

University of the West of England, Bristol
Enabling diagrammatic de-abstraction and modelling of engineering problems
Peter Hale – PhD supervision team, Tony Solomonides and Ian Beeson

Abstract

Problem -

Enable de-abstraction of engineering problems from engineers' representation to computer models and code.

To what extent can diagrammatic representations of problems can be used in order to provide modelling solutions.

Solutions -

A source tree is created, then translated to computer code, then represented as a result tree.

Introduction

Purpose -

To test this problem -

- C.S. Peirce (1906) -
- 'Prolegomena to an Apology for Pragmatism'
- "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"

To limit the Scope –

- Research restricted mainly to engineers (who often use diagrams)
- To domain of modelling (which often requires diagrams)

Introduction Continued

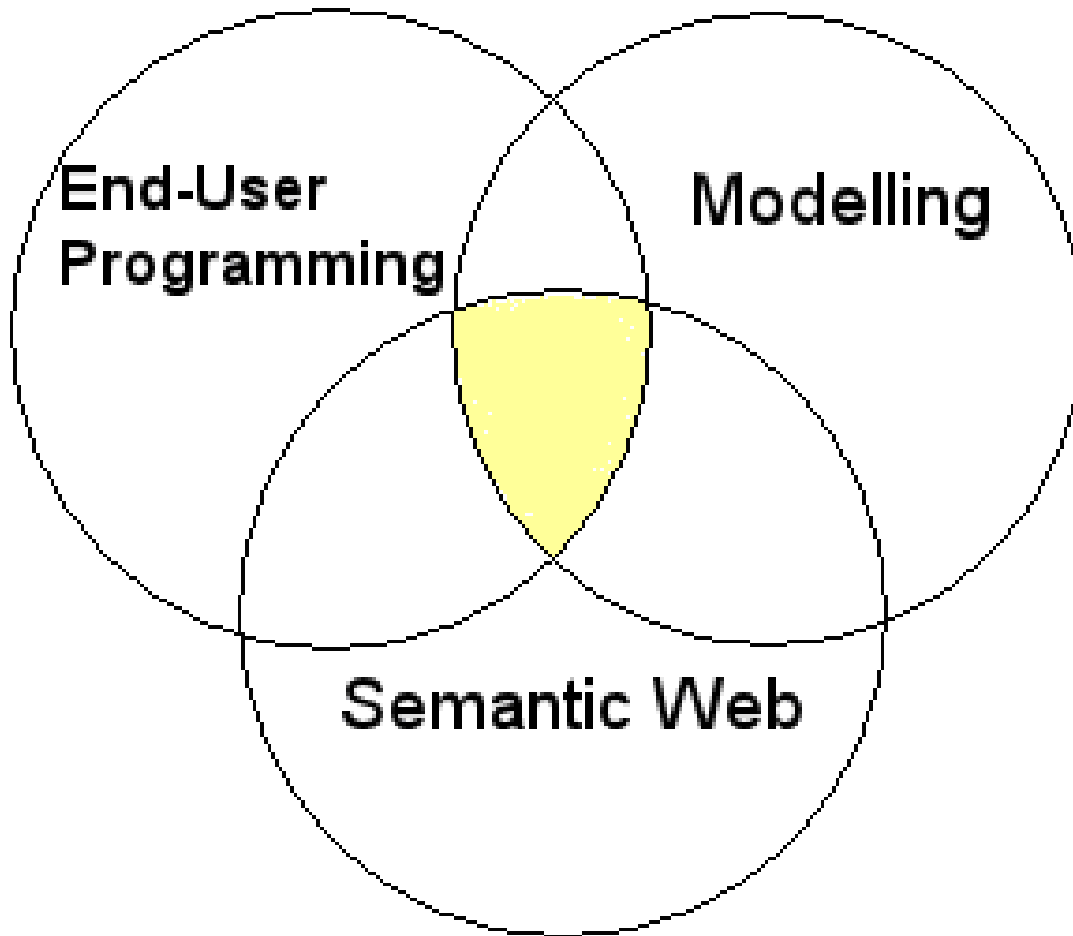
Benefits -

- Enables engineers to visualise problems such as representation of a product data structure in a familiar way
- Gives a visual and colour coded representation of equations
- Visualisation is easier to navigate and understand than that in spreadsheets, and more maintainable

Wider Implications -

- This research could also be used for business modelling, process modelling, and workflow

