Visualising Early Engineering Design Information with Diagrams

Filippo A. Salustri
Ryerson University

Janaka S. Weerasinghe
Ryerson University

Rob H. Bracewell
University of Cambridge

Nathan L. Eng
University of Cambridge

Please Cite:
Salustri, F. A., Weerasinghe, J. S., Bracewell, R. H., & Eng, N. L. (2007). Visualising early engineering design information with diagrams. *Journal of Design Research, 6*(1–2), 190–217.
doi:10.1504/JDR.2007.015569

VISUALISING EARLY ENGINEERING DESIGN INFORMATION WITH DIAGRAMS

Filippo A. Salustri

Department of Mechanical and Industrial Engineering, Ryerson University
350 Victoria St., Toronto, ON, M5B 2K3, Canada
E-mail: salustri@ryerson.ca

Rob H. Bracewell

Engineering Design Centre, Department of Engineering, University of Cambridge
Trumpington Street, Cambridge, CB2 1PZ, UK
E-mail: rhb24@cam.ac.uk

Nathan L. Eng

Engineering Design Centre, Department of Engineering, University of Cambridge
Trumpington Street, Cambridge, CB2 1PZ, UK
E-mail: nle20@eng.cam.ac.uk

Janaka S. Weerasinghe

Department of Mechanical and Industrial Engineering, Ryerson University
350 Victoria St., Toronto, ON, M5B 2K3, Canada
E-mail: jweerasi@ryerson.ca

Abstract: The authors report on their development of computer-based diagramming tools to support the early stages of engineering design and to improve the capacity to innovate. We believe that the human brain is currently the best available device to perform early designing. Thus, good tools will facilitate designers' abilities to "see" information patterns, reflect on them, and achieve insights they may not have achieved otherwise. Our work suggests that diagram layout and style are at least as important as the textual content for their rapid comprehension. These features can be easier to manage in software than in textual descriptions, leading to tools that are more robust and usable. We are working along several lines of inquiry intended to explore particular aspects of the matter, including using existent tools such as concept maps, and developing alternative tools to test ideas about supporting early design by diagramming. While we have not formally evaluated much of our work, anecdotal evidence is encouraging.

Keywords: information visualisation, diagrams, diagrammatic representation and reasoning, concept maps, rugplots.

Biographical notes:

Filippo A. Salustri, has been teaching, researching, and practising design engineering for 17 years. He has been involved with research and design of cars, aircraft, spacecraft, robots, temporary structures, toys, home appliances, and medical equipment. His research interests include formal and informal methods of designing, information visualisation, and web-based design tools. He is a member of the Design Society, the Design Research Society ASME, CSME, IEEE, and INCOSE; he is a founding member of the Canadian Design Engineering Network, and a member of the Canadian Design Research Network.

Rob Bracewell is a Senior Research Associate and Assistant Director of the Cambridge Engineering Design Centre. A mechanical engineer, he has previously researched in the fields of ocean wave-energy exploitation, mechatronics and advanced robotics. Recently, his main interests have been in creating software tools to help designers, in generating and evaluating innovative design solutions, and

unobtrusively capturing the rationale behind their decisions. His software tool DRed, researched and developed in close collaboration with Rolls-Royce plc, has rapidly gained acceptance as the standard tool for capturing and communicating design rationale across the company.

Nathan Eng earned his B. Eng. in Aerospace Engineering (2004) and MASc. in Mechanical Engineering (2006) from Ryerson University in Toronto, Canada. His past research work under the supervision of Prof. F. A. Salustri focused on representation and interaction tools for augmenting designer's work in the early phases of design. His design experience has been built by participating and coaching student design competitions through which he has won national awards in aerospace and product design. He is currently studying towards a Ph.D. as part of the Design Process Improvement (DPI) group in the University of Cambridge Engineering Design Centre, under Dr. Rob Bracewell.

Janaka S. Weerasinghe is a Master's candidate at Ryerson University in the Department of Mechanical and Industrial Engineering, where working in the field of collaborative engineering design tools. Under the supervision of Dr. Filippo Salustri, Weerasinghe is exploring the variety of collaborative tools available in order to find effective methods to facilitate the engineering design process for non-collocated design teams. Weerasinghe did his undergraduate studies in Aerospace Engineering, also at Ryerson University, under Dr. John Enright.

Introduction

Much of the information developed and used during the early stages of engineering design is written in natural language text. The authors hypothesise that a diagram is a much better representation of some of the kinds of information used in early design. We believe that using diagrams can improve the effectiveness and efficiency of early engineering design and actually promote innovation by facilitating human cognitive processes.

For the purposes of this paper, we define a diagram as a simplified visual model of a thing, composed of graphical elements annotated with text fragments (symbols, words, or short phrases), meant to capture key concepts, entities, and relationships.

When properly constructed, diagrams are able to present information in ways that:

(a) are easily comprehended by the human mind,

(b) make explicit information that might otherwise be difficult to "see," and

(c) can provide a holistic view of the matter that the diagram describes.

This is supported by the increasing popularity of diagramming tools like concept maps, as well as research in learning (Novak and Gowin, 1984).

Diagrams can be very difficult to draw neatly and maintain "on paper," and this may have contributed to their relative neglect in the past. However, with the advent of powerful and plentiful computers, it is at least conceivable that creating and maintaining useful diagrams can be done very effectively and efficiently.

The authors are working to develop computer-based diagramming tools that facilitate the early stages of engineering design activities.

The task is more than just one of programming a user interface. We have undertaken to study how early engineering design information is "best" represented (where defining "best" is itself an open question) as a form of communication in collaborative engineering settings.

The authors refer to the early stages of engineering designing (hereunder, early design) as those that precede the development of geometric models of products. These stages usually include requirements elicitation and specification, conceptual design, systems design, and embodiment design.

Early designing is typically characterised as follows.

- Only vague, qualitative, and unstructured information is available during early design.
- Information needs to be organised flexibly, quickly, and in a way that is easily comprehended by others.
- The decisions made during early designing are often the most critical for product success because they set the tone and strategy of the whole project, and document a holistic sense of the design.
- Early design information is often most visible to "outsiders" (clients, users) because it is information about how the product will behave and respond to client and user needs.
- Establishing key relationships between product and project elements is vital because these relationships drive the establishment of the system interfaces.

These characteristics are very difficult to address with "linear" textual descriptions – i.e. text descriptions arranged as linear narratives having a beginning, middle, and end. Linear text is very well suited to detailed information presented in roughly sequential order, but it is not well suited to early designing because information accretes during these stages in arbitrary ways influenced by the abilities and experiences of the participants, the needs of the problem at hand and the organisation executing the design, and opportunity. People not intimately connected with the project and with engineering (e.g. clients and users) can find it very difficult to grasp textual descriptions and can become so overwhelmed by detail as to miss general qualities of a proposed design that may or may not address their needs.

The designers themselves can have difficulty linearising their own thinking during the relatively amorphous stages of early design. This means they will invest more time than necessary to articulate their work – time that could be better spent developing the design itself.

Finally, linear text descriptions can rarely capture the depth and intricacy of the relationships between design elements as well as a diagram. We note that individual relationships might be well explicated with purely textual descriptions, but the network of interconnectivity between design elements can be more naturally represented diagrammatically.

There is a tendency for engineering designers to "doodle," to use whiteboards in meetings, to describe key design elements and relationships graphically. This has been observed many times by Salustri and by Bracewell in actual engineering design settings. These diagrams have been observed to be (a) focal points to keep discussions on target, and (b) highly effective ways to help designers keep the overall design concept in mind as they develop details. As such, we see diagrams as a way to augment the cognitive capacity of human designers, rather than displacing those activities onto the tool.

As a tool to supplement designers' innate abilities, diagrams can also help one think differently about problems. The layout of a diagram can bring to light patterns – such as cycles and loops, hierarchies, etc. – that may be very difficult to identify otherwise. These patterns prompt the user to "see" the information differently, which in turn promotes different perspectives on the problem or situation that the diagram addresses. Although there is no way to guarantee innovative thinking, it is more likely that individuals given the means to think differently about problems will find innovative solutions. Thus, diagrams can promote innovation.

We have also noted that while a good diagram can be tremendously useful, a poor diagram can be disastrously ineffective. The questions then arise: how can one distinguish a "good" diagram from a "poor" one, and how can one use that knowledge to implement tools that will facilitate "good" diagramming practices?

While we do not yet have definite answers to these questions, we have made some headway towards possible answers. Obviously, there is a certain amount of subjectivity here – people will respond to a diagram based on their own experience and knowledge. However, there are also certain rules we believe turn out to be quite fundamental.

In this paper, the authors will explain how we are studying the problem of diagramming for the early stages of designing, and indicate our conception of what possible future tools of this sort might be like. The basic breakdown of the work presented here is as shown in Figure 1. While the branch dealing with DRed treats in detail one particular area of design visualisation, the Design Schematics branch takes a broader but less detailed perspective. This approach is meant to provide to the authors the richest possible dataset on visualisation in design engineering as quickly as possible.

