Computer-Aided Design and Applications

http://www.cadanda.com

Visualizing Information in the Early Stages of Engineering Design

Filippo A. Salustri[1], Nathan L. Eng[2] and Janaka S. Weerasinghe[3]

[1]Ryerson University, salustri@ryerson.ca
[2]University of Cambridge, nle20@cam.ac.uk
[3]Ryerson University, jweesari@ryerson.ca

ABSTRACT

The early stages of engineering design are the most crucial for successful product development, yet they are not well supported with computer tools compared to other, downstream stages. This paper will discuss recent efforts by the authors to create visual representations of, and tools for, the non-geometric, qualitative information typical in the early stages of engineering design. It appears evident that there is tremendous opportunity to improve the capacity of designers to think both critically and creatively through diagramming in early design, but the field is still embryonic and much work remains to be done.

Keywords: visualization, qualitative information, early design.
DOI: 10.3722/cadaps.2008.697-714

1. INTRODUCTION

The authors have observed that there is relatively little use of diagrammatic visualization of qualitative information in the early stages of designing. We define early design as the stages of designing preceding embodiment – i.e. before a design is given *shape*. This generally includes the tasks of problem analysis (including "requirements engineering"), ideation and concept design, and systems design.

In engineering design, text documents are the norm in most industries, and form is established downstream, usually using conventional CAD software. In product design, industrial design, and architecture, heavily annotated sketches and "doodles" are often used to capture the spirit or essence of a design. These may or may not be computerized. Still, across these disciplines, diagrams seem rarely used as "official" representations of early design information. This is peculiar, given the tendency of designers to be "visual thinkers." Yet, in terms of overall value, diagrams are rich artifacts [4] around which one can also bring together stakeholders, which is key for effective, sustainable design (e.g. the House of Quality [11]).

The authors have therefore started to investigate this situation, and to develop diagrammatic methods and tools that will hopefully be beneficial to practicing designers. This work proceeds on various fronts, and this paper will summarize the work done and left to do. The rest of this paper is laid out according to these research fronts.

There are two principles underlying our work that are worth describing at the outset.

Principle #1: Simplicity is Power. This applies to both functionality and usability. Tools that are simple to use are more easily learned and adopted. They are also more robust. Simple tools can be more effective because they are more likely to fit easily ("out of the box," as it were) into the broad range of messy, dynamic practices in the real world. Simple tools are also easier to make extensible, and such extensions tend to be less dependent on one another, further simplifying the system.

Principle #2: Diagrams Augment Cognition. While we do not believe that diagrams are a "silver bullet," we do believe that they represent an alternative representation of information that provides different and valuable perspectives on the object of study. As such, a good diagram augments the capacity of the diagram's user to achieve goals. Visualization literally "makes visible" (or "evident") things that might not otherwise be so. This can promote new ideas, and lead to deeper insights. We do not believe that diagrams can or should replace human designerly cognition. As such, we see the user and the diagram existing in a kind of symbiosis, or as elements of a system, able to do together what they might not otherwise achieve. Thus, our work is firmly rooted in "human-centricity."

2. QUALITATIVE DIAGRAMMING TOOLS IN EARLY DESIGN

A *diagram* is a model of a thing composed of graphical and textual elements. Since it is a model (i.e. an incomplete or imperfect representation of a thing), its form will depend on the agent that creates it and must suit the needs of those agents. Thus, we can expect many different possible diagrams for a single thing. It is important that diagramming tools exhibit sufficient flexibility to adapt to the needs and capabilities of its users.

Qualitative information (as opposed to *quantitative* information) regards the distinctive attributes and characteristics of things. As such, we cannot depend on mathematical representations to be sufficient. This excludes a wide assortment of existent tools – most notably CAD packages. Indeed, we distinguish our area of interest as that expressly *not* served by conventional CAD software; by the time a designer understands enough about a design to be able to specify geometry as required by CAD software, most of the critical design decisions about the item have already been made. Our interest is in helping designers during those early stages of designing where quantitative information about the product being designed is just not available.

Our work began with a review of existent diagramming tools, which we briefly summarize here.

Generic diagramming software, such as Visio, SmartDraw, and OmniGraffle, are able to represent a wide variety of diagramming styles and layouts. However, unless one is willing to invest significant time to understand the software and use extensive but sometimes arcane "macro" facilities, it is difficult to create layouts for specific types of diagrams. We do recognize that many packages support, for example, UML diagram types, but we find UML to be very difficult to understand in other ways (described below, with respect to its variant for systems engineering, SysML).

Where we find these tools to be useful is in prototyping layout schemes to show users and get feedback. In principle, once good layouts are discovered, they can be "coded" into a diagramming platform that is specifically targeted at the engineering design community. The envisioned diagramming platform will be discussed in Section 5.

We also reviewed the two major classes of general diagramming tools intended to assist in thinking and problem-solving: concept maps and mind maps. Concept maps are network hypergraphs of concepts (nodes) connected by relations (links) [16]. Originally intended as learning tools for children, concept maps are now regularly used in research and business settings. Concept mapping software packages, such as CmapTools (http://cmap.ihmc.us/) and Inspiration (http://www.inspiration.com/), are very simple, easily learned, and extremely flexible. While they do not have complicated algorithms to "mine" or otherwise analyze information, they excel in helping the human user to think. We like to view such tools as *augmenting* human thinking power, rather than replacing it; the system including a human and a concept map is able to reason better than the human is alone.

Mind maps represent groups of concepts built around a hierarchy. They do not have labeled arcs, so they rely on the meanings of adjacent nodes as a source of inference of their relationships. These simplifications make them suitable for use in activities such as brainstorming, where quick "branching" of ideas and some ambiguity (to free interpretation of relationships) is critical.

The reviewed concept map and mind map tools are not sufficient for our purposes because they do not facilitate the structured layout that we have found is essential in engineering design visualization. However, their simplicity and sympathy to typical thinking processes have motivated us to study them more closely (see below).