Research Area



Aim

- Apply the research first where it can have the most use
- Encourage others to expand it for other domains and other users
- Simplify modelling/computing for computer literate non programmers, this includes many engineers
- Enabling users such as engineers to model the manufacturing and design
- Wider aim - to prototype research for enabling a larger range of software users to model their problem
- Enabling more end user participation

Methodology

- Create collaborative tools - enable users to develop software in a way they will be familiar with from their use of spreadsheets
- 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
- Translation steps from user to computer and Vice Versa
- Iterative Design of an Iterative System

Methodology - Iterative Design

Research Development Diagram



- When things don't work out right can go back a stage instead of having to start again

User Driven Modelling/Programming

Advantages –

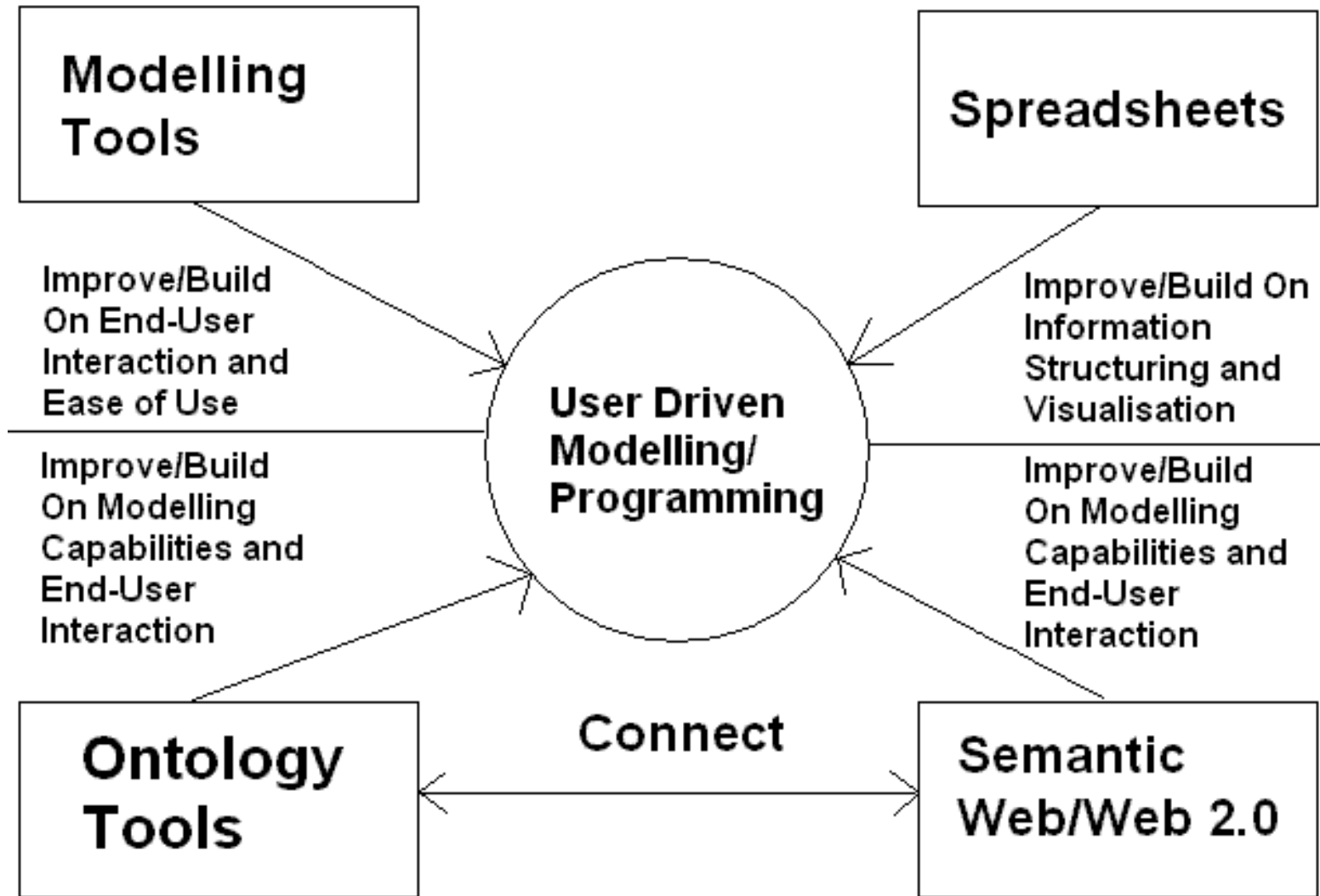
- Uses a modelling approach for creating modelling solutions
- Involves creating systems to create systems
- Makes it possible to solve problem by breaking it down into stages - allowing software developers to concentrate on the most complex software problems and domain experts to be able to concentrate on their domain problem
- Standardisation to allow software developers to create modelling systems for generic purposes
- These can be customised and developed by domain experts to model their domain