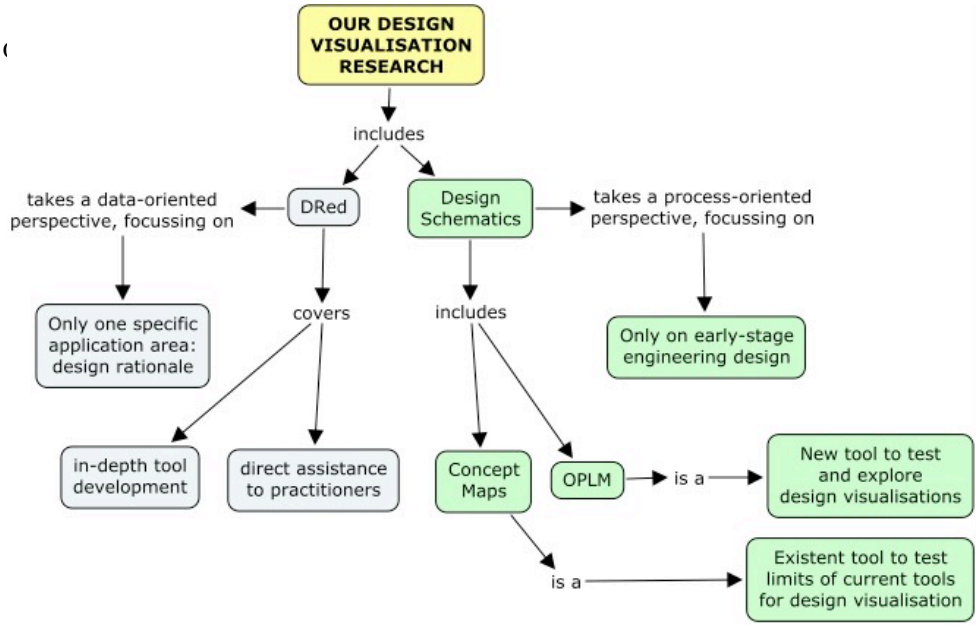


Figure 1 (

The rest of this paper

describes the major lines of inquiry that the authors are pursuing, followed by a discussion of evidence gathered so far that summarises our findings to date.

Lines of Inquiry

In this section, the authors will describe their current lines of inquiry, all aimed at gaining a deeper understanding of the role played by diagrams in design cognition, and developing new tools to support diagrammatic representation of early design information.

As suggested by Figure 1, our research takes a two-pronged approach. One prong is to study in detail one aspect of designing, namely design rationale. DRed, the Design Rationale Editor, delves quite deeply into the needs of practicing design engineers in that regard. From DRed, we expect to gain experience in the minutiae of practitioners' expectations and needs with respect to design rationale. In time, lessons learned from this study will be "folded out" into other aspects of designing.

The authors recognise that such a tight focus may unnecessarily bias our solutions to just that one aspect of designing, so our second prong involves taking a much broader view, which we term Design Schematics. This includes the Opensource Product Lifecycle Management project (and the OPLM "Navigator") on the one hand, and the application of concept mapping to early designing in general on the other. While this part of our work does not delve into specific details of any one part of early designing, we do expect to gain from it a broader overview of general information visualisation needs.

In due course, we will merge the results of these two research efforts, with the intention of attaining a good overall understanding of the visualisation needs of early design engineering.

DRed: Design Rationale Editor

DRed is a simple software tool, descended from the venerable gIBIS (Conklin and Begeman, 1988), which is intended as a more functional replacement to the traditional bound designer's notebook. It is used in close conjunction with a conventional file explorer and web browser, and the designer's standard set of analysis, CAD, office, web and communication applications. As well as capturing rationale from the earliest stages of design onwards, it facilitates the creation and mapping of an on-line design folder, storing all of the files of various types generated during a design project, which is structured according to the dependencies inherent in the design rationale. On completion of the project, the folder can be published on-line using a conventional web server, for future reference within the company.

The rationale is captured using DRed in a set of charts, each displaying a graph of nodes linked by directed arcs, and stored in a single file. Charts appear as zoomable, scrollable, 2-dimensional surfaces of unlimited extent scrolling rightwards and downwards. It is however, considered good practice to limit the contents of a single chart to that which can be printed at a readable scale. There is no inherent hierarchical decomposition of charts – every chart in the design folder exists at the same level. DRed elements (nodes) are normally created, positioned, and linked manually by the user. User feedback has suggested that, particularly in cases where a graph is large and predominantly tree-structured, the option of providing automatic layout would be helpful, and this is a planned future development.

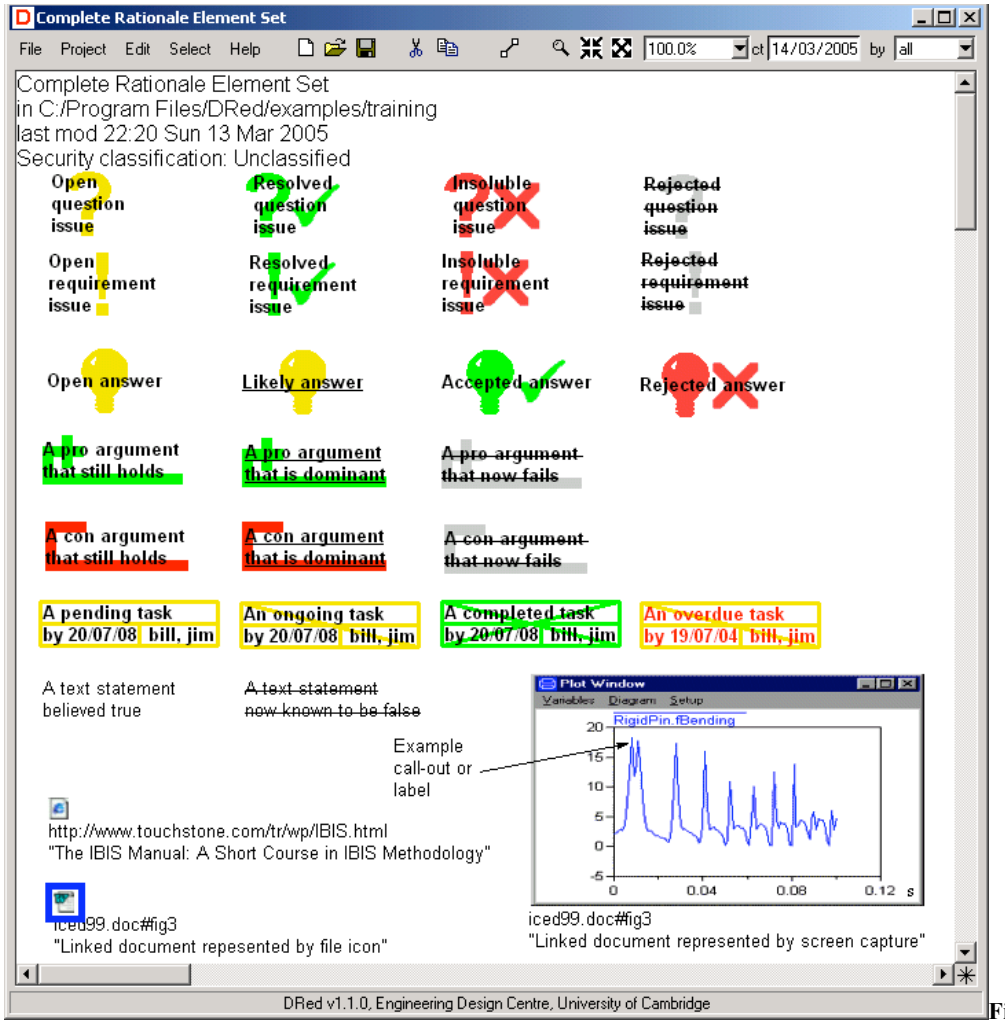


Figure 2 Element types index for DRed.

The user chooses elements, from the predefined set of element types shown in the rows of the matrix layout in Figure 2 namely Issue, Answer, Pro, Con, Text, Task and File. The current set is not claimed to be comprehensive for every possible application, but experience shows that they seem to be a suitable "core set." If additional types are found to be necessary, simple text elements named with a suitable heading can be used in the interim. The graphics aim to clearly distinguish each type from the others. Any element on a chart can be linked without restriction to any other, and any element can easily be converted from its existing type to another. Thus it is possible to use simple text elements for rapidly "brain dumping" and linking ideas, with the decision of whether the elements should be defined as an issue, answer or argument delayed until later. Each element type has a predefined set of statuses, signified by changes in colour and geometry of the background shape or font style of the text (shown in columns of Figure 2). These statuses are changed by the designer as work progresses, generally from "unresolved" to "resolved," although earlier decisions may also be revoked when new information is uncovered or problems realized. This gives a clear view of the progress of the design and allows the knock-on effects of revoked decisions, or the discovery of new information, to be propagated through the rationale. While colour is used to make the rationale clearer to interpret on screen, the differently shaped graphics are designed to make the meaning unambiguous in greyscale hardcopies.

Unlike common practice in previous gIBIS-like tools such as Questmap (Conklin, Selvin, Buckingham Shum and Sierhuis 2001), elements are not represented by an icon and short label, but rather by an automatically resizing text box of arbitrary width and height, the text overlaying coloured graphics to signify the node type and status. The elimination of separate icons saves precious screen area, allowing larger "knots" of related elements to be arranged and viewed at a glance, making the rationale easier to appreciate, as well as being consistent with Tufte's guidelines for "good" design (discussed below). With a typically complicated and tightly packed rationale, anecdotal evidence also suggests that the brain finds it easier to register an element's type and status from a graphic behind the piece of text rather than adjacent to it.

DRed elements have no hidden design information that is only visible by invoking an attribute dialog. Everything captured is directly visible in the form of text and graphics. This means that a printed hardcopy is nearly identical to the screen rendering. Similarly, complete and comprehensive rationale graphs can be pasted easily into reports and presentations. Such considerations have proved very important in the acceptance of DRed in industry.

Unlike most other gIBIS-derived tools, DRed only has a single type of link, a unidirectional arrow, which represents some sort of dependency. The meaning of that dependency is inferred from the types of the elements at each end of the arrow. This is because feedback from users indicated that they found the meaning of links clear from the types and contents of the nodes, so forcing them to specify the link semantic explicitly seems to be unnecessary. As a rule of thumb, when the designer changes the status of any element, arrows should generally point from it to all other elements, the statuses of which should be reviewed as a direct consequence of that change.