The authors also reviewed libraries of graph theoretic algorithms, such as JGraph (http://www.jgraph.com/) and AiSee (http://www.aisee.com/). While all the standard graph theoretic layouts are supported in such packages, the kinds of layouts needed to represent information typically used in early engineering design are not available. (The kinds of

layouts we have found useful will be discussed in Section 5.) Therefore, such packages are of limited use in this particular application area.

We also looked at several diagramming methods, including bond graphs [13], entity-relationship diagrams [2], IDEF [1], SysML [9], Compendium (http://www.compendiuminstitute.org/), and DRed [5]. In each case, we considered "typical" diagrams and analyzed them in terms of clarity (how obvious is the information it represents), density (how much information can be represented), breadth (what different kinds of information can be represented), and scope (what stages of early designing it can cover). Below is a very brief summary of our findings.

Bond graphs, IDEF, Compendium, and DRed are unable to represent the breadth of information we want to achieve. For instance, bond graphs and IDEF cannot capture requirements, and the IBIS-based tools cannot represent system architectures. Entity-relationship diagrams, SysML, Compendium, and DRed have no support for controlling/specifying layout, except according to generic graph theoretic algorithms. Bond graphs and IDEF do have some layout capabilities, but these are too rigid to represent the kinds of information we find in early design (see Section 5); this results in diagrams that appear "cluttered" and are hard to understand as a result. Finally, bond graphs, entity-relationship diagrams, and SysML have syntaxes that are very complicated and difficult to learn and understand; SysML, like UML generally, also includes a wide variety of very different types of diagrams, which further hinders quick comprehension by users.

We have concluded from our analysis that:

- Simplicity is important. The simpler the tool – even though its scope may be limited as a result – the easier it is to use, and the more likely users are to adopt it willingly and "naturally."
- Network hypergraphs are essential. The richly interrelated information elements typical in early designing are highly coupled, and representing those relationships is essential.
- Diagram layout is essential. A proper layout for a diagram can actually simplify it without loss of semantics. Yet none of the reviewed tools allow configurable layout. An example of the importance of layout is given in Section 5.
- Language density must be balanced against usability. SysML is a very dense diagrammatic medium, but it's complex structure and variety of graphical entities makes is very difficult to understand and use. Concept maps, on the other hand, are extremely easy to use largely because of their "bare bones" syntax; but they are also limited in their expressivity. A balance (a "global optimum") must be found between these two extremes.

We find, then, that there is no existent tool suitable to our purposes. Therefore, a new framework for diagramming tools for engineering design must be developed. To maintain the simplicity of a final solution, we begin with the simplest and most flexible tool we have found – concept maps – and will enhance it only as necessary to represent early design information, as determined by experimentation and further research.

3. EXPLORATORY CASE STUDIES

Given the review of existent tools (Section 2) it became evident to the authors that there was relatively little basic understanding of how people use and interact with diagrams during early engineering design. Not only do we lack the raw data to stimulate analysis and further research, but we also lack the opportunity to *gather* that data. It is extremely difficult to observe practicing designers in realistic settings, and to let them experiment with different tools and methods not necessarily sanctioned by their employers. Engineering corporations protect carefully their internal development processes – which they would have to expose if we were to observe practitioners *in situ*. These corporations are also very sensitive to productivity losses that can be expected when "trying out" new ways of doing things. Indeed, this problem has become so pronounced in the design research community, that there have recently been complex and expensive research projects proposed to do nothing more than capture the "raw design process data" that researchers need (e.g. [12]).

To address this lack, at least partially, the authors and other colleagues have undertaken a number of exploratory case studies dealing with the use of diagrams in engineering design settings. These case studies are all based on design activities carried out by students and other researchers. We understand there are inherent shortcomings in substituting

students or academics for practitioners, but nothing better was available to us. A brief description of the case studies follows.

3.1 Basic Diagramming Preferences

Two properties of "useful" diagrams for engineering design have become evident from simple informal tests with students and some practitioners. Because of their universality, we present them here; refer to Figure 1, below.

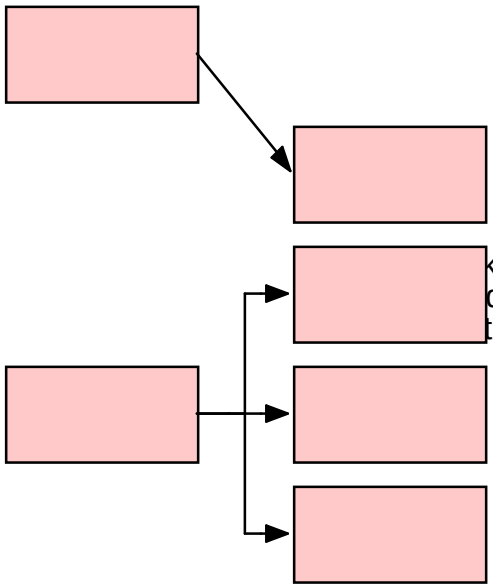

Fig 1 (a): Angled arrows and sharp corners are **not** preferred.

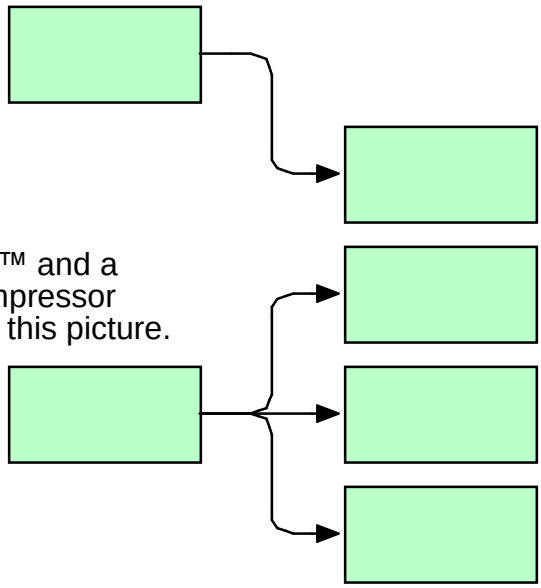

Fig 1 (b): Horizontal/vertical arrows and rounded corners **are** preferred.