Tools and Technologies

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

Tools and Technologies 2



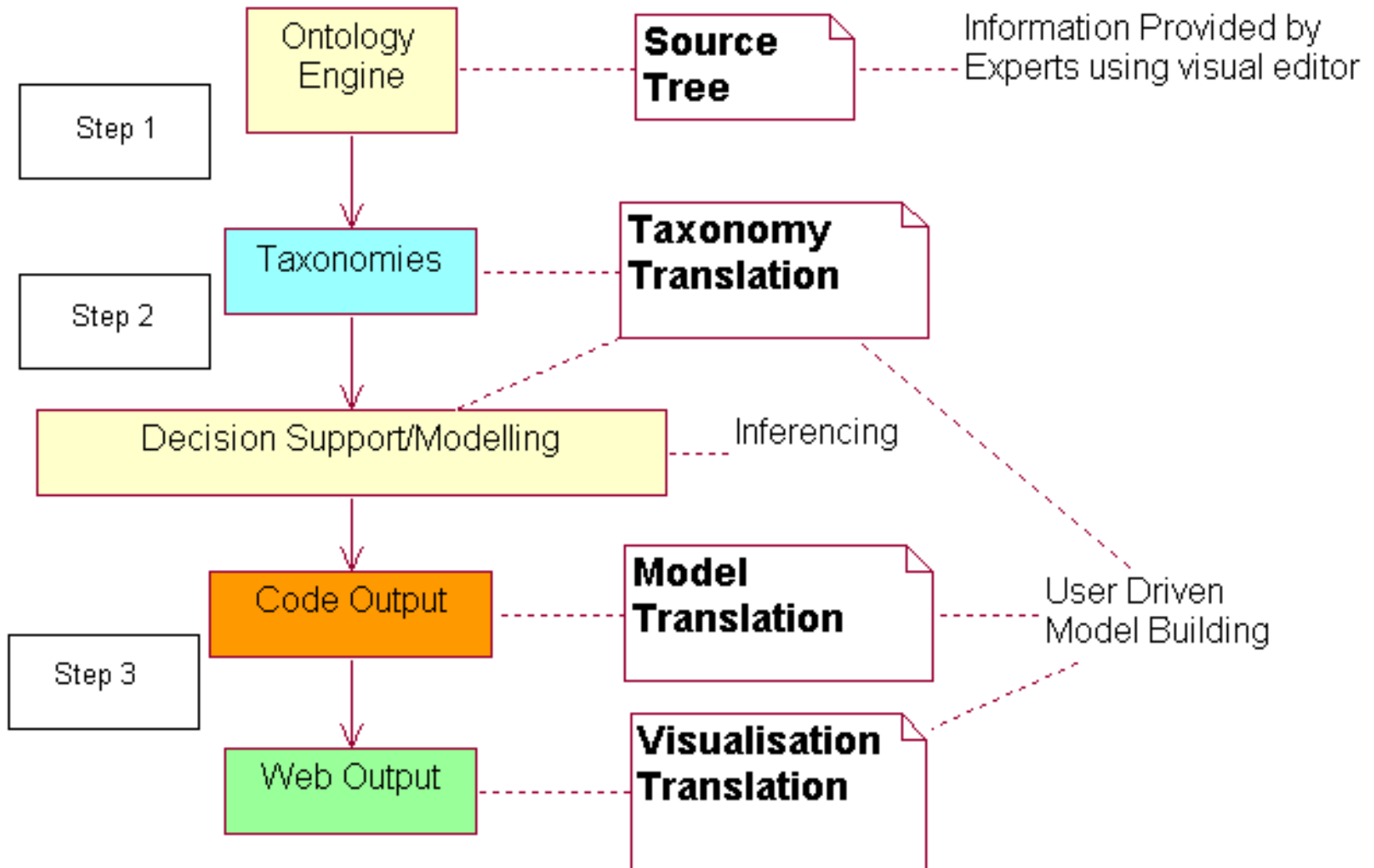
Implementation

- Demonstrates construction of diagrammatic representations of cost using the example of an aircraft wing parts
- Diagrammatic representations achieved by visual representation of items and equations to represent cost
- These items and equations are represented in standardised categories used in engineering - 'materials', 'processes', 'cost rates' etc
- Other items and equations could be represented in the same way, so other engineering products could be represented
- 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