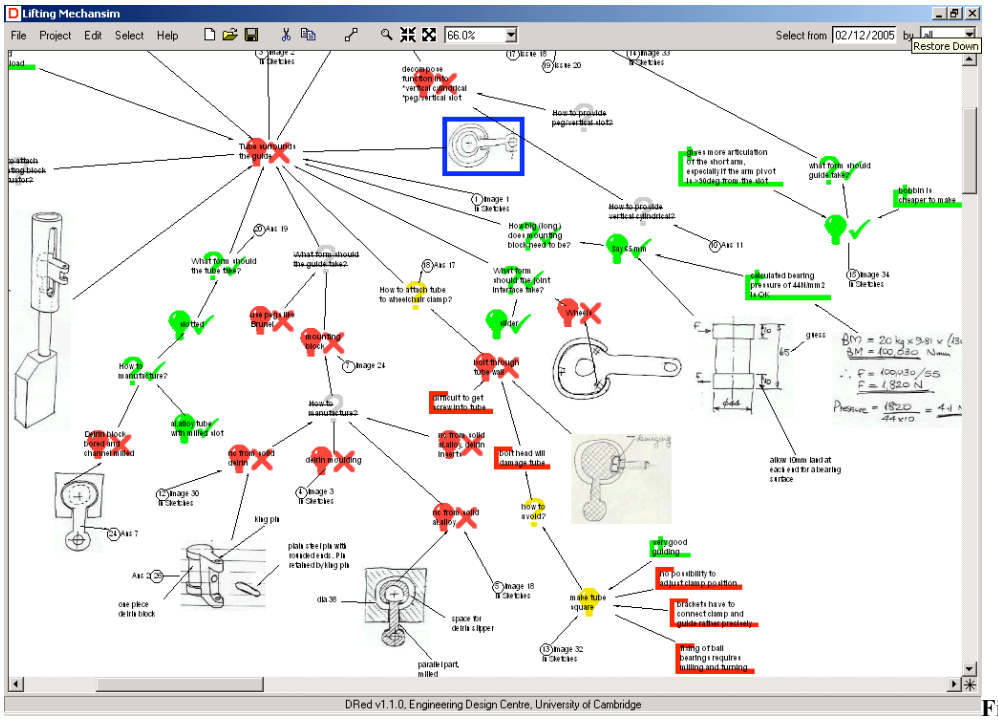


Figure 3 Part of DRed rationale for a wheelchair mounted arm support.

Figure 3 shows part of the conceptual design rationale for a wheelchair mounted arm support, intended to increase the independence of people suffering from various neuromuscular disorders, such as muscular dystrophy. These people are wheelchair bound and suffer loss of function in the large joints of their arms, but some wrist and finger strength may remain. The device allows wheelchair users to move their hands into a position where they may make use of their residual strength. The displayed part of the rationale shows the exploration of the options for the design of the vertical prismatic joint which is motor driven to support the weight of the arm at a chosen height. The issue is whether the slider should run on the inside or outside surface the tube, and, for the former option, how the internal slider might be fabricated. Note how different decision points and activities are evident by the tree structure and node types, and how sketches and images are incorporated into the chart.

Dependencies between elements belonging to different charts are made via *tunnelling links*. These appear to tunnel into the work-plane, reappearing elsewhere and continuing on to their destination element. Tunnels may also be used on a single chart to avoid links crossing confusingly, or where the two linked elements are so widely spaced that they cannot both be viewed without scrolling. Each of the pair of tunnel mouths is shown as small circular icons. Several of these can be seen in the example DRed chart show in Figure 3. Double clicking a tunnel mouth, or hitting the return key while a tunnel mouth is selected, moves the mouse pointer "through the tunnel" to the opposite mouth, displaying its chart and leaving the mouth visible and selected. Thus having glanced at the surroundings of the far end of the tunnel, the user can immediately return if desired by double clicking or hitting the return key once again. Such links permit the rationale for larger design projects to be distributed across multiple charts, and laid out legibly, while facilitating navigation between them.

Files from any other application used as part of the design process may be linked into the rationale using elements of the *File* type. If these files are created as part of the design project, they are stored in the design folder and referenced by a relative file path. Alternatively, the document may be on a remote server. The default is for the file element to be displayed as a small icon representing the type of the referenced document. However, as an alternative, a screen captured bitmap of the document contents can be displayed on the DRed chart. DRed links anchored to an element normally terminate on the element border, positioned to point at the centre of the element. However, for these bitmap graphical elements, links can be anchored to a specific location in the graphic. This enables for example, references to CAD files to be displayed within DRed as a view of the screen captured from the tool, with links anchored to the location of individual features, labelling or expressing issues related to them. By double clicking the file element, the referenced document is loaded into its software application and displayed. As shown in Figure 3, the File element can also be used to capture freehand design sketches, whether optically scanned from paper, or if DRed is used on a Tablet PC, directly drawn on the computer screen using the pressure sensitive stylus. The use of File elements to associate files with related nodes in the rationale, allows DRed to provide a unified navigable map of the emerging design folder. Once finally completed and approved, the folder can be made available for searching and read-only browsing, by simply copying it into the file space of a standard intranet or internet web server. The repository is then accessed by using a standard web browser, configured with DRed as a helper application.

Research and Implementation of DRed

As described in detail by Bracewell and Wallace (2003), a vital factor in the successful research and implementation of DRed was how the software application "Graphlet" was employed. This is a general-purpose interactive tool for creating and manipulating node-arc graphs, written at the University of Passau (Himsolt, 2000). In the early stages of this research, it was used as a knowledge-modelling tool, to explore the applicability of IBIS-like representations in graphically structuring the DR contained in existing design reports. Once this process had led to a promising graphical DR representation being proposed, Graphlet was used as the basis to create a tool for designers to use, by gradually modifying its general-purpose graph editing GUI to give an effective tool for DR capture. It was recognized from the start that, to be employed successfully, the tool would need to be as intuitive as possible to anyone familiar with the conventions of Windows diagramming software such as MS Draw or Visio. Graphlet does indeed follow these conventions.

However, the crucial feature of Graphlet for this application is its use of a "two language" programming philosophy, in which a robust and efficient compiled core underlies a configurable user interface written in a high-level interpreted language. This approach has been used successfully in a number of design research projects, for example n-dim (Dutoit, Levy, Cunningham and Patrick., 1996), and Schemebuilder (Bracewell and Sharpe, 1996). Graphlet's core is a very solid graph-processing library, and beneficially, the implementation of DRed required no modification of this. By progressively customizing Graphlet using pure scripting, it was possible to produce a DR capture tool, of steadily increasing power, effectiveness and ease of use, with a reasonably polished graphical user interface from the outset.

A common bottleneck in this type of research is the time consuming and difficult problem for the researcher to convert partially worked out research ideas into a sufficiently precise specification for satisfactory implementation by a software developer. The only way to avoid this problem completely is for the researcher to write the software personally, which can be very effective if productive choices of languages, components, and tools are made. This was the approach taken here, with all the software being written by Bracewell. Thus, it was possible to release, in less than one month of software development, an initial prototype DR capture tool, named DRed v0.1.

While very limited, DRed v0.1 was sufficiently functional to support real design work by a small group of collaborating designers at Rolls Royce. Moreover, due to its solid foundations in the unmodified underlying graph library and the run-time error checking of the modified and newly written scripts, it proved robust enough for designers to use with confidence. First reactions were very positive, and accompanied by many helpful suggestions with regard to functionality and usability. There were also indications that its use in recording the design process, rather than being a hindrance, was actually proving beneficial in lending structure to designers' thoughts. An efficient and productive research cycle of idea generation, implementation, and testing ensued. Seven further releases, of steadily increasing capability and refinement, followed over the next eight months. The tool in its resulting state was successfully evaluated in a six-month trial with forty designers (Bracewell et al. 2004).

Three years of further development, steadily increasing use, and generally favourable reports followed, leading to its acceptance as part of the standard Product Lifecycle Management (PLM) toolset, and deployment across the company worldwide. The "Discussion of Experiences" section later in this paper includes one such

favourable report: the use of DRed to support the conceptual design of an innovative power offtake, which provided a very significant improvement in fuel consumption, for the latest Rolls Royce civil aero engine.

Concept Maps

Salustri and Weerasinghe have been examining the use of *concept maps* in early design.

Concept maps have been quite successfully used as learning and learning assessment tools. In particular, our research uses an existent concept-mapping tool called CmapTools (http://cmap.ihmc.us/). Its developers assume a network-based model of learning wherein new information is absorbed by interconnecting it with existent information in the mind (Novak and Gowin, 1984). The more richly interconnected the information is, the "better" learned it is.

We suggest that designing – especially in the early stages – is a learning exercise. The acts associated with specifying the initial requirements, concepts, and systems of a product are acts of exploration and discovery, which result in learning. Once a design problem is understood well enough, at least part of the design answer is usually quite apparent. Indeed, many people think that design problems and their solutions evolve concurrently. In this view, then, effective design tools promote the integration of new information into a designer's mind by being compatible with the network-based model of learning described here, and for which concept maps were specifically created.

Concept maps consist of *nodes* connected by *links*. A node is a geometric shape (usually a rectangle) containing a text fragment that represents a concept, idea, item, or question. A link connects two or more concepts together, and represents a relationship between the concepts. Links are labelled with a text annotation describing the nature of the relationship. The links can be unidirectional or bi-directional, depending on the nature of the relationship. In order to achieve a higher level of organization, graphical properties of these elements can be changed. The style of nodes can be altered to group common concepts by colour, shape, or font of the node's text. The style of links can also be altered to reflect various types of relationships between nodes, including line thickness, arrowhead style, colour, and style (e.g. dashed versus solid lines).

Concept maps make no specific commitments to what such style variations mean. Instead, one expects the user to establish conventions regarding style that are meaningful to the expected "readers" of the map. In a design setting, this means that each user community can establish their own norms, thus increasing flexibility while allowing the groups to develop their own semantics for the maps, thus improving the capacity of the maps to convey knowledge.