Figure 1 (a) shows a diagram with angled lines and sharp corners. In Figure 1 (b), all lines are composed of horizontal and vertical elements, and the corners are rounded. Diagrams per (b) are universally preferred by engineering designers and students that we have studied. It is unclear why exactly this is, but nonetheless if we want a tool to be adopted, then it must "appeal" to its user community. In the case of rounded corners, we suspect that the preference has to do with the ability to follow particular paths where lines split. One may make an analogy to roadmaps, where the curved corners represent "on ramps" and "off ramps."

Because of the overwhelming preference shown by those we have studied so far, we adopt these two rules (rounded corners, and no angled lines) as fundamental diagrammatic terms.

3.2 Concept Maps in Engineering Design

A particular concept mapping tool has been applied generally to two research projects in engineering design. The tool, called CmapTools, is a java-based application with both a client and server element, thus allowing "cmaps" to be shared. The application also has a collaboration tool that allows simultaneous editing of a single cmap by multiple, non-collocated users. This application, developed by the Institute for Human and Machine Cognition, is true to the original idea of concept mapping as devised by Joseph Novak.

One of these projects involves the use of CmapTools in a graduate design course in Mechanical Engineering. In the winter semester of 2006, 17 graduate students from Ryerson University participated in a concept mapping exercise. This exercise was to be a practice run for a concept mapping workshop (discussed below). The goal of this experiment was to observe any patterns in style and layout that may be attributed to engineers in particular. This would provide an insight into how engineers gather their thoughts and present them through a visual medium.

For the exercise the students were given 30 minutes to create individual maps answering the question 'What is Design?' where they were free to draw upon all aspects of their personal lives to answer the question, i.e. education, culture, experience, etc. All the maps were categorized based on the types of layout features they exhibited to see if there was a predominant layout structure common to these engineering students.

The most common layout was the hierarchy, occurring in 35% of the maps, followed by linear, web graph, and scattered branch patterns, and the least used structure was of cyclical and network structures. One interesting aspect of the exercise was the style that the students used when creating a set of child nodes from a parent node. The largest group of maps, containing 47% of the maps drawn, had an element of linear subset structures where the subsequent nodes were aligned either horizontally or vertically. Another structure we found was a 'blooming' pattern seen in 17% of the maps, where the subsequent nodes would encircle the parent node.

Regardless of the structure that was used in their maps, we observed a general lack of interconnectivity between the nodes. Novak attributes low connectivity among nodes in a hierarchy pattern to a poor understanding for the subject being discussed [16]. One possibility for the students' narrow view on the subject may be due to limited time given to generate the map, thus they did not have the opportunity to broaden their thinking to include other aspects of design. While we were somewhat disappointed by the results – we had hoped for a richer assortment of cmaps – students generally found the exercise interesting, and several continued to use concept maps in their studies.

The second project was part of the "Automobile of the 21st Century," a Canadian Network of Centres of Excellence, which is a national research network intended to substantively advance automotive engineering in Canada. One of its many constituent projects is a multi-university endeavor to explore how engineers collaborate in the early stages of automotive engineering design. Graduate students from a number of universities across southern Ontario formed at team to design an environmentally friendly urban vehicle. To study collaboration, the project members were introduced to several collaborative tools in an attempt to see what aspects of the tool were helpful in facilitating their collaboration, of which concept maps was one of them.

Following some training in concept mapping and the CmapTools program, each member created a concept map that represented an outline of their understanding of the project. From an instructional point of view, this exercise would be important as it showed the variability of individual understanding regarding the content of the meeting. Exploration and analysis of the differences in the maps could provide insight into how things were presented and interpreted.

Initially, each team member created his own "cmap" about the project and updated it in response to discussions with other team members. This was not a good use of the tool because information in individual cmaps was duplicated or even incorrect. Initially, little attention was paid to layout as the cmaps were constructed; but, over time, the group started to notice structures that existed within the information (e.g. iterative or recursive loops in tasks, and dependencies between task elements) that were only recognized when the cmaps were rearranged into an "appropriate" layout for them. This was true even though the participants were shown how layout could improve a diagram's capacity to communicate information. This is a lesson that must apparently be learned by experience.

Furthermore, as the team members grew more accustomed to the software, the authors noticed that (a) convergence to a common understanding of the goals and structure of the project occurred very quickly, and (b) interactions between the team members increased in frequency. The authors cannot prove that using CmapTools collaboratively *caused* improved productivity and convergence in understanding, but we do believe that a relation exists, one that we intend to explore further in the future.

3.3 Concept Maps in Design Generally

We studied concept mapping in three other projects. These are described briefly below.

The first project involved the use of concept maps to assist a group of researchers looking for common research themes. In December 2006, several researchers met to explore the possibility of combining various fields of research study to offer a graduate level degree in human factors. The group agreed to use CmapTools as the means to describe the relationships between their research interests, and to find points of synergy and compatibility.

The meeting was run in a stenographer style, where one person was designated as the stenographer and had complete control of the cmap. The rest of the group instructed the stenographer, who would record information in the cmap for all to see and discuss.

The participants found many relationships and links between the different research topics in the cmap that were not apparent to them until they saw and reflected upon the cmap. Even though the meeting was only three hours long, all the participants agreed that significant progress was made. The only time the meeting slowed down in pace was when a researcher had to explain to the stenographer how their input ought to have been represented, but following this the discussion picked up again.

The meeting could have been held in a computer lab, so that each researcher could run CmapTools, and use it's internet-based collaboration service to work together on the same cmap. However, this would have required significant time to train the participants, few of whom had ever used CmapTools. In the end, it was more efficient to employ a stenographer for this activity.

