areas. Crapo et al. (2002) assert the need for a methodology for creation of systems to enable more collaborative approaches to modelling by domain expert end users, and that this combined with visualisation would allow engineers to model problems accurately. Huhns (2001) and Paternò, (2005) both explain that alternatives to the current approach to software development are required. Modelling languages such as Alloy explained by Wallace (2003) can be used as an interface to an end user programming environment. Transformation from a model building environment to program code has been investigated by Gray et al. (2004).

Conclusion

Even if programming is made easier, only a proportion of people would actually be interested or capable of doing this. But there is still an advantage to colleagues such as people in the same team or department as an end user programmer, even if they are not undertaking programming themselves. Then all in the team have much closer access to someone, the end user programmer, who understands their, and the team's tasks, requirements, and projects. This closes the gap between those producing software systems, and those who require the software. This also makes it easier to iterate through solutions and solve problems more quickly and collaboratively.

Experienced programmers can build a modelling environment that can then be used by non programmers to create models or solve other software problems. This was achieved for the DATUM (Design Analysis Tool for Unit-cost Modelling) project with Rolls-Royce, and the modelling environment created was used by their engineers. This is described by Scanlan et al. (2006). Collaboration, simulation and modelling have been investigated to determine the requirements for future research in modelling of problems. This should allow translation from a model based representation of software to the actual software. This can involve semi automatically producing software for a Semantic website from visual representations of the problem. The core of this modelling infrastructure is automated generation of models created with World Wide Web Consortium (W3C) standards based languages, and the visualisation of information represented in such W3C standard ways. This research investigated alternative approaches to software development, which give users greater involvement, and can actually be used in combination. This partially automates the process of software creation via a collaborative process and equation tree that maps the problem structure, and user interface creation by providing a means to manage a diagrammatic and/or tree based representation.

References

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

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

Gray, J., Zhang, J., Lin, Y., Roychoudhury, S., Wu, H., Sudarsan, R., Gokhale, A., Neema, S., Shi, F., Bapty, T., 2004. Model-Driven Program Transformation of a Large Avionics

Framework. *In*: *Third International Conference on Generative Programming and Component Engineering GPCE*, pp 361-378.

Hale, P. 2008. http://www.cems.uwe.ac.uk/~phale/EconomicModels/ModelsVisualised.htm

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

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

Kraus, A., Knapp A., Koch, N., 2007. Model-Driven Generation of Web Applications in UWE. http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-261/paper03.pdf *In Proc. MDWE 2007 - 3rd International Workshop on Model-Driven Web Engineering, CEUR-WS/,* Vol 261, July 2007.

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

Peirce, C.S. - 1906. Prolegomena to an Apology for Pragmaticism - http://www.existentialgraphs.com/peirceoneg/prolegomena.htm.

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

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

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