The graphical layout of nodes and links in concept maps is perhaps the most significant factor in facilitating comprehension. It is easy to create a "randomly arranged" concept map that is very difficult to understand. It is much harder to rearrange the elements to bring out particular information features. One obvious example is a cyclic structure. The cycle itself, once identified, can mean something quite specific to the users. However, in even modestly complex concept maps, arranging the layout manually to explicate these features can be very difficult. To assist in this, many concept mapping tools, including CmapTools, provide *auto-layout functions* that let users reorganise concept maps at the push of a button. While this functionality can be somewhat useful, it is also limited by the lack of semantics explicitly associated with map elements. Thus, we expect that new layout functions will have to be added to CmapTools eventually, if concept maps are to be effective design aids. However, it is not yet clear at all what kinds of layout may be most expressive for design situations. Indeed, discovering the concept map layouts that best represent design information is essentially equivalent to determining the kinds of knowledge structures that designers use.

We are using CmapTools in a project on collaborative design within the Canadian Network of Centres of Excellence called The Automobile of the 21st Century (http://auto21.ca). In this project, principal investigators from several Canadian universities are studying designing in the North American automotive industry, to develop new tools to facilitate collaboration. To conduct the study, the team is installing existent collaboration tools; student teams will then undertake the design of a *small urban vehicle* using those tools. Other students will observe and monitor their work. TeamCenter, a product lifecycle management software system by UGS, was selected. This package is widely used in the automotive sector and readily available to the researchers. However, TeamCenter does little to support the relatively unstructured and qualitative information that is vital to early design. Indeed, the team was unable to find a suitable tool for the early stages of the project. We have thus adopted CmapTools for this purpose.

CmapTools has a server element, which allows concept maps to be shared via the Internet. We have set up the server so that students from all participating schools are able to develop concept maps about the urban vehicle. These concept maps will capture both the design activities the students undertake and some aspects of the vehicle design itself. Weerasinghe will then analyse the concept maps generated by the team as well as its activities to create and use those maps to address two questions. Firstly, what are the benefits of using concept

maps to help capture, use, and communicate early design information? Secondly, what modifications or enhancements to CmapTools would improve the tool's effectiveness as a design aid?

As an example of the kind of concept map being described here, we refer to Figure 4, which was constructed with CmapTools to capture the nature of the Auto21 project at the end of its "kick-off" meeting. This single diagram captures the goals of the project, the participating groups, the major milestones, and the overall research activities. As such, this kind of diagram can become a touchstone for the project, to keep all work properly grounded and directed. The content of the nodes in Figure 4 is not as important here as are the obvious sub-structures of the project that are evident from the layout of the diagram. The coupling between project elements, as determined by the meeting participants, is also obvious. This attracts the attention of the participants, to ensure that complexity implied by the coupled elements is properly accounted for in downstream project activities.

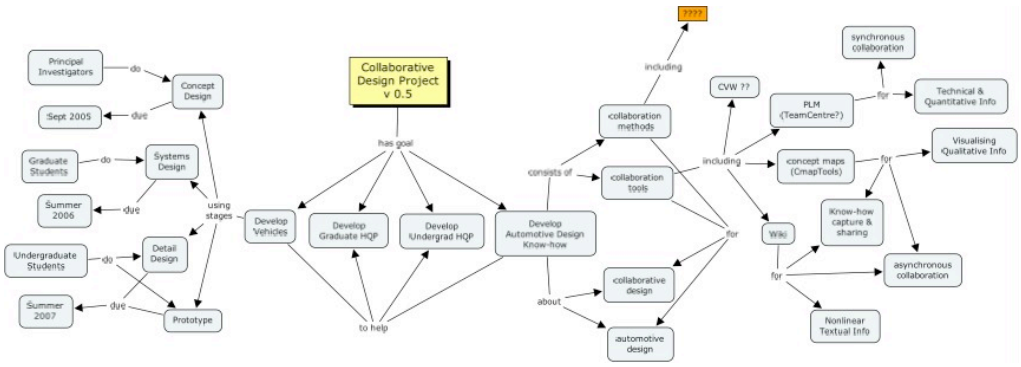


Figure 4 A concept map showing a tentative overview of the Auto21 project on collaborative design. Recognising the content of the nodes is not as important as noting the obvious substructures made evident by the layout.

C. Visualizations for Opensource Product Lifecycle Management

Salustri is also working with Eng to develop a visualisation system for *product lifecycle management* software. Product Lifecycle Management (PLM) is a comprehensive approach to managing a product throughout its life. It can provide faster time to market for new products, improved design revision control, greater design for sustainability, and re-use of work from previous projects.

The Opensource PLM (OPLM) project aims to provide SMEs with the ability to do effective PLM without the typically prohibitive resource and cost requirements of the larger commercial systems. Opensource software has been demonstrated to garner significant support bases, thus possibly replacing the need for some commercial PLM software providers (Dreiling et al, 2005).

PLM applications deal with complex, massively interconnected information. The currently available information interaction methods are insufficient to manage the thousands of disparate information elements associated with a product's lifecycle. Furthermore, contextual and implicit information must be included *to enable a higher-level of thinking and decision-making* in the product's design.

This is not just a question of data interaction, but one of knowledge management. To this end, the first part of the OPLM project is focused on the creation of intuitive, computerised, visualisation tools (a *visual navigator*) to interact with PLM data repositories.

Shaping the Visual Navigator

Modern computers give designers unprecedented information and data management capabilities, but they cannot yet manage the qualitative and unstructured information of early design. The navigator must connect the information access capabilities of computers with the creative abilities of the designer. Visualization is the key because it enables the highest density communication between these two types of processors.

Even an exceptionally broad-based company like IBM, which itself markets PLM systems developed by Dassault Systemes, does not have the kind of diagrammatic visualization system prompted by our research. They do, however, seem to have an understanding of the value of such a system:

> *"Visualization can help you find solutions to key business problems by presenting that data in a form that allows for rapid detection and resolution of potential problems that could hamper*

manufacturing, cause production delays and result in increased production costs. Moreover, visualization will allow geographically distributed teams to collaborate efficiently towards better, more economical and successful products. If you could see, really see, the scope of the data and how disparate datasets interrelate, you could make that insightful breakthrough." (IBM, 2005)

The tool advertised in this IBM document, however, is one for securely sharing display content, not one for actual information rendering and *shaping* for human consumption.

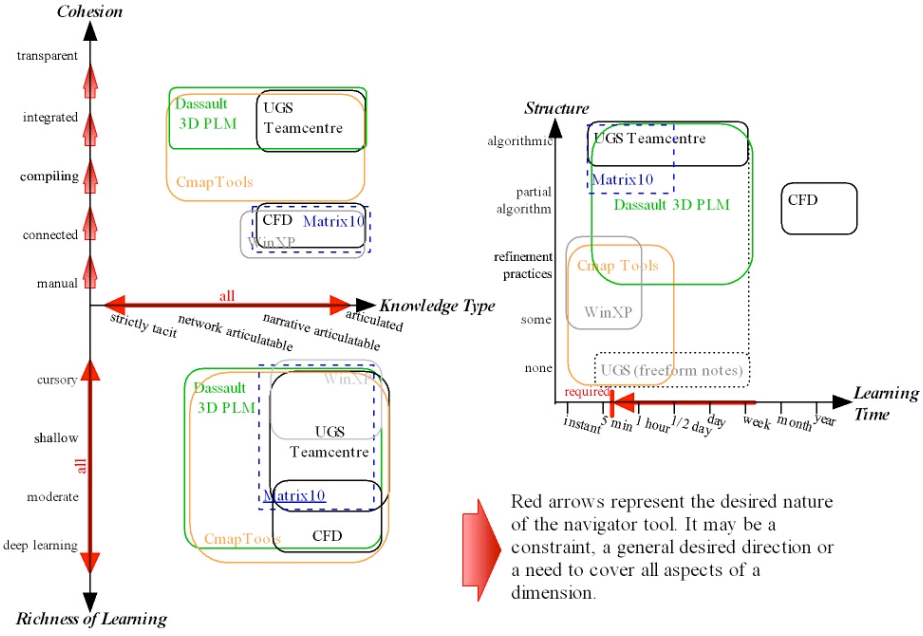
The analogy of *information shape* can be used to further characterize the function of this navigator tool. There are many subtle differences between the activities inferred by "navigating" information versus "shaping" it. Dillon (1997) provides several arguments for the preference of the space analogy. Navigation is just about finding things, whereas *shape* suggests many more dimensions and modes of interaction.

The navigator can thus be re-described as a means to sample the *shape* of information, and then *encode relevant* aspects of that shape into a meaningful visualisation. As the designer manipulates that visualisation, the tool can capture the input by reversing the process and applying the changes to stored information.

Defining the meaning of *shape*, *relevant* and *encode* is the navigator research problem in a nutshell.

Qualitative Diagrams in the Early Design of the OPLM Navigator

Creating methods for visualisation is the *encoding* problem in this research. To begin the solution, we first examined existent software solutions, ranking them on a "rug plot" diagram (Figure 5), per (Tufte, 2001), against knowledge level, information structure imposed or available, richness of learning, speed of learning, and cohesion. The rugplot's detailed construction and use are presented in (Eng and Salustri, 2006). That material is summarised here, followed by the results as they pertain to the design of the navigator.



(All information is accurate based on author's research from available media from manufacturer's websites and print media)

Figure 5 This figure is the result of the "rug-plot" process applied to some popular tools from the fields of information technology, engineering, and PLM. Note the volume of material needed to textually express the contents of the graphic. This is a good illustration of the expressive power of diagrams. The unpopulated areas of the charts are areas where a successful *visual navigator* would provide unique functionality or usability.

Visualization guidelines with examples of their application are as follows:

- Each element is multifunctional: axes show constraints and colour-coded data act as a legend.
- The total of all data, measures, and targets are juxtaposed on the final diagram for easy cross-comparison.
- The axes are based on differences that are significant to the user instead of attempting to fit an obfuscating numerical scale.
- There is micro-scale composition in the individual data squares showing each tool's characteristics while there is also macro-scale information in the overall shapes of occupied and vacant spaces in each space.