The second project involved a Toronto Area hospital. In 2001, a graduate student in the Department of Mechanical and Industrial Engineering at Ryerson University presented a thesis to on the use of UML and SADT to create a business model for the radiology department of St. Michael's Hospital; over 15 diagrams were created. After studying the UML diagrams, we thought it would be interesting to compare UML and concept maps as visualization techniques.

A single cmap was constructed based on the ultrasound procedures and guidelines outlined in the other student's work. The cmap outlined the entire radiology process starting with a patient's entry into the hospital until return to the recovery ward. This map was then compared with the UML diagrams. It was found that this single concept map was able to capture almost all of the information presented in the many UML diagrams. The cmap also presented it in a more readable manner (per comments by various colleagues and representatives of the hospital) whereas the UML diagrams were hard to follow since there was no intuitive 'starting point' or flow to the diagrams. Furthermore, the UML diagrams could not capture the path taken by actors (patients, nurses, doctors, etc). On the other hand, the cmap showed clearly a path through each sub process for each actor, and it is clear who was needed at each step within these processes.

SADT was used to complement the information in the UML diagrams. The SADT renderings implied a sequential ordering to the steps of the process, whereas the process was in fact far more complex and dependent on decisions taken by the medical staff. Thus, this kind of task-oriented decision-making was found to be lacking in the original diagrams. The cmap, on the other hand, clearly represented the decision points and branches resulting from them.

Finally, we note that the cmap was still not considered sufficient. Eventually, the authors created a Chapin Chart (aka a Nassi-Schneiderman diagram) [6], which amounted to a cmap without arrows connecting nodes. Instead, connected task nodes are physically adjacent. Indeed, a Chapin Chart is mathematically equivalent to an unlabelled graph. The result was a diagram that seemed simpler yet contained exactly the same information. Project stakeholders who reviewed the Chapin Chart found it "easier" to understand than the cmap. This reinforces our belief that layout and diagram structure is as important for comprehension as appropriate diagrammatic content.

The third project involved planning a concept mapping workshop for the Canadian Design Research Network (CDRN). In May 2007, a group of researchers from Simon Fraser University (SFU) in British Columbia, Ryerson University (RU) and the University of Toronto (UT) in Toronto, and École de Technologie Superiéure (ETS) in Quebec came together to organize the workshop. The purpose of this workshop was to showcase as well as explore the power of concept maps and how they can be used in collaborative design setting. It was hoped that this workshop would attract people from various areas of design research beyond engineering and architecture, as well representatives from industry, to promote concept maps in design. The workshop would be run at three different locations (SFU, UT, and ETS) and take advantage of distance collaboration software (including CmapTools).

At the May meeting, the organizers tested the facilities needed to run the workshop, and have a "dry run" of material to be presented at the workshop. AccessGrid (http://www.accessgrid.org) and Skype (http://www.skype.com) were used for general communication.

Certain features of CmapTools in this setting were noted as particularly relevant. It presented a very simple style of diagramming that all participants quickly became comfortable with, regardless of training or background. CmapTools can place a user-specified background image on a cmap. This was seen as very useful to the architects and urban planners present, because they could use CmapTools to annotate an image of a site, building, or community.

At the end of the session, all present were convinced the workshop would be very useful for all participants, and the technology had proved itself nicely. Unfortunately, funding for the workshop was cut shortly thereafter. We still hope to hold the workshop, but there will be some delay.

3.4 Summary of Case Studies

These case studies have highlighted our efforts to study the use of concept maps in design. The experience and information gathered from all this work will guide our future research to develop more powerful, yet simple, diagramming tools for designers.

4. LARGE-SCALE APPLICATIONS: PRODUCT LIFECYCLE MANAGEMENT VISUALIZATION

The authors had an opportunity to focus their visualization research on *Product Lifecycle Management* (PLM) for some time. Key considerations in designing a diagramming tool for use in the real, messy world of engineering design are highlighted by designing for this challenging management approach.

4.1 Design Visualization Issues in Product Lifecycle Management

PLM is an approach to product management that demands a systems perspective that takes the entire life of a product into account for design, production, use and disposal. Its effective implementation promises to 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 need to maintain consistent perspectives on the design throughout a product's life in order to make these complex management decisions means that PLM involves immense amounts of information management. The wide-ranging scope of potential demands on information also makes it exceptionally challenging to create effective software solutions.

Early design must tie into these solutions because PLM should, in theory, lead to constant revisiting of design thinking. Effective decisions downstream require an understanding of a product's intricacies and origins.

The authors have observed that existing PLM systems do not seem to properly support early design concept visualization. In 2005, at Ryerson University conducted a survey of small and medium-sized enterprises (SMEs) [3] and found that companies believe in the promises of PLM but that they see the lack of support for design, and especially conceptual design, as a significant issue. This makes sense from a conceptual perspective since most PLM systems we observed were built on CAD and numerical analysis tools which require thinking at a level of quantitative detail that is usually irrelevant and even *detrimental* to early concept design.

Visualization in PLM tends to consist of file trees and various free-form annotations. It also brings together the usual computer-aided engineering outputs like CAD, CFD and FEM by improving revision management and approval workflows. It helps with data management but does not seem to go beyond that towards augmenting design thinking. A case in point is an issue found during a visit to one of the major PLM vendors. They demonstrated key visualization features of a remote content sharing system. It was emphasized that this system enabled controlled sharing and visualization of physical parts from a web browser. The system was compatible with most CAD formats and could readily combine different file types into a single assembly to look for geometric interferences. When queried, however, the vendors admitted that it was not possible to explore the offending geometry simultaneously with finite element results on that part. Material shape supports loads, constrains movement or it may be a necessary result of a manufacturing processes. In short, material shape serves a function and changing it means making a decision about its function. The PLM solution met the integration challenge of geometry and information management but did not immediately support design thinking about how to resolve the conflicts properly since it failed to bring the information that was *truly* relevant to that level of decision-making into the appropriate context.