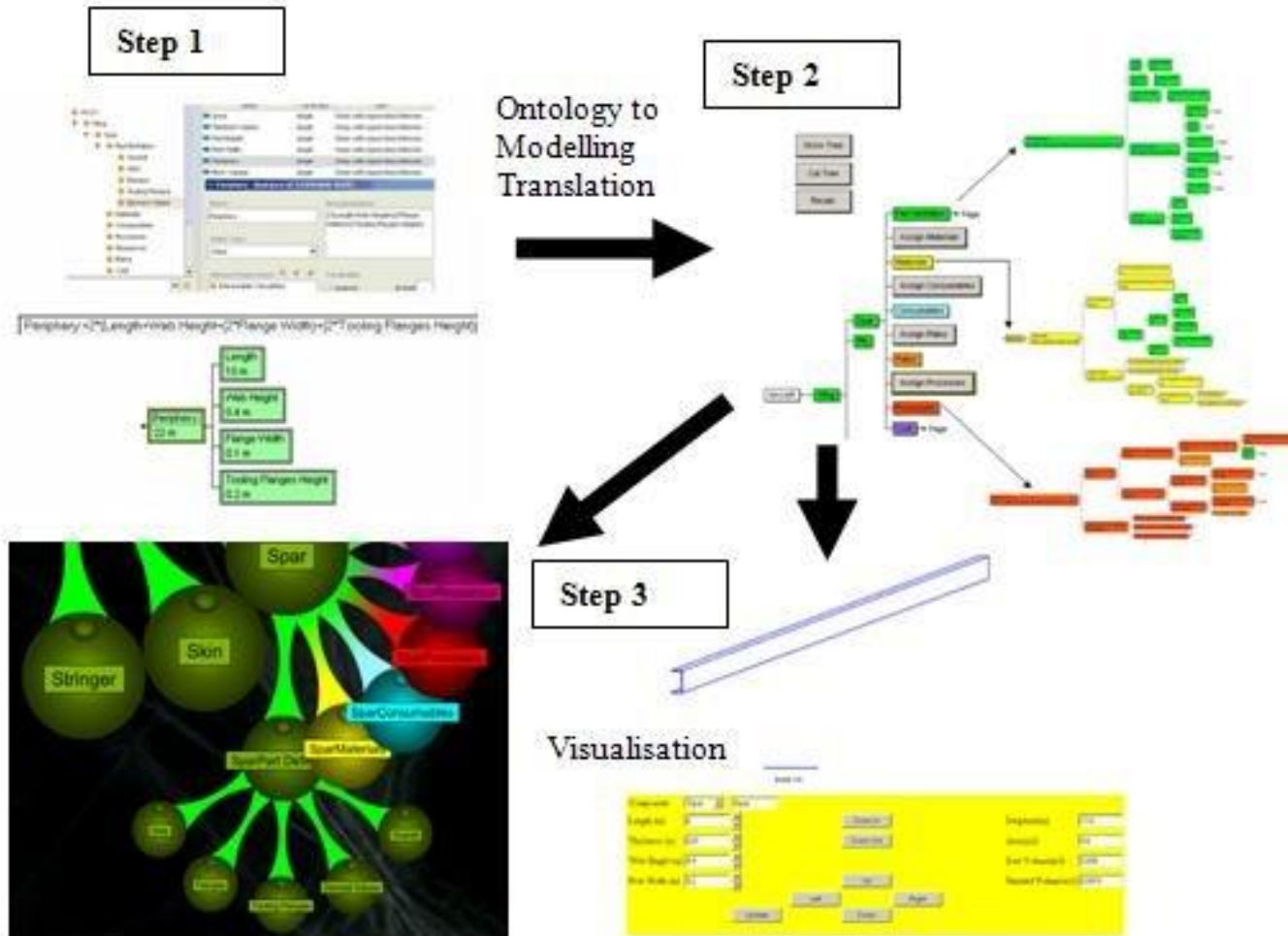
Translation

- 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
- 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 software languages, to allow for language independence