- To avoid clutter, the axis text labels are also division marks and the data are offset slightly to make differentiation clearer.

In sum, this visual implements the visualization concepts that would make the OPLM Navigator work well. This could ultimately represent one of the "views" within the navigator environment.

The rugplot itself was generated through a five-step process:

- Define the "dimensions" by selecting characteristics of interest and dividing them at a useful level of detail.
- Measure each tool to find its "shape" within that dimension.
- Select the dimensions that show differentiation between the measured material and the desired design.
- Plot the results on various axis pairings.
- Look for the causes of open spaces, crowded areas, and correlations.

The results are presented below with the selected axes of knowledge level, information structure imposed or available, richness of learning, speed of learning, and cohesion.s

All of the infrastructure, time, training involved in PLM must result in better decisions. This means better information propagation and better knowledge transfer. This is why "knowledge type" and "richness of learning" were examined in the rug-plot from the outset. The divisions of knowledge were expanded from the descriptions in (Davenport and Prusak, 1998) and the "richness of learning" divisions are based on the networked learning model for "meaningful" learning presented in (Novak and Cañas, 2006).

It was also understood that any newly introduced tool would need to find acceptance among the established practices in a workplace. The key barrier is expected to be the amount of time required before a new user generates new work because of the novel tool. "Learning time" is a measure of this.

The PLM-specific tools reviewed in the rug-plot seem to lack a number of functions (shown in Figure 6) such as:

- capturing unstructured knowledge and information typical of early design and required for "meaningful" learning; and
- providing an accessible means of quickly getting new, relevant work done.

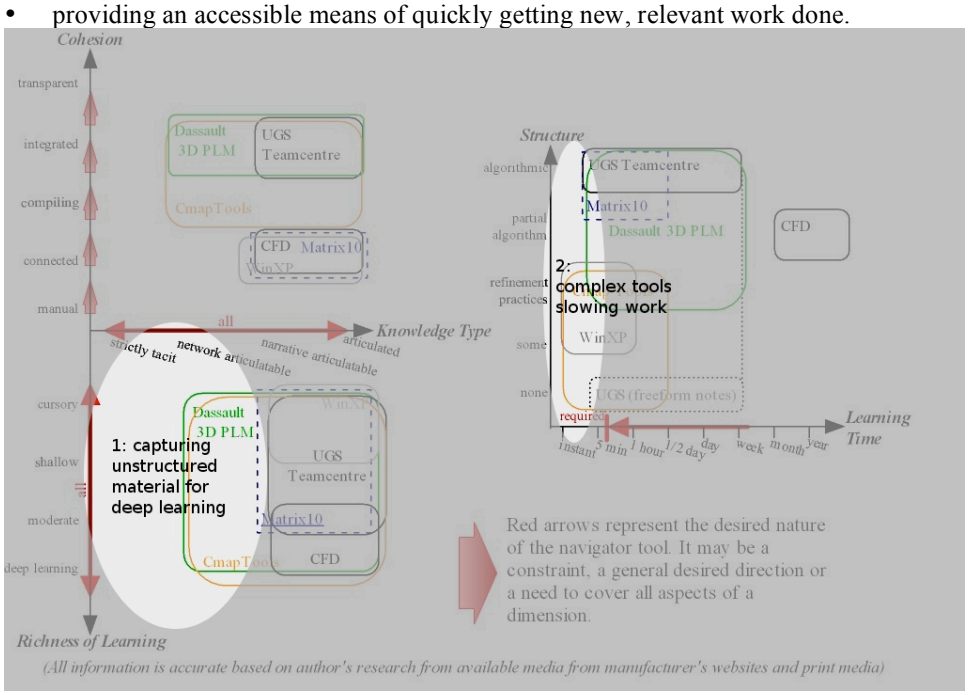


Figure 6 Highlighted areas of interest in the rug-plot.

Conversely, we note the flexibility of CmapTools in "breaching" the edges of the areas of interest. This suggests strongly to us that concept maps are a good starting place for the OPLM navigator.

Some free-hand sketch tools are available in many of the reviewed packages. They allow the addition of an unstructured graphical layer on top of some physical visualisations. One can even attach specific textual comments to locations on an image. However, those comments lack the connectivity of a hyperlinked network diagram (like DRed with its tunnelling links) and hence fail to represent fully the depth of interrelationships

between information items. Ideally, the tool must let the user see "decision points" associated with geometric and conceptual features of the design. A complete PLM tool must answer transparently questions like:

- What did the designer, engineer, or accountant use to make a decision?
- Who made the choice?
- How does it fit in with the problem definition or the inherited functions of this sub-system?
- What previous decisions constrained this outcome? Will this one unnecessarily constrain the next?

A specific example stems from a software package presented for the Auto21 project mentioned in the section of this paper on Concept Maps. One of the key features was the software's ability to highlight geometric incompatibilities in an assembly of parts originating in different CAD environments. Objects could be assembled and interferences, if any, would be highlighted. Geometry, however, is a product of a need. A shape may need to fill a space, leave a gap, or carry a load. If a designer cannot see the physical model in the context of a stress analysis result or a functional constraint, wise resolution of geometric issues become problematic because the knowledge was not adequately captured. In view of this, it was noted that the software being demonstrated could not combine the CAD data with additional information like finite element analysis results. This would add a great deal of value when making decisions about the resolution of unwanted interferences.

This is an example of the need for *context* in PLM. Comparisons can be made between any two pieces of information. *Useful* comparisons, however, must be guided by knowledge of the problem and the connection between particular issues and the related elements of the design.

Developing the OPLM Navigator: a New Environment for Early Design Phases Integration

The OPLM navigator is being developed through a combination of research and practice. Background research involving visualisation, knowledge management, user interface design, and design methodology provides a basis for initial design decisions. Useful elements of the theoretical research will eventually be compiled in the form of *design patterns* (Salustri, 2005) to capture the theoretical knowledge and thus make the knowledge and information accessible to future navigator developers.

In engineering applications, utility rules all. The compiled interface concepts are assembled into working prototypes or mock-ups for testing by real users. An opensource prototype visualisation test-bed is being written in Tcl/Tk so that style and layout algorithms can be tested for feasibility and effectiveness. The following sections provide further details of this work.

To develop the navigator prototype, we reviewed several alternatives. The current prototyping tools we have available are as follows.

- Manually drawing maps allow un-constrained drawing and testing of map features. They are also useful during group sessions because much of the value of visual maps is still obtained in a hand drawing.
- HTML-based image maps simulate a "dumb" front-end visualisation with some of the behaviours of a computerized tool.
- Existent concept mapping tools, like IHMC CmapTools, provide basic functionality and a simple test bed for layout and manipulation research. It can also create HTML image maps for portability.
- A prototype diagramming tool written in Tcl/Tk demonstrates solutions that are more comprehensive and gives us greater control to explore new visualisation methods and interfaces.

General Navigator Specifications and Concepts

The background research and experimentation has lead to a set of preliminary design specifications at various levels of abstraction that will guide future developments. The OPLM Navigator must:

- Convey the *shape* of *relevant* information through an effective, meaningful *encoding* method.
- Be intuitive and effective enough for users to quickly generate new and "significant" work on product development problems.
- Emphasize task-focus; the less one thinks about how to use the tool itself, the more complex a problem one can solve.

- Adapt to fit with each company's existing tools and each user's needs at a given time. The users must not have to adapt to the tool.
- Work and flow through information in a way that compliments and captures the "thought path" of an individual solving problems.
- Emphasize utility over technology – use it where it works, defer to other tools where they're better.

The following sections outline the best ideas from the current navigator concept followed by further details on the prototype. The figures in this section were generated with SmartDraw 7 for convenience; they are not interactive, but the "TclMap" prototype presented in a later section is a working tool.

Zooming-User Interface paradigm

Assuming the tool will have control of the overall computer interface, the most appropriate scheme seems to be the *Zooming Interface Paradigm* (ZIP) proposed by Raskin (2000) and demonstrated by elements of DRed. The key requirements for the OPLM Navigator ZIP are:

- All levels of detail are accessible from any point in the space.
- Zooming and scrolling functions are always available.
  - Zooming would be a basic alternate mode available through a "pseudo mode" key (such as a mouse button or scroll wheel). It makes navigation faster because one can zoom out and back in precisely, without loss of context, much faster than one could scroll to some unknown point off-screen.
  - Scrolling occurs when the mouse cursor is moved to the edge of the screen. Scroll rate should be proportional to the "push" past the edge of the screen. This is an intuitive way of moving past the visible limit of the model without the inefficient use of scroll bars that slow down the scrolling and distract from the task.
- Clean interface; minimal persistent elements helps increase information density and maintain focus on the task.
- Although it is ill advised to "hide" functions, editing is considered secondary to navigating and general interaction with the map. That is why editing functions are not always visible (see also the next section on radial pop-up menu).

This is a significant departure from the standard PC/Windows navigation system. Simplicity in the design and context-sensitive documentation should ease the learning curve for this tool. The following sub-sections elaborate on adaptations of Raskin's work (2000) to OPLM.

Radial Pop-up menu

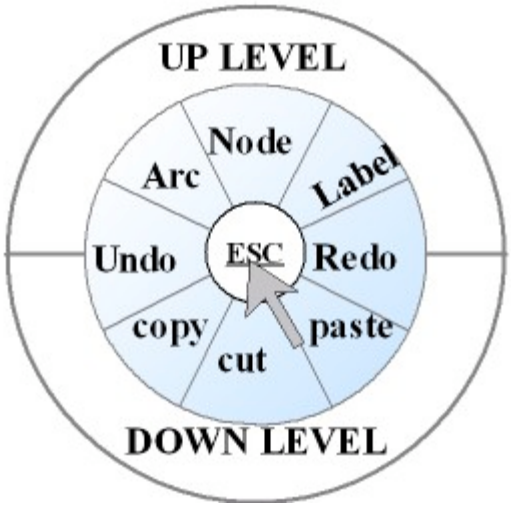


Figure 7 Proposed layout for radial pop-up menu. It is designed to take advantage of not only spatial cues, but also cues based on body/arm/hand position and direction.