IHMC CmapTools is noted as a key tool in other sections of this paper. A study as part of Eng's graduate research involved taking a design from initial formulation through to systems design using concept maps to articulate, visualize and structure the information elements related to design thinking. The concept maps demonstrated the required

simplicity: showing how simple boxes and arrows of different styles were sufficient to represent the different levels of hierarchy throughout design work. Combining this study with the issues raised in PLM, a design was proposed for a "navigator" tool that combines the general flexibility of a concept map with features fitted to conceptual design work.

4.2 Concept Map "Navigator" Design Highlights

The PLM "navigator" must afford free-form expression in early design wile having capabilities for information organization and navigation needed to tie into the upstream stages of PLM. The tool's design seeks to achieve this through a combination of streamlined user-interfaces and engineering design-based map styles, which augment design work in a way that fits real-world practice. Critical elements were found to be the need to create immediate value during documentation work and an emphasis on features to augment design thinking. The design is presented in depth in [9] and key points are outlined in this sub-section.

As noted earlier, much of the preliminary design documentation exists as plain text. Creating these documents means performing synthesis on various outputs of early design, including discussions, meetings, and preliminary calculations. This means that a great deal of human work (and its associated resource cost) is spent abridging other materials into a form that may not contribute to improved understanding. Reporting, after all, is done after designing. In the context of PLM, report-type artifacts are required in recreating an understanding of a design, but the mere act of documenting also requires significant additional work. The solution, given that reuse of documents is not guaranteed, is to insure that documentation system create value *at the same time as they are used* for capture. Successful diagram-based systems such as [5] (which is deployed and used as a part of standard practice in a major international aerospace company) are also designed based on this principle – that the engineer must see value in "first use" before it a document has potential for reuse. Along these lines, one can also consider the negative impacts that computerized tools tend to have. They can interfere with the fast, social processes of design work if the user interface is inadequate or if it requires too much cognitive load to interpret. This is discussed in the following paragraphs.

In the context of supporting complex work activities, it was recognized early on that advantages sought by these information management systems are largely psychological, not technical [15]. Augmenting design thinking is a major theme in the navigator design. Key areas highlighted here are user interaction, and potential methods for fluidly managing concept structures.

The synchronous collaborative work in early design meetings provides a clear example of the need for effective interfaces. The discussion thus far has focused on the value of diagrams as a visual input, but a computerized tool is about the interaction cycle, the "dialogue" of representations, between the user(s) and the visualization [7]. Using existing diagrammatic tools in a meeting, one quickly realizes why design companies such as IDEO still advocate the use of numerous Post-It™ or similar notes [14]. It is very difficult to keep up with conversation using most diagramming software; paper-based methods offer distinct advantages in speed and unrestricted forms of interaction. By slowing the users, the computerized representation is interfering with that social process of work.

The navigator design seeks to alleviate this by automating many of the detailed operations in diagramming so the user's attention can be spent on actual design issues. Some existing diagram programs such as FreeMind (http://freemind.sourceforge.net) do an excellent job of providing keyboard shortcuts and automated layout. The free-form structure of concept maps, however, creates a more complex challenge. This is addressed by the concepts presented in Figures 3 and 4, which emphasize the potential of partial automation. It demonstrates a number of examples that streamline work on the "selected nodes" in the smaller sub-networks within very few operations. This arose directly from experiences where subtle layout operations were found to be the bulk of concept map rendering and refinement work. Aligning nodes, manually grouping text and searching for patterns by moving one node at a time constitute unnecessarily poor and cumbersome mechanisms for exploring the rich structure of design concepts.

Computer-mediated communication requires structuring and de-contextualization of information [21]. Text documents provide single narratives that are not always representative of the smattering of ideas discussed and dead-ends pursued in design work. As design reuse requires a more complete understanding than just a single "story", these many perspectives need to be maintained. Since they are part of the same "nearly decomposable system", [18] each of these perspectives is also necessarily connected to the others, through related or even overlapping concepts. As the design evolves, it is also inevitable that many maps and diagrams will need some sort of refactoring or reorganization. As concepts are carried through various types of diagrams and modified, maintaining the all-important layout of each

map is logistically difficult. The problem is that effective layouts are a combination of what the computer can interpret (graph layout algorithms) and what people need to see (manual layout).

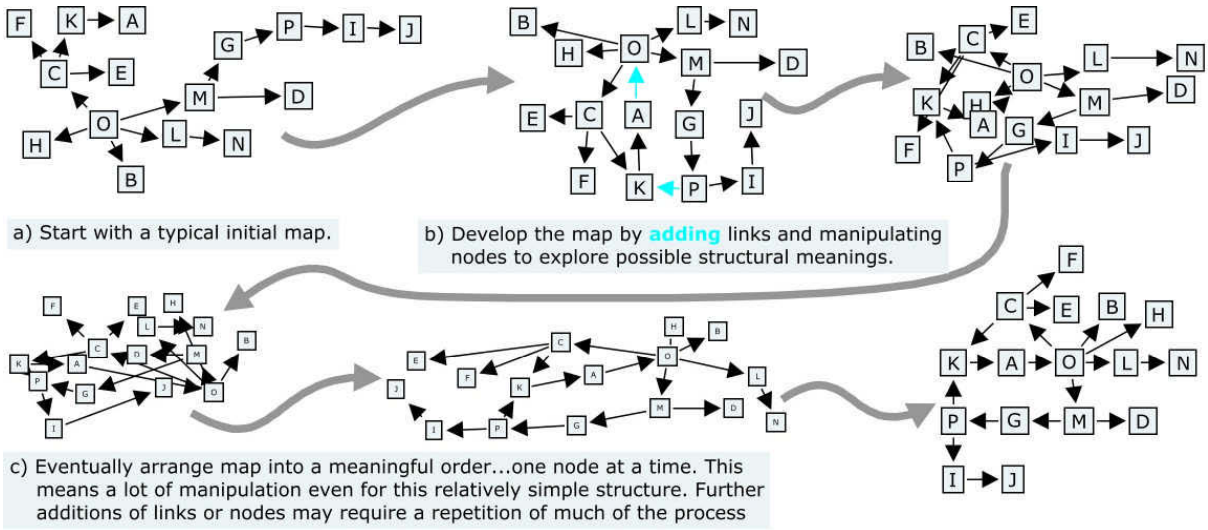

Fig. 2: A Typical Diagram Layout Scenario.

