

VISUALISING EARLY PRODUCT DEVELOPMENT INFORMATION

Filippo A. Salustri and Jayesh Parmar

Abstract

The authors believe that an appropriate diagramming tool can be of substantial benefit to designers, especially in the early, pre-geometry stages of product development. We find no such tool to exist. We therefore introduce *design schematics* (DS) as such a tool. We outline the general benefits of diagramming and then consider the advantages and disadvantages of some existing diagramming methods. Our analysis of this motivates the development of DS. Several examples demonstrate how DSs can capture important information during early design stages. We are currently developing a computational tool that implements DS and discuss some of the issues we face in this regard. While there is not yet any quantitative data by which DS can be evaluated, there is anecdotal evidence suggesting that the tool has potential to be of benefit to practising designers.

Keywords: visualisation, product model, concept map, non-geometric information

1 Introduction

The early stages of product development are the most crucial. The Pareto Principle tells us that 80% of the performance of a product with respect to cost, quality, function, usability and ergonomics, environmental friendliness, etc. is set by the decisions taken in the first 20% of the product's development process. Products are also becoming more complex in response to user sophistication, new technologies, and government regulations. This places a further burden on the product developer.

Globalisation further complicates matters by broadening the potential marketplace, the field of competitors for that market, and the organisational, geographical, and cultural characteristics of the product development teams themselves. Design information must be made relevant to all aspects of the product development process for concurrent engineering and related practices to yield their full benefits. Teams must share information with colleagues, suppliers, and managers who may be physically distant, so it is essential that design information be presented in a clear, concise way. There are cultural issues too: in the "global village", design information written in one natural language can be misinterpreted by recipients not very familiar with it. Lessening the dependence on natural language will aid in the internationalisation of design information and thus benefit culturally diverse design teams.

One way of dealing with both the importance of early product development and its increased complexity is to find strategies to simplify the methods and tools used by product development teams. However, it is exactly in the crucial early stages of product development (before product geometry has been established) that the least amount of work has been done to find clean, efficient methods to represent and reason about products and product models. In the end, what en-

engineering enterprises really need is for product development teams to think clearly and creatively, and to think about the right things in the right way, as they develop products. In order to facilitate rapid development of innovative products, flexibility and low process “weight” are essential. A “light-weight” process is one that lessens the burden on the development team of documentation, administration, and process structure to just what is absolutely necessary. This frees the team’s time and resources to focus on the product itself. In other words, the “right” product-modelling tool for the early design stages will *facilitate* human cognitive abilities rather than *displace* that burden from the human designer to the tool itself.

A *graphical*, rather than textual, notation can succeed here [1]. It is cliché to say that a picture is worth a thousand words, because it is true. In design meetings, designers typically use diagrams to aid the team’s work. These diagrams function as cognitive cues to the designers about important issues, decisions, and actions. Furthermore, appropriately *laid out* diagrams can communicate a holistic perspective of information that is very difficult to extract from purely textual descriptions, thus giving designers a better overview of their work [2]. Diagrams can also cross borders of culture and language as well as discipline and sector, providing a possible means to integrate the design stages across all stakeholder areas.

Many graphical and diagrammatic techniques exist, including bond graphs, IDEF, entity-relationship diagrams, the Modelica language for physical systems modelling, the various kinds of block and flow diagrams, and concept maps. These tools can serve in some stages of product development. However, these tools are best suited and most readily applied to specifying designs once the overall conceptual and systems design stages have been completed.

In summary, the authors believe that it is possible to design and implement a simple tool that will be small, usable, and effective for at least some of the upstream stages of design processes by giving up the requirement that all information must be carefully structured. The authors’ approach, called *Design Schematics* (DS), includes four components that cover the key stages of early product development. The components are *usage scenarios*, *product design schematics*, *function schematics*, and *product architecture schematics*. The approach is intended to be the foundation of a new kind of computer tool specifically for product developers and designers. This paper will focus on the language of DS rather than on implementation issues.

2 Fundamentals

The authors are studying a variety of existing diagramming tools and methods. To the authors, the main disadvantage of all of these methods except *concept maps* is that they impose too many constraints on early stage design for the sake of providing well-defined structures to specify product information unambiguously and prepare it for various analyses. The chief hindrance of concept maps, on the other hand, is their lack of structure, which prevents designers from achieving the concentrated focus needed to treat design issues effectively and efficiently. Design Schematics is being developed to lie somewhere between concept maps on the one hand, and the more structured tools on the other. We are studying each existent tool to identify features that can be combined with concept maps to guide designers while preserving the flexibility needed in the early stages of designing.