The use of a radial pop-up menu (Figure 7) replaces a static toolbar with a persistent, context sensitive tool. It has the advantage of lessening screen clutter and increasing command access speed. The most quickly accessible pixel on the screen is the one right under the pointer. Context sensitivity minimizes the number of options it has to contain. A comprehensive pop-up menu means all options immediately visible and useable. There is also no distraction to look for a command by looking away from a point of interest to scan the edges of the screen, so the user can remain task focused instead of interrupting a train of thought just to execute a command. Similarly, "escape" is provided as the first option right below the cursor when the menu initially opens. This fast "undo" option lets the user quickly continue their activity if they accidentally activated the menu. The usual way of exiting these menus (at least in MS Windows) involves clicking on some random location outside the menu or hitting the actual escape button on the keyboard. This loses one's position before the error and risks activating some other element on-screen which would cause yet more delay and distraction.

The key feature of this menu is its radial layout. Most pop-up menus are rectangular. Once a user becomes accustomed to it, the radial organization should result in faster, more accurate access times because:

- There is the same minimal distance for the cursor to travel to reach each first-level option. The access speed of these multiple items is akin to that of the second item on a drop-down list.
- The direction becomes part of the input, adding to the differentiation between the options. This is more accurate because it is easy to overshoot an option on a linear menu, where everything is in the same direction. It is much less likely that a user will accidentally move the mouse a significant distance in the wrong direction in a radial menu.

Finally, this menu uses words to indicate commands instead of icons. Words are more effective than icons for displaying commands because, unless a user is familiar with all of them, icons will hide meaning (Raskin, 2000).

Reasoning Path Map

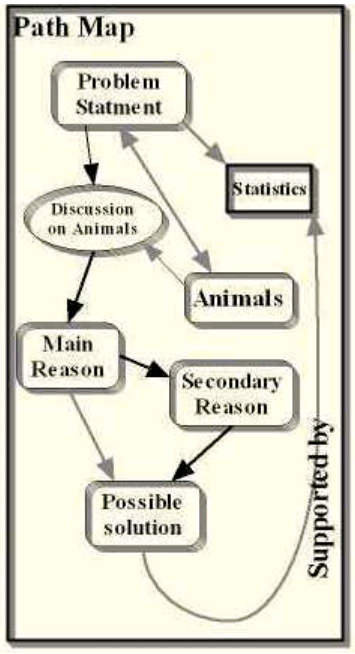


Figure 8 Concept for *reasoning path map*. This map would be generated automatically by a PLM tool and driven by recognition of key action types. Though it must assume a certain general model of activities, this solution can still capture distinctive user-based and task-based characteristics.

The *reasoning path map* (see Figure 8) acts as an enhanced navigation history tool. All arrows represent significant navigation actions that are automatically generated as a user works on the PLM "shape." This tool aids navigation by jumping the navigator to a previous state by simply selecting a node. It can later be reviewed and annotated with specific arrow types (black ones in Figure 8). Unlike a normal "history" function, it also allows branching from a given point. This reasoning path was found to represent the conclusion or to most effectively articulate the work done. It is an instant capture and articulation of design reasoning. It also

supplements current user thinking by providing a reasoning structure. Conclusions can be fully referenced and "re-played" for review and feedback by other users without re-formatting the work as an email or even a conversation. This is a way to automate some "book keeping" aspects of design work.

D. Design Schematics

Finally, Salustri has also been working to develop a generic but integrated set of diagramming tools and for early design, with particular emphasis on requirements specification, function design, and systems design. Rather than focussing on tool development, this work has been theoretical, involving the study and discovery of rules that apply to design engineering visualisation.

We have already found that there are some patterns of diagrammatic layout that are very easily learned – indeed, as described below, we have found that diagrams can be quite intuitively grasped by users without any "training." Four such rules we have found are as follows.

1. Smaller nodes fully contained in a larger node are intuitively interpreted as constitutive elements of the larger.
2. Partial overlap of two nodes is usually interpreted as indicative of some functional, rather than structural, coupling.
3. Both students and engineering practitioners prefer links between nodes to be combinations of horizontal and vertical element (Figure 9, left side), rather than single straight lines at arbitrary angles (Figure 9, right side).
4. Small radii rather than sharp corners significantly help viewers of diagrams to "see" the flow indicated by an arc. For example, in Figure 10, it is not clear if nodes B and C are connected directly to each other; however, the radii used on the right side of the diagram make it clear that there is no direct connection between nodes D and E.

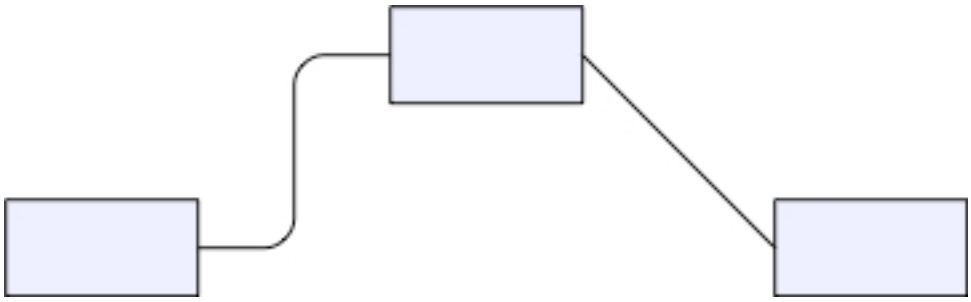
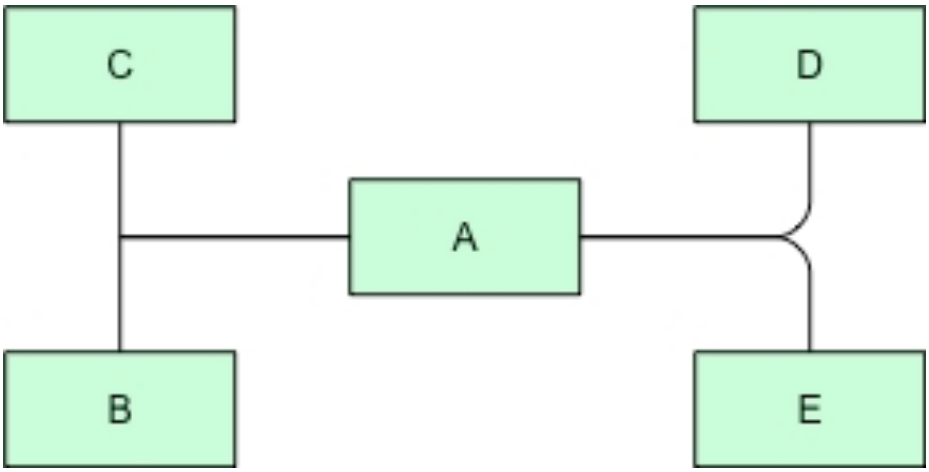


Figure 9 Engineering designers prefer rectilinear links (left side to angled ones (right side).

Figure 10



Simple radii can clarify the "direction" of links. On the right side, it is intuitive that there is no direct connection between nodes D and E. However, on the left side, whether there is a direct connection between nodes B and C is ambiguous.

Having identified these rules, Salustri, Eng, and Weerasinghe are working to embed them into software. The OPLM Navigator discussed above is the platform in which we are doing this (details to be given in the following section).

We recognize that no single diagram can represent all pertinent early design information well. Therefore, we define design schematics as a *set* of diagram types that have common elements. A *product design schematic* (PDS) identifies product characteristics, functional requirements, constraints, and performance metrics (Salustri and Parmar, 2004). A *product architecture schematic* (PAS) reuses functional requirements from a PDS to let designers connect functions with links denoting transfers of mass, energy, and information; PAS diagrams are similar to the "architectures" of Stone et al (2000) but constructed in a top-down manner rather than the bottom-up approach of Stone et al. Other diagram types, such as a *usage schematic* for use cases, are being developed.

Each diagram represents a user perspective on the design problem. The shared elements facilitate maintaining cognitive connections between the diagrams as the design evolves. Eventually, when implemented in software, users will simply request a change of perspective and the appropriate elements will be selected and displayed in the best possible form. Currently, we are using the OPLM Navigator software (discussed below) to test our notions of what constitutes a "usable" diagram.

Discussion of Experiences

Design Rationale and DRed

Having been accepted by Rolls Royce's Design Process Council, DRed has been incorporated into the company's standard Product Lifecycle Management (PLM) tool set. DRed is now the required presentation tool for design rationale at Design Scheme Review meetings. While jet engines for civil airliners are generally characterised by evolutionary rather than radical design changes, an example of the effectiveness of DRed in supporting conceptual innovation is its recent use in the design of the power offtake of the Trent 1000 engine, used in the new Boeing 787 Dreamliner. Commercial confidentiality prevents the publication of the actual rationale charts created. However, we can report that the innovation and resulting performance benefits of DRed are demonstrated by the Trent 1000 design being singled out as follows, in the brief sales description of the engine, on the Rolls Royce website:

http://www.rollsroyce.com/civil_aerospace/products/airlines/trent1000/engine.jsp

"Unique Intermediate Pressure power offtake design innovation delivers:

- *lower fuel burn (for both long and short missions) – up to 6% fuel saving on short haul routes compared with High Pressure power offtake systems*
- *lower idle power for low noise and low brake wear*
- *better engine handling characteristics*
- *direct to bottom line cost saving for airlines."*

A 6% saving in fuel burn for short haul routes is of course a huge improvement in the context of civil aero engines, and only achievable by significant concept change rather than the usual incremental improvements. The role of DRed in supporting this innovation was described as follows, by a leading member of the team responsible for the design of the Intermediate Pressure Turbine power offtake: *"DRed proved to be a powerful tool when considering the complex issues surrounding the many concept options. It enabled rational down select of the leading designs. This has proven to be a valuable reference pack throughout the design evolution."*

Further evidence of the perceived value of the tool, in supporting design and innovation, is that the team responsible for its research, implementation, and introduction into the company's design process won the Rolls Royce Research & Technology Director's Creativity Award for 2004.