A typical map will start out as illustrated in Figure 2a by building or associating ideas one upon another. This layout is what one might expect in a mind map from a very short brainstorming session. The advantage of the computerized map is that this captured representation is easily evolved and integrated into existing material through further layout manipulation and hyperlinking. Taking that evolution to its conclusions (Figure 2c) however, can be as time consuming as the initial creation of the map). As the user examines the concept nodes, associations often bridge across or break-down the initial hierarchy. Different types of relationships and structures emerge through iterative experimentation with both the structure and the contents of the nodes. For existing tools, almost every operation of this exploration is manual which can waste a great deal of time without adding a great deal of value to the representation. The cognitive value here is in the structuring of the representation, not the use of the computer pointing device, yet as the level of abstraction of the structure being manipulated increases (from nodes to groups to groups of structures) so does the sheer number of raw manipulations.

The concepts laid out in Fig. 3 show simple ways of alleviating some of that tedious manipulation which go beyond the simple layout and selection methods in surveyed tools. The key is *partial* application of automated layout operations (indicated in **bold**), selected by the user depending on what they think is most meaningful. The process starts with the initial map setup from Figure 2b and results in a much cleaner representation of the final layout. The value of this final layout would depend on the contents, but the value of the operations is a significant decrease in rendering time from start to finish with the freedom for more creative experimentation with the structure.

It must be noted that a few of the features already exists in limited forms. Examples of these include overall layout of maps in many tools, the selection of parent and children nodes (ex: Compendium) and the alignment of groups of nodes (ex. CMapTools). The operations suggested in Figure 3 were chosen because they were often found necessary during studies with existing tools.

The next step in augmentation beyond simple layout manipulation is a particular form of automatic layout recognition that provides a richer exchange between the user and the computer system. Figure 4 details the outputs the proposed system. The idea is to be able to start with a given map (or selection within that map) and to have the computer search for and display a number of options. The user then accepts one or many to be applied to the map.

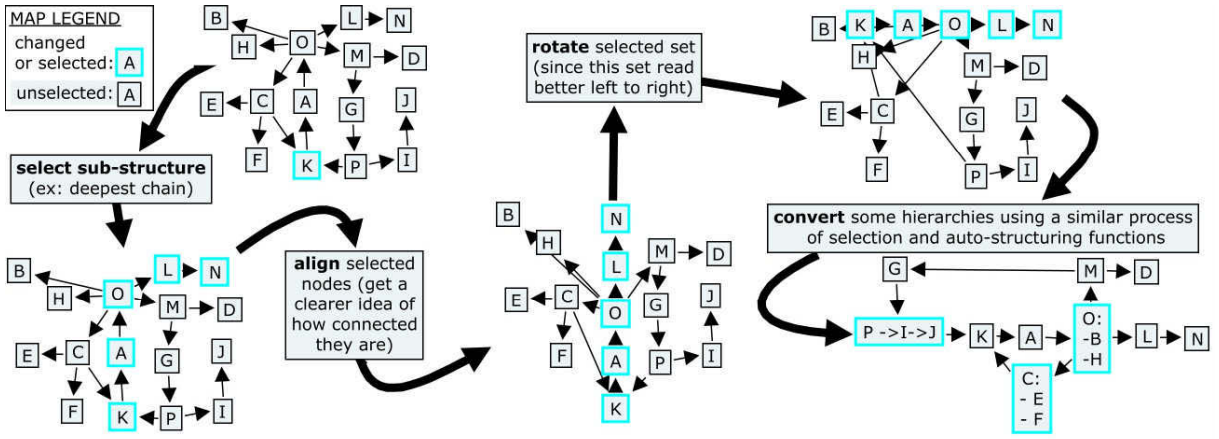

Fig. 3: Simple Concept Map Layout Manipulation Aids.

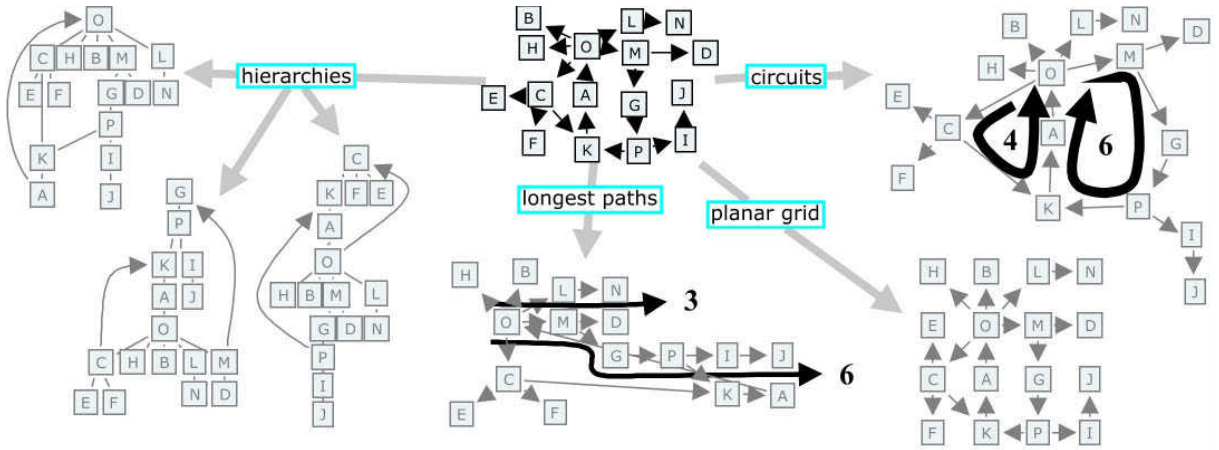

Fig. 4: Layout Recognition Augmentation Concept.