2.1 Requirements Analysis for Early Design Stages

The authors consider the *early design stages* to include strategic planning, requirements analysis and specification, conceptual design, systems design, and configuration and layout. A representation for this kind of information must meet the following requirements.

1. Promote thinking and learning about design problems rather than just specifying solutions.
2. Be extremely intuitive and easy to learn.
3. Be flexible enough to represent different kinds of information consistently.
4. Provide structure to disambiguate kinds of information arising in the early design stages.

2.2 Designing as Learning

The authors subscribe to the notion that learning is the integration of new information into a learning agent's existent *cognitive structure* [3], which can be seen as a richly interconnected network of information upon which cognitive processes operate. The more richly interconnected new information is, the greater the impact of the new information on cognitive processes of a learning agent, and the better the new information has been learned.

We believe 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 answer (the design) is usually quite apparent. Indeed, it is a common perspective that design problems and their solutions evolve concurrently. In this view, then, effective design tools promote the integration of new information by being as compatible as possible with the network-based model of learning described here.

2.3 Concept maps as learning tools

Concept maps are visual learning tools meant to target specifically the mode of learning noted above. Concepts are shown as nodes and relations as labelled arcs connecting the nodes. Concept maps that take the form of *network graphs* have been correlated to both deep learning and innovative thinking [4].

Concept maps have been classically used in early childhood education as both learning and assessment tools. In business, they are often used elsewhere during brainstorming to capture quickly and easily ideas that are not precisely defined or understood. By using such an informal technique, the emphasis remains on the brainstorming itself (i.e. thinking) and not on the mechanistic aspects of specifying its results.

However, our experiments with concept maps for product representation has shown that they are just too simple to represent the required scope. Improvements include the following.

Layout management. We have found that the layout of nodes in a concept map can communicate substantive information, and that the best layout for a particular diagram depends on the kind of information in the map.

Hierarchies. It must be possible to collapse one map into a single node within a larger map; this would allow direct representation of hierarchical levels of information.

Hypergraph support. Generally, concept maps for design will be hypergraphs (e.g. supporting one-to-many links between nodes). This is rarely provided in existing systems.

Extended link labelling. Conventional concept maps allow labels only on nodes and mid-way on links (i.e. named links). However, we have found that also allowing labels on both the heads and tails of links substantially increases the flexibility and expressive richness.

Filtering. All-inclusive maps can quickly become too cluttered to be understood easily. Since designers tend to focus on specific aspects of designs sequentially, it must be possible to temporarily filter out unneeded information. The capacity to filter out various portions of a map would help maintain the integrity of the map as a whole while presenting only the minimum required information.

Links between links. The relations represented by links can themselves be thought of as concepts; that is, a whole concept map could be developed to explain a relation like *welded to*. Such maps would be obviously pertinent to manufacturing engineers and engineering analysts.

Boolean and cardinality constraints. Conventional concept maps do not support boolean relationships between nodes or links, nor do they support cardinality (e.g. that an assembly requires *four* of one kind of part).

3 Design schematics

This section gives an introduction to design schematics through a set of examples. The DSs included in this paper were drawn “by-hand” using SmartDraw. A case will be made later in this paper regarding the development of computer tools that implement DS. Our main interest here is to discuss the diagrammatic language of DS and to indicate its representational strength for early product modelling.

As a sample problem, we use the design of a powered screwdriver (taken from [5]).

To begin, we consider the description of a *usage scenario* (US), which allows a designer to think about how a product might be used. Information to build a US can come from focus groups, consumer reports, marketing information, designers, etc. USs help designers focus on customer needs, how a product will be used, and issues that arise as a result (e.g. safety). A DS for the powered screwdriver might be as shown in Figure 1, which shows three USs. The arcs in USs show the order of actions taken by users. The terminal node at the right end of this kind of DS has an implied feedback link connecting to the initial (leftmost) node. The number *1* at the branch in the *Usage* US indicates that only one of the branches can be followed. The links are unlabelled because there is only one kind of relationship in a US; the links indicate the order of steps in a product’s use. “PSD” stands for “powered screwdriver”.

Designers can then add nodes indicating product characteristics (PCs – things a product must *be*), functional requirements (FRs – things a product must *do*), and constraints that follow from reasoning about the US’s design implications. Figure 2 shows the *Setup* usage scenario, with some of these other entities. The branches in Figure 2 are not labelled because the default meaning of a branch is that all the entities at the heads of those links must be attained for the entity at the root of the link to be attained. Blue nodes are PCs, green nodes are FRs, and yellow nodes are constraints.