TclMap: A Testbed for the OPLM Navigator

Figure 11 shows the current state of the navigator test program. Its basic feature set is modelled after IHMC CmapTools; however, it implements a number of modifications that move it towards the system concept defined in the previous section:

- Always-available zoom using mouse scroll wheel;

- Panning by “dragging” the canvas;
- Continuous text feedback in the status bar;
- Arcs are constrained to remain vertical and horizontal while only bending with rounded corners;
- Commands are accessible only through the pop-up menu; there are no toolbars to clutter the diagram area (see Figure 12).

The first two points relate to moving the tool towards a ZUI. The third is both a development tool and a continuous-learning tool. The last is a product conclusions presented in the Design Schematics section. The path generation and node anchoring of these types of links is still an open issue. Hiding links behind nodes is worse than having diagonal ones.

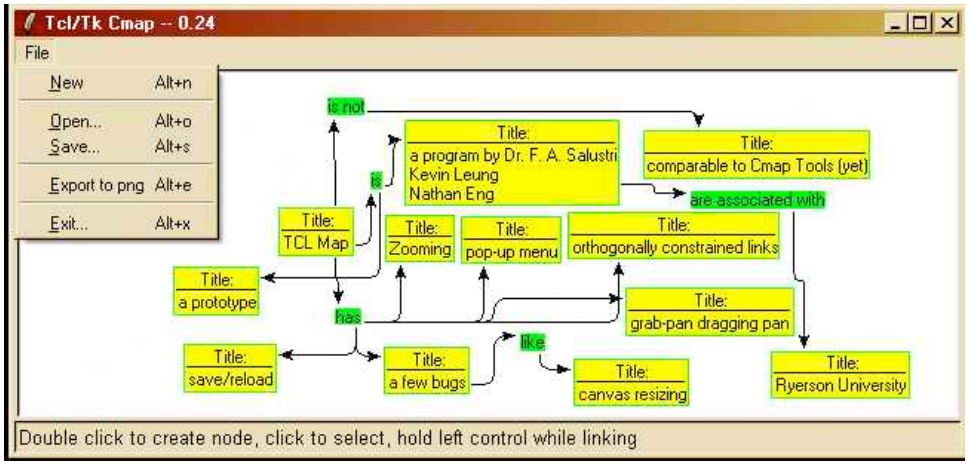


Figure 11 TclMap screenshot (using ActiveState ActiveTcl8.4.9.0 on Microsoft Windows XP).

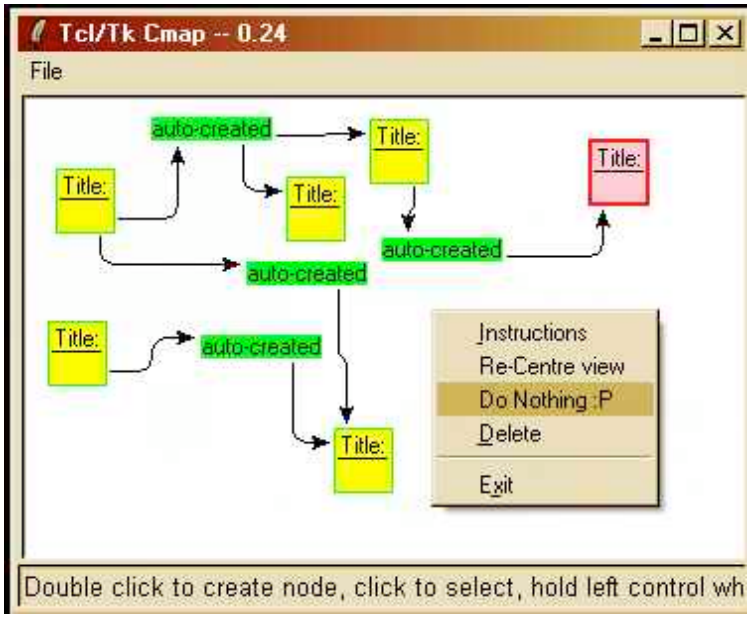


Figure 12 TclMap linear pop-up menu.

Tcl/Tk was chosen as a development platform because of its maturity and cross-platform compatibility. This tool was developed and tested on a Sun Microsystems workstation, Microsoft Windows PC and Apple G5 Mac without significant portability issues. Most of the latter are absorbed by the Tcl/Tk interpreter. The current version is available on-line from the project's web page at http://deseng.ryerson.ca/xiki/Oplm/TclMap.

Proper testing of TclMap or its variants will require further development. Small amounts of testing are useful throughout, but the launch of a useful test platform is awaiting a number of specific points. These minimum requirements are proposed to achieve useful testing:

- It must be stable and bug-free in executing all basic concept mapping operations.
- It should have a working file management system that includes background saving features.
- It needs to be packaged into a single installer for easy set-up by any level of user.
- The help and integrated documentation must be current to the version of the tool.
- It must have the capacity to manage the amount of work generated in a semester-long undergraduate design project.

The last point enables TclMap to be used by students in design activities during design course work. In practice, this means handling a few hundred nodes and their associated maps as well as some hyper-linking capability for easy navigation of websites and relevant files. This will enable the collection of a large feedback sample.

Computer-aided visualization alone is not sufficient. Creation of a useful system for augmenting design thinking will require the inclusion of many existing tools that are inextricably rooted in paper, text, physical prototypes, and human-to-human interaction. Successful design companies such as IDEO, for example, stress the need for physical prototypes to do good design (ABC, 1999). Recognizing these limits and interfacing with these tools is critical because forcing the use of lesser methods is a failure in the technology. It is also an interesting side note that augmented thinking was the original goal of pioneers such as Vannevar Bush, Douglas Engelbart, and Theodor Nelson when they envisioned what has now become *information technology* (Dillon, 2000). While existing technology has been invaluable in enhancing data management capabilities of individual workers, it has also left users awash in "information overload." The final components of this "Navigator" will hopefully clear up much of that overload and enable that next step in development, towards augmented understanding in design.

Design Schematics

DS is a diagramming *approach* for visualising early design information, and not a tool in and of itself. DS-oriented tools are still in development. Therefore, a quantitative evaluation of DS is not yet possible. Nonetheless, the Salustri has used structured diagrams built with existing diagramming software (like Smartdraw and CmapTools) in various teaching and research settings. In a senior undergraduate mechanical systems design course taught by Salustri, students use a PDS diagram to capture and reason about design problems. Students uniformly found the exercise "heavy" because they had had no previous exposure to concept maps or similar tools. However, they also agreed that their appreciation for the complexity, breadth, and depth of a design problem was significantly improved. Students said they "knew the problem cold," and subsequently reported experiencing far fewer problems as they designed than they did in other assignments where DS-like diagrams were not used.

Salustri has also conducted "demonstrations" with practicing engineers. In these demonstrations, audiences are shown diagrams of a conventional mechanical assembly (such as the one in Figure 13 and 14), and given 15 seconds to understand as much about it as possible; they are given no further assistance. The audience is then asked a series of questions.

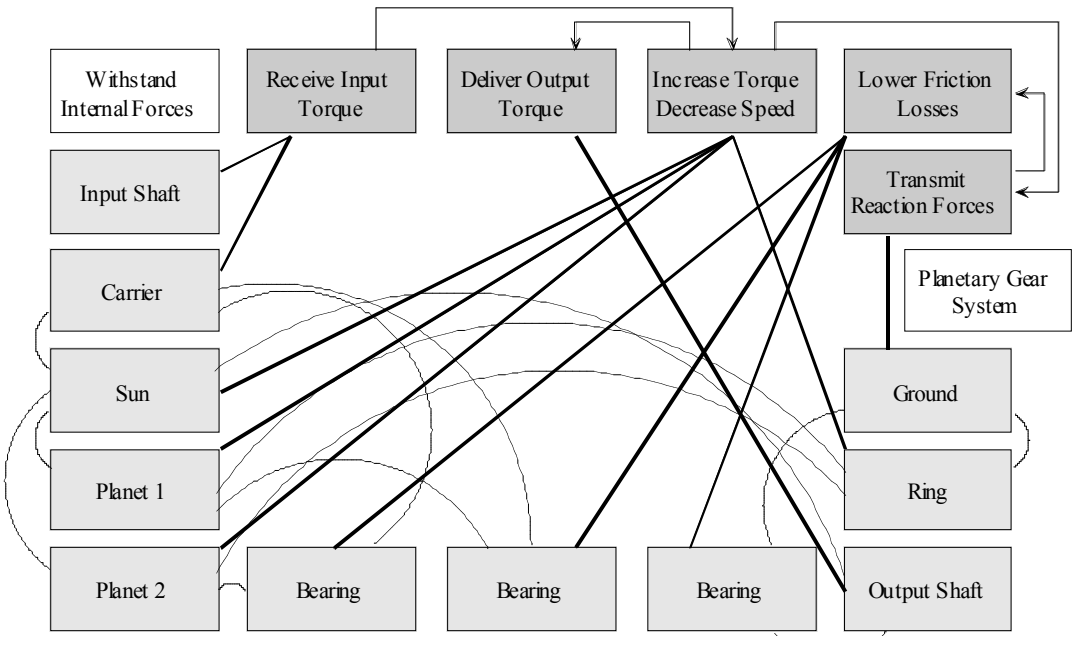


Figure 13 A diagram

of a planetary gear train created without using *design schematics* rules.