This concept contains a number of important features. Firstly, it strikes a balance between the advantages of computational speed and human thinking. Given modern computing hardware, these simple graph layout searches and renderings are computationally trivial even though they would require quite a while for a person to perform manually (almost an hour in the case of the Figure 4 using CMapTools). On the other hand, the computer still does not "know" what it is encoding, so the decisions about meaning are kept in the hands of the user. A second important feature of this design is the cognitive value of juxtaposing different related options. This affords simultaneous, random comparisons between different map options which should inspire more ideas on effective layout. The third feature supports its simplicity and feasibility: the success of this approach does not require particularly robust recognition algorithms since it presents many options at once. As an example: a recognition system that only has a 50% success rate at proposing structures of interest has a 93.75% chance of providing at least one successful option if 4 results are presented at once. This design aims to be effective through rapid human-computer feedback instead of a futile attempt at computational perfection in a qualitative application.

An OPLM navigator tool must ultimately provide multi-scale, multi-hierarchy views of systems that remain consistent and up-to-date throughout PLM activities. The proposed concepts provide some of the mechanisms required to create this structure and make its creation process available for computer-based capture. Other existing features like various diagrammatic languages for specific applications and the transclusion of elements across many maps are also critical. Something further is required, however. While the content remains qualitative, much of the underlying structure of the diagrams and the need for cross-map "book keeping" as elements are modified in different views demand some

automated rendering and updating across a system of diagrams. This hints at the need for diagrammatic tools similar to present parametric CAD systems. The sequential operations of diagram authoring and version control are analogous to the operations on 3D part drawings: specifying relationships, constraints and geometry. The core challenge remains, however, that this parametric system *must* consistently combine layouts and styles that are part formalized and part manual. This is a challenging but critical open research question of the navigator work.

5. DIAGRAMMING BY DESIGN PROCESS STAGE – DESIGN SCHEMATICS

It is clear to the authors from the work outlined above and our review of the literature that there is no single tool or method well suited to diagrammatically representing information in early engineering design. Salustri has undertaken a project to develop a systematic diagramming framework for this purpose. Called *design schematics* (DS), is intended in the long-term to capture all the lessons learned from our other work in the area, and provide an integrated framework for capturing and sharing diagrammatic information for engineering design.

Within this project, we subscribe to the notion that learning is the integration of new information into a learning agent's existent cognitive structure, which can be seen as a richly inter-connected network of information upon which cognitive processes operate [16]. 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 design answer 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 into a designer's mind by being as compatible as possible with the network-based model of learning described here.

It is also evident that there is no one diagramming scheme that is ideal for every possible kind of diagram. On the one hand, a *single diagramming scheme* would be the simplest to use because of the universal applicability and consistency. On the other hand, we have found that the *layout* of a diagram is just as important as the diagrammatic entities to communicate information. Different layouts serve different purposes, so the nature and intent of the diagram will guide the selection of a layout. Thus, *multiple diagramming schemes* provide important benefits as well. Finding the balance between the simplicity of a single scheme and the expressive richness of multiple schemes is an essential and open research question. However, in keeping with the principle of simplicity, the authors are starting with the simplest possible system, and then adding to it only when necessary to significantly enhance expressive richness.

For these reasons, DS is based on *concept maps*, which are hypergraphs connecting nodes that describe concepts with arrows that describe relationships. Upon this base, we begin to add structure intended to address directly the information visualization needs of designers. Below is a description of the current "versions" of diagram types that constitute DS. The specific diagrams in this paper were developed with conventional diagramming tools because we have not yet written appropriate software. One can regard these diagrams, however, as a requirement specification of what appropriate software would have to produce. We will use the example of a powered screwdriver (taken from [19]) to help ground the discussion.

We note that the order in which the diagrams are presented below does *not* imply that they must be used only in that order. In this regard, our goal is to provide diagrammatic "building blocks" that can be rearranged to meet the needs of a particular design process.

To begin, we consider the description of a *usage schematic* (US), which allows designers to think about how a product might be used in a given context or operational environment. Information to build a US can come from focus groups, consumer reports, marketing information, designers, etc. A US helps designers focus on customer needs, product use, and issues that arise as a result (e.g. safety). A possible US for the powered screwdriver is shown in Figure 5, which shows three task paths. The links in the US show the order of actions taken by users. The terminal node at the right of the US 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".

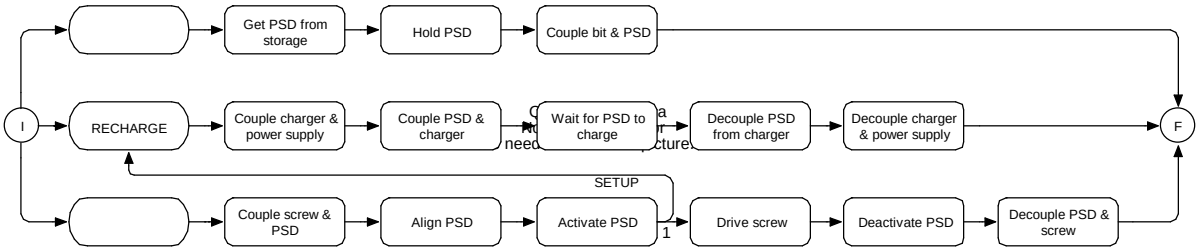

Fig. 5: A usage schematic for a powered screwdriver.

The links in the US 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 *USE* 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 design implications of the US. Figure 6 shows the *SETUP* usage scenario, with some of these other entities. The branches in Figure 6 are not labeled 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 red nodes are constraints.