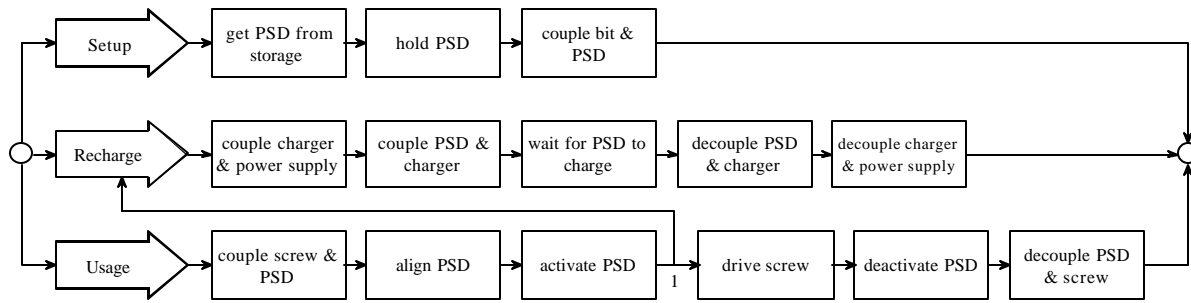


Figure 1. A DS with three usage scenarios for a powered screwdriver.

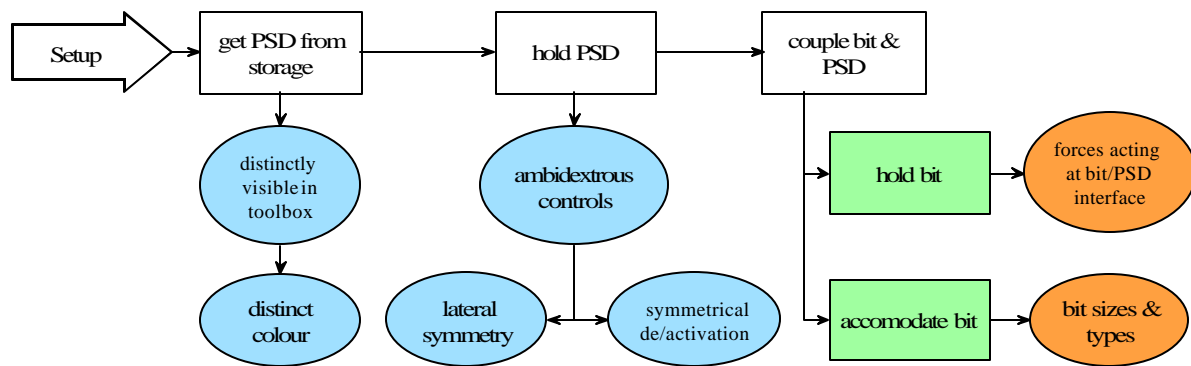


Figure 2. A US augmented with PCs, FRs and constraints.

One can extract all the PCs, FRs, and constraints from a US and, by rearranging them, form a *product design schematic* (PDS) – a DS of the basic characteristics and functions expected of a suitable design. Our PDS representation combines aspects of Pugh’s [6] and Dym’s [7] work. The details of our representation are not important here and will be presented in future paper. Our goal is to show how a PDS diagram can represent important design information. A *partial* PDS of the screwdriver is shown in Figure 3, for the *Usage* US in Figure 1. The links are unlabelled because the node colours imply the relationships. Bold nodes are of high importance, nodes without borders are of low importance, and the others are of middling importance. The left-most segment of PCs defines some of the basic characteristics of any product. The next segment shows some PCs and FRs specific to the problem. The number of segments varies with the complexity of the problem. General PCs are linked to more specific PCs, which are linked to the FRs that define how a product achieves them. FRs in turn link to constraints that limit a product’s performance with respect to the FR. PDSs that form hierarchies are consistent with the Independence Axiom of Suh’s Axiomatic Design [8]. However, such uncoupled designs are rare. The highlighted links (in red) in Figure 3 indicate interactions between PCs and FRs, which appear as *cross-links* in a PDS and which indicate graphically violations of the Independence Axiom. Suh’s design parameters have not yet been incorporated into our PDSs; this is a point of future work.