Figure 13 was created without any particular intention to organise the information for maximum communication. Figure 14 was created following guidelines Salustri devised for DS diagrams. The guidelines include using: colour (shades of grey here) to distinguish components (in white) from functional elements (in greys); overlap to indicate (a) relationships between functions and (b) which components are responsible for providing those functions; geometric layout to draw similarities to the physical configuration of the assembly; and distinct colours of rectilinear links between nodes to indicate flows of energy (in white) and physical connectivity (in black).

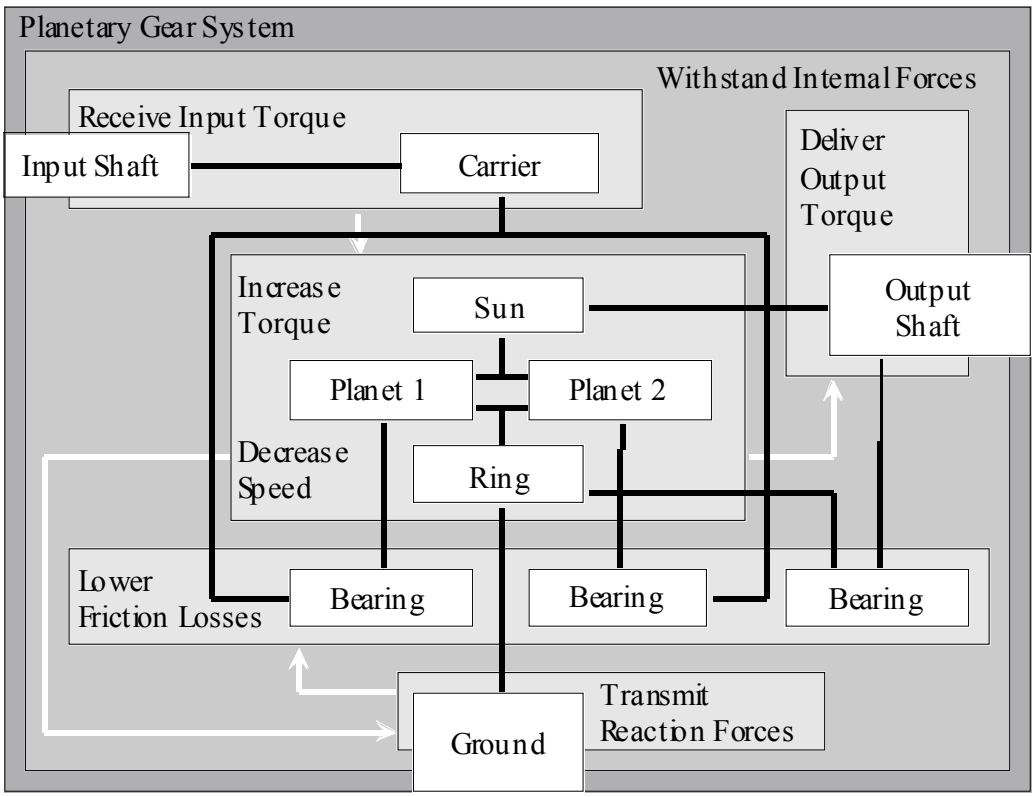


Figure 14 A "proper" design schematic of a planetary gear train.

Almost invariably, audiences perform better when answering questions about Figure 14 than Figure 13. Responses to Figure 13 are laboured and hesitant, slower in coming, and more clumsily articulated; respondents also made frequent reference to the graphical entities in the diagram itself. On the other hand, responses to Figure 14 were simple, accurate, and quick. Respondents *never* referred to the graphics when answering questions about Figure 14. This suggests that the diagram "vanishes" in the minds of the participants and is replaced by a mental model of the artefact itself. This is exactly what we would like in a good diagram: that the diagrams become transparent to the user.

Another instance of the importance of layout involves Figure 4, discussed in a previous section. Only the final layout is shown there. The original layout, constructed during the kick-off meeting of the Auto21 project described previously, confounded the participants. They were unable to recognise their contributions or what kind of information the figure was meant to communicate. However, once laid out as shown in this article, all the participants were readily able to recognise their own contributions and they agreed that the figure represented fairly the outcome of the kick-off meeting.

Though such demonstrations are unscientific, they do provide consistent anecdotal evidence to suggest that DS can be beneficial in design environments. We are currently planning a series of experiments to attempt to gauge the relative merits of DS compared to other approaches. Because it can be onerous to construct DS diagrams "by hand" (using Visio or similar products), we are first creating a simple diagramming tool, written in Tcl/Tk, which will be used as a platform. This tool was also discussed above as part of the OPLM Navigator project. Different versions of the tool will be created, implementing different diagramming features. Participant teams will then use different tools to collaborate on identical problems. We will observe these activities and analyse the results to determine which features are best suited to most people.

Conclusions

This paper has presented some of the authors' recent work with diagrams as a use of information capture and management for early designing. Simple tools, carefully designed and implemented to facilitate engineering activities can offer a viable alternative to large and complex software systems. Industry is willing to accept such tools if they are easy to learn and provide "added value" to product development processes. The potential for innovation lies in the capacity of a diagram to cause users to think differently about a design problem or solution. Diagrammatic tools, as envisioned by the authors, do not automate design activities, but augment the inherent cognitive abilities of the users. While much work remains to be done, we are optimistic that substantial benefits will result.

Acknowledgements

The authors acknowledge the support of: Rolls-Royce and BAE SYSTEMS through the UTP for Design; the UK EPSRC through various research grants; the National Sciences and Engineering Research Council of Canada; the Ontario Centres of Excellence; and the "Automobile of the 21st Century" Network of Centres of Excellence (Canada).

References

ABC Nightline. (1999) "The Deep Dive". Aired July 13, 1999. VHS Video.

Bracewell, R.H., Ahmed, S. and Wallace, K.M. (2004) DRed and design folders: a way of caputuring, storing and passing on - knowledge generated during design projects. *Proc. ASME Design Automation Conference*, Salt Lake City, USA.

Bracewell, R.H., Sharpe, J.E.E., (1996) "Functional descriptions used in computer support for qualitative scheme generation - "Schemebuilder"", AI EDAM Journal - Special Issue: Representing Functionality in Design, v10, n4, pp. 333-346.

Bracewell, R.H., Wallace, K.M. (2003) "A Tool for Capturing Design Rationale", in: Folkeson, A., Gralen, K., Norell, M. and Sellgren, U., (eds.) Proceedings of 14th International Conference on Engineering Design, Stockholm

Conklin, J., Begeman, M.L., (1988) "gIBIS: A hypertext tool for exploratory policy discussion", ACM Transactions Office Information Systems, Vol. 4, pp.303-331.

Conklin, J., Selvin, A., Buckingham Shum, S. and Sierhuis, M., (2001) "Facilitated Hypertext for Collective Sensemaking: 15 Years on from gIBIS", Proceedings 12th ACM Conference on Hypertext and Hypermedia, Århus, Denmark

Davenport, T. H. and L. Prusak. (1998) *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press, Boston.

Dillon, A. (2000) Spatial-Semantics: How Users Derive Shape from Information Space. *Journal of the American Society for Information Science*, 51(6):521-528.

Dillon, A. and Vaughan, M. (1997) 'It's the journey and the destination': Shape and the emergent property of genre in evaluating digital documents. *New Review of Multimedia and Hypermedia*, 3:91-106.

Dreiling, A., Klaus, H., Rosemann, M., and Wyssusek, B. (2005) Open Source Enterprise Systems: Towards a Viable Alternative. *Proc 38th Intl Conf on Systems Sciences*, IEEE.

Dutoit, A., Levy, S, Cunningham, D, Patrick, R, (1996) "The Basic Object System: Supporting a Spectrum From Prototypes to Hardened Code", Proceedings OOPSLA 96, v31, n10, pp104-121.

Eng, N., and Salustri, F.A. (2006) "Rugplot visualisation for preliminary design," Proc 2006 Intl Conf of the Canadian Design Engineering Network, Toronto. (CD-ROM)

Himsolt, M., (2000) "Graphlet: Design and Implementation of a Graph Editor", Software - Practice and Experience, Vol. 30, No. 11, pp.1303 – 1324.

IBM. (2005) IBM Deep Computing Visualization Solutions for Product Lifecycle Management. *Report DCS00970-USEN-00*.

Norman, D.A. (2002) *The Design of Everyday Things.* Basic Books, New York.

Novak, Joseph D. & Alberto J. Cañas "The Theory Underlying Concept Maps and How to Construct Them". Technical Report IHMC CmapTools 2006-01 Available: http://cmap.ihmc.us/Publications/ResearchPapers/TheoryCmaps/TheoryUnderlyingConceptMaps.htm. [Accessed: July 17, 2006].

Novak J.D. and Gowin R. (1984) *Learning how to learn*. Cambridge University Press, Cambridge.

Raskin, J. (2000) *The Humane Interface: New Directions for Designing Interactive Systems.* Addison-Wesley, Toronto.

Salustri, F.A. (2005) Using pattern languages in design engineering. *ICED 2005 Proceedings (CD-ROM)*, Melbourne, Australia.

Salustri, F.A. and Parmar, J. (2004) Product design schematics: structured diagramming for requirements engineering. *Proc 8th Intl Design Conf*, Dubrovnik, pages 1453-1460.

Stone, R.B., Wood, K.L., and Crawford, R.H. (2000) A heuristic method for identifying modules to develop product architectures. *Design Studies*, 21(1):5-31.

Tufte, E.R. (1990) *Envisioning Information.* Graphics Press, Cheshire, Connecticut.

Tufte, E.R. (2001) *The Visual Display of Quantitative Information.* 2nd ed. Graphics Press, Cheshire, Connecticut.

Tufte, E.R. (1997) Visual Explanations: Images, Quantities, Evidence and Narrative. Graphics Press, Cheshire, Connecticut.