Translation Process



Translation Stages



Example Illustration

User Driven Model Development Simple Illustration

The screenshot displays the Protégé ontology editor interface. At the top is a menu bar with 'File', 'Edit', 'Project', 'Window', and 'Help'. Below the menu bar is a toolbar with various icons for file operations and editing. The main interface is divided into several panes:

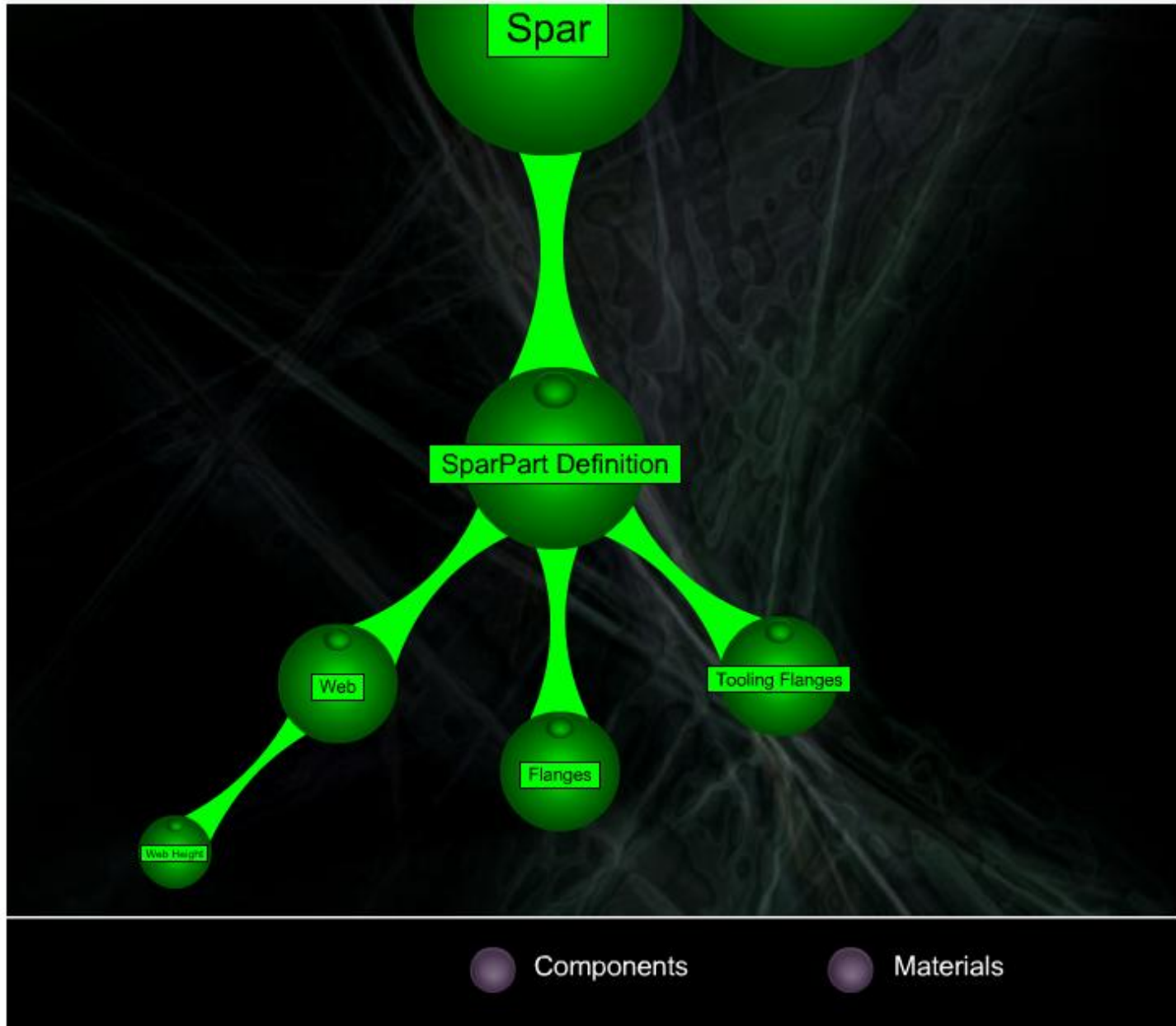
- CLASS BROWSER:** Shows the class hierarchy for the project 'Rectangle'. The hierarchy is: Rectangle (selected) > Shape > Intermediate > Rectangle. Under 'Rectangle', there are sub-classes 'Derived Values' and 'InputValues'. Below the hierarchy is a 'Superclasses' section showing 'Shape' as a superclass of 'Rectangle'.
- CLASS EDITOR:** Shows the details for the selected class 'Rectangle'. It includes a 'Name' field containing 'Rectangle' and a 'Documentation' field. Below these fields is a table with columns 'Cardinality' and 'Type'.