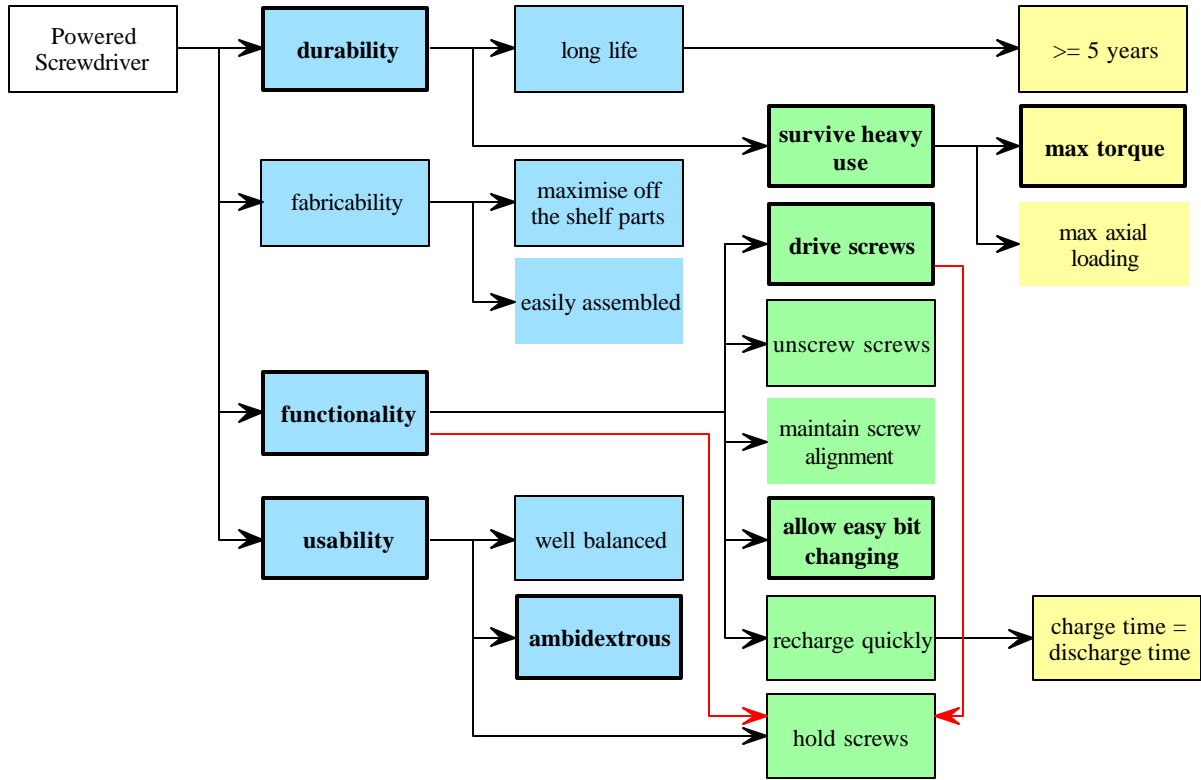


Figure 3. A partial PDS for the powered screwdriver.

One may then develop a *function schematic* (FS) of the problem, which is a DS that captures the result of function decomposition. It is similar to the function diagrams of Wood et al [5], and is used to expand the FRs into a function model of the product. A fragment of a FS for the powered screwdriver is shown in Figure 4.