A callout box with a pointer to the 'Rectangle' class in the hierarchy contains the following text:

This is 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.

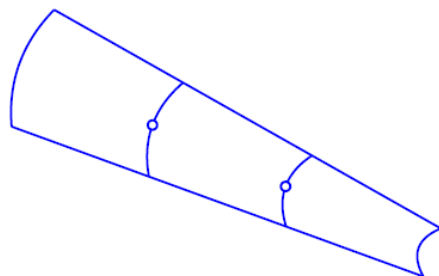
The ontology editor can be used to produce anything from a simple taxonomy like this to a large and complex ontology.


Web Tree Representation



Web Diagrammatic Representation

Component Illustration



Scale 1m = 

BottomSkinDepthofCurvature (m)	0.3	BottomSkinArea (m ²)	16.821445	$\text{BottomSkinArea} = \text{BottomSkinLength} * (\text{BottomSkinRootChord} + \text{BottomSkinTipChord}) / 2$ $\text{BottomSkinFinishedVolume} = \text{empty}$ $\text{BottomSkinPartHeight} = \text{BottomSkinDepthofCurvature}$ $\text{BottomSkinPartThickness} = 0.539 * (\text{BottomSkinRootThickness} + \text{BottomSkinTipThickness}) / 2$ $\text{BottomSkinPartWidth} = (\text{BottomSkinRootChord} + \text{BottomSkinTipChord}) / 2$ $\text{BottomSkinPeriphery} = \text{BottomSkinRootChord} + \text{BottomSkinTipChord} + (2 * \sqrt{((\text{BottomSkinRootChord} - \text{BottomSkinTipChord}) * (\text{BottomSkinRootChord} - \text{BottomSkinTipChord}) / 4) + (\text{BottomSkinLength} * \text{BottomSkinLength})})$ $\text{BottomSkinRawVolume} = (\text{BottomSkinLength} * \text{BottomSkinPartThickness} * ((\text{BottomSkinRootChord} + \text{BottomSkinTipChord}) / 2) + (\text{BottomSkinPartThickness} * \text{BottomSkinPeriphery} * \text{BottomSkinRawVolumeConstant}))$	<input type="button" value="Zoom In"/> <input type="button" value="Zoom Out"/> <input type="button" value="Up"/> <input type="button" value="Left"/> <input type="button" value="Right"/> <input type="button" value="Down"/>
BottomSkinLength (m)	9.167	BottomSkinFinishedVolume	empty		
BottomSkinPartComplexity	Extremely Sir	BottomSkinPartHeight (m)	0.3		
BottomSkinRawVolumeConstant (m)	0.005	BottomSkinPartThickness (m)	0.0012127500		
BottomSkinRootChord (m)	2.6	BottomSkinPartWidth (m)	1.835		
BottomSkinTipChord (m)	1.07	BottomSkinPeriphery (m)	22.067729642		
BottomSkinRootThickness (m)	0.0035	BottomSkinRawVolume (m ³)	0.0205340206		
BottomSkinTipThickness (m)	0.001				
BottomSkinManholeDiameter (m)	0.2				
BottomSkinNumberofManholes	2				

- Interaction with a different view of the Design Problem

Summary and 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
- 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
- Achieved for the DATUM (Design Analysis Tool for Unit-cost Modelling) project with Rolls-Royce, Scanlan et al. (2006)
- Enables collaboration, simulation and modelling by translation from a model based representation of software to the actual software
- Gives users greater involvement
- Partially automates the process of software creation via a collaborative structure that maps the problem, and user interface creation by diagrammatic and/or tree based representation

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) - Creation of systems to enable more collaborative modelling by domain expert end users, with visualisation, would allow engineers to model problems
- Huhns (2001) and Paternò, (2005) explain that alternatives to the current approach to software development are required
- Kraus et al. UWE - UML-based Web Engineering
- 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)

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