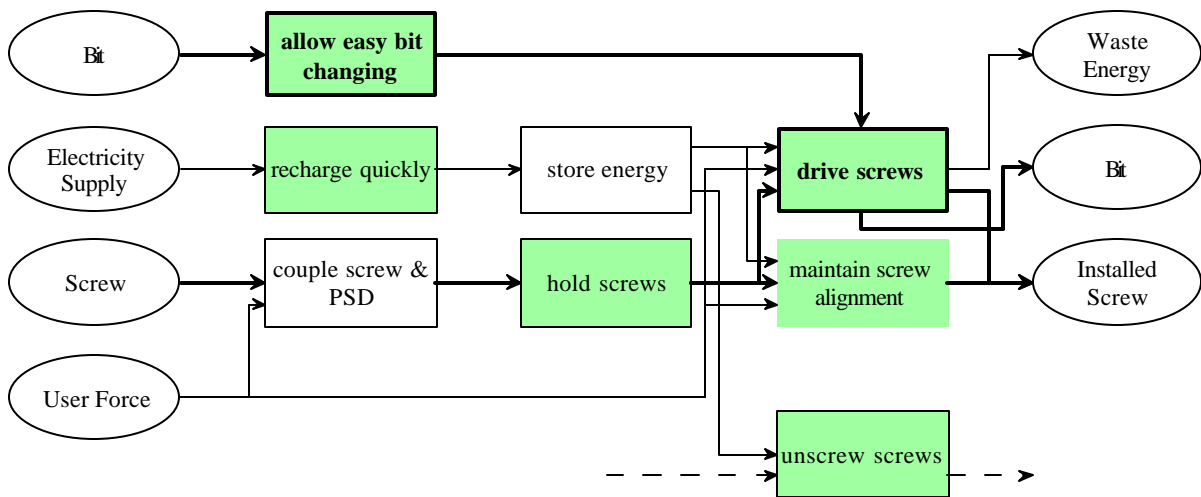


Figure 4. A partial FS for the powered screwdriver.

To save space, only some functions from the PDS are shown in Figure 4. Our intention is to demonstrate the DS language and not to provide a complete account of the screwdriver problem. In actual cases it will be important to indicate where and how a DS is *incomplete*. One way is shown in Figure 4: the dashed arrows on the function *unscrew screws* shows that there is more to this FS than is shown. The green nodes are *identical copies* of their counterparts in the PDS. Also, the sides of function nodes have specific purposes. Borrowing from IDEF, inputs into the left side are resources consumed by the function and inputs into the top of a node indicate non-consumed resources (e.g. the bit of the screwdriver).

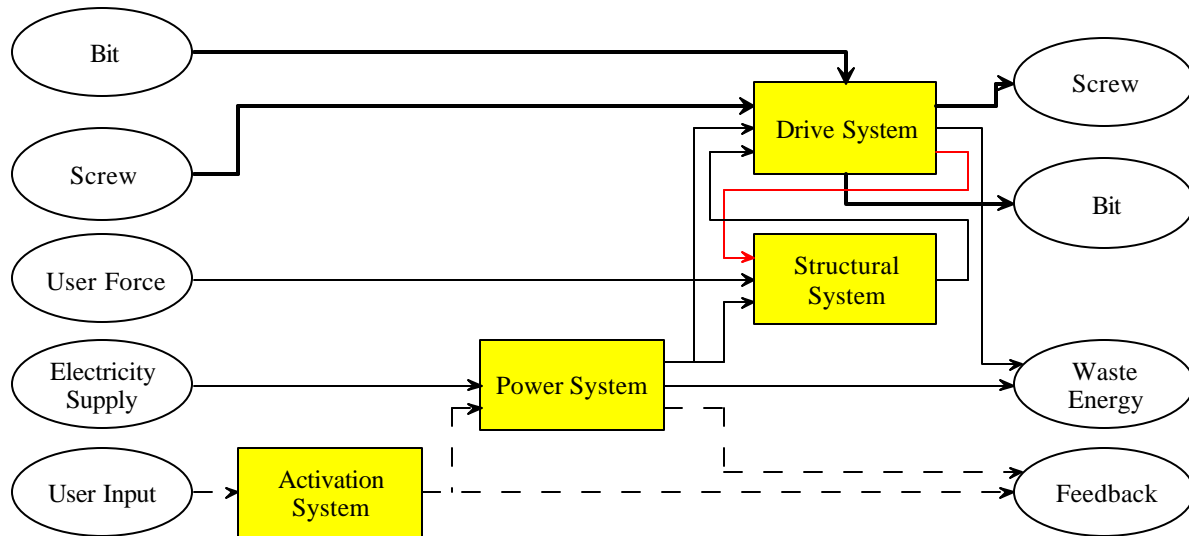


Figure 5. A possible PAS for the powered screwdriver based on the FS in Figure 4.

Finally, one can use the FS to develop a product architecture schematic (PAS). Based somewhat on the diagrams by the same name in [9], our PASs are more structured. Again, the construction rules of the PAS are not as important here as the notion that they represent important design information that can be represented diagrammatically within the Design Schematic framework. A PAS for the powered screwdriver is shown in Figure 5. Note the feedback between the *structural* and *drive* systems (highlighted in red for clarity) is clearly evident by a simple visual inspection of the PAS. This is due to the layout of the diagram, which is designed to highlight such patterns of interaction in the diagram.

4 Computerisation of Design Schematics

Clearly, computer support for the development and manipulation of DSs is important, particularly with respect to layout of the schematics. In this section, the authors will discuss how we are approaching the development of such software. There are three fundamental purposes for the computerisation of DSs.

Computerisation improves usability. The DSs presented in this paper were laid out “by hand” with conventional diagramming software. The task of laying out the nodes and links was oner-

ous because the layout itself communicates important design information. Implementing a computer tool to lay out DSs automatically would solve this problem.

Computerisation improves information integrity. Nodes and links will be represented independent of the style of DS, so that they can appear consistently in multiple DSs and thus communicate information more effectively to designers.

Computerisation increases the potential for design automation. Since a DS carries a certain amount of semantic content, that content can be examined and manipulated by artificial intelligence engines to conduct various design analyses.

The basis of a computerised DS (cDS) system is straightforward and well understood: there are many graph layout engines and software packages available both commercially and for research purposes. The distinction for cDS systems is in the “intelligence” needed to support design-oriented semantic content, particularly for layout purposes. This notwithstanding, we expect a DS software package to be relatively lightweight in terms of disk and RAM footprints, CPU speeds for acceptable performance, etc.

To address this, the authors are developing an *extensible intelligent graph layout engine*. Within this engine, different *modes* will be available as run-time modules. A mode will have a layout algorithm and various standards for node and link appearances. Each kind of DS will have its own mode. Modes will be able to filter nodes and links so as to present only information pertinent to that mode. It will be possible to implement new modes and simply load them in the engine as “plug-ins”.

For example, a *usage scenario* mode will implement a layout algorithm that will ensure (a) the user action nodes will appear organised in the style of Figure 2, and that other nodes will be arranged as well as possible without altering the action node layout. It will provide users with a defined set of nodes and links that can be used in that mode. The user will then be able to switch between USs and, say, PDSs while maintaining integrity of the underlying structure of nodes and links. Changes to one DS will transfer into others as defined by the specific filters and layout algorithms for the corresponding modes.

5 Initial Assessment of Design Schematics

There are no diagramming tools specifically for product development and design. As such, DS is a unique tool and no direct analytic comparison to other diagrammatic methods is possible. The authors have, however, had some experiences using the DS framework that can be informative with regards to assessing the framework.

The authors have used structured diagrams built with existing diagramming software in various teaching and research settings. In a senior undergraduate mechanical system 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. 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 diagrams were not used.

In research settings, the authors have found that DSs help clarify thinking, document activities, and collaborate in teams. We have used DSs and DS-like diagrams to study automotive drivetrains, small appliances and tools, and small housing structures, as well as to manage research projects. (These experiences will be the subject of a future paper.)

6 Conclusions

The authors have introduced a new tool to assist designers, *design schematics*, which is based on concept maps, augmented with certain features from other diagramming systems. The focus is on the nature of the representation and not on the admittedly significant computational aspects. Several examples are given to indicate how DSs can capture meaningfully information during the early stages of product design, when (a) little or no information about product geometry is available and (b) decisions have the greatest impact on product performance. Since DS is still under development, there is no quantitative data by which to evaluate the tool. However, some anecdotal evidence gathered by the authors suggests that the tool does have potential to be of benefit to practising designers.

Acknowledgements

The author acknowledges the support of the National Sciences and Engineering Research Council of Canada for funding the work reported herein.

References

- [1] Kulpa, Z. "Diagrammatic representation and reasoning", Machine Graphics Vol. 3, 1994, pp. 77-103.
- [2] Marshall M.S., Herman I., and Melançon G., "An object-oriented design for graph visualisation", Software Practice and Experience, Vol. 31, 2001, pp.739-756.
- [3] Novak J.D. and Gowin R., "Learning how to learn", Cambridge University Press, Cambridge, 1984.
- [4] Novak J.D., "Learning, creating, and using knowledge: concept maps as facilitative tools in schools and corporations", Lawrence Erlbaum Associates, New Jersey, 1998.
- [5] Stone R.B., Wood K.L., and Crawford R.H., "A heuristic method for identifying modules for product architectures", Design Studies, Vol. 21, 2000, pp.5-31.
- [6] Pugh S., "Total design", Addison-Wesley, London, 1991.
- [7] Dym C.L. and Little P., "Engineering design: a project-based approach", Wiley & Sons, New York, 2000.
- [8] Suh N.P., "Principles of Design", Oxford University Press, 1990.
- [9] Ulrich K.T. and Eppinger S.D., "Product design and development", McGraw-Hill, New York, 1995.
- [10] Steward D.V., "The design structure system: a method for managing the design of complex systems", IEEE Transactions on Engineering Management, Vol. 28, 1981, pp.71-74.