One can extract all the PCs, FRs, and constraints from a US and, rearrange them to form a *product design schematic* (PDS) – a DS of the basic characteristics and functions expected of a suitable design – essentially a requirements specification for the product. Our PDS representation combines aspects of the work by Pugh [17], and Dym and Little [8]. The details of our representation are not important here because we are developing DS to support a variety of different representations. (Recall: a key feature of the DS approach is to ensure the flexibility to support the actual processes used by design engineers, which vary from one situation to another.) Our goal here is to show how a PDS diagram *may* represent important design information. A partial PDS of the screwdriver is given in Figure 7.

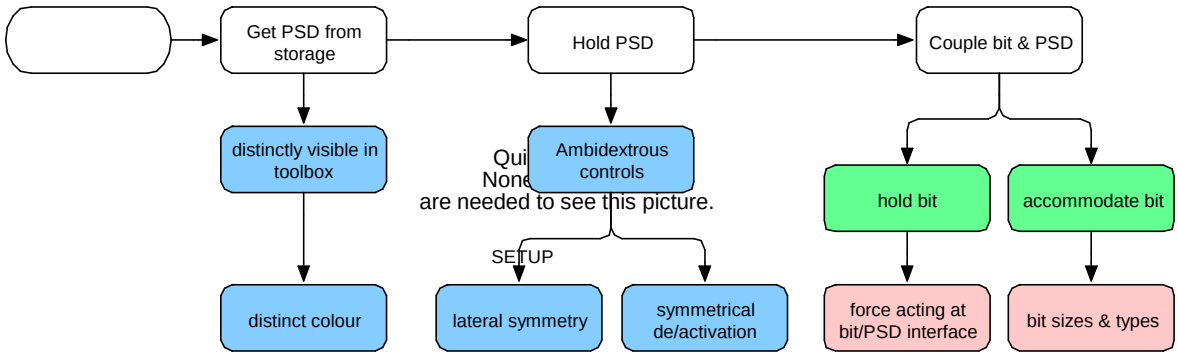

Fig. 6: Adding PCs, FRs, and constraints to the usage schematic.

The rules for a PDS are quite simple. The links are unlabeled because the node colors imply the relationships. Bold-bordered nodes are of high importance, nodes without borders are of low importance, and the others are of middling importance. The left-most "column" of PCs defines some of the basic characteristics of any product. The next two columns of nodes show some PCs and FRs specific to the problem. The number of columns may vary 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. PCs and FRs in turn link to constraints that limit a product's performance with respect to the FRs and PCs. A PDS is also arranged such that no link (arrow) has more than two bends in it. While this can produce

diagrams that are quite sparse, it also facilitates "clumping" of related entities and highlights gaps in the requirements. These clumps and gaps can clarify informational features in the requirements that would likely be hidden in conventional, text-based requirements documents. For example, it is clear that there are substantive gaps in the constraints for the requirements in Figure 7.

We note that we will make extensive use of color in this paper, as a simple and meaningful way to communicate information. However, we are not suggesting a set standard meaning for colors. Ideally, appropriate software for these kinds of diagrams would allow color maps to be defined by the user. Thus, a given individual, team, or organization can create a color scheme that is most meaningful to them. This can also address situations where users are, for instance, color blind – by selecting colors that are in fact visible to the user community, or even by not using color at all.

PDSs that form hierarchies are consistent with "uncoupled" designs arising from the Independence Axiom of Suh's Axiomatic Design [20]. However, such designs are rare. The highlighted links (in red) in Figure 7 indicate interactions between PCs and FRs, which appear as cross-links in a PDS and which indicate graphically cross-coupling between requirements, which turns the hierarchy into a network graph. There are specific characteristics of the graph in Figure 7 that allow detection of the cross-links. Most obviously, direct links between two FRs directly represent coupled design functionality; this is shown by the two red links in Figure 7. There are likely other graph theoretic properties that we will uncover as we continue to study and develop DS.

This kind of visualization can benefit design teams by helping them to, for instance, identify complexity in their understanding of the design problem. It can also help project managers and requirements engineers specify clearly the expectations of the client. While the PDS does not *automate* design activities, it should help them reach a deeper understanding of the design problem faster.

From a PDS, 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 in [19] 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 6. The difference between the diagramming style in [19] and that of FSs is that FSs are developed top-down – from the product as a whole down through subsequent levels of detail – whereas the diagrams in [19] are typically developed in a bottom-up fashion – from primitive functions up to more complex ones. Our method is for product development whereas the method in [19] is typically for reverse engineering. We do note, however, that most of the layout heuristics in [19] (e.g. the dominant flow heuristic) are also applicable to FSs.

To save space, only some functions from the PDS are shown in Figure 7; our intention is to demonstrate the DS language, and not to provide a complete account of the screwdriver problem. The green nodes are identical copies of their counterparts in the PDS. The white nodes are functions from other parts of the Usage Schematic; the fact that these functions need to be here indicate a level of functional coupling between the parts of the usage scenario that may not have been apparent to the designers at the outset. Since we envision each diagram type as analogous to a "view" of a single database, it is reasonable to expect that there will be no duplication of data between different diagram types, thus helping to ensure integrity of the stored information.

A FS can have three kinds of links between nodes: thick links represent mass flow, thin links represent energy flows, and dashed links (not present in Figure 8) represent flows of information. Mass flows can represent things like fuel moving from gas tank to engine, or passengers getting on and off an elevator. Energy flows represent things like the flow of electricity into and out of components, and solar radiation, as it moves between system elements. Information flows represent data moving between computer elements, or the light reflecting to the driver from a rear-view mirror. We note that identifying a flow depends on the designer's intent. For example, if one were to give a person a large book, it could be modeled as a mass flow (if the person needs a doorstop), an energy flow (if the person were going to burn the book for warmth), or an information flow (if the person were to read and learn from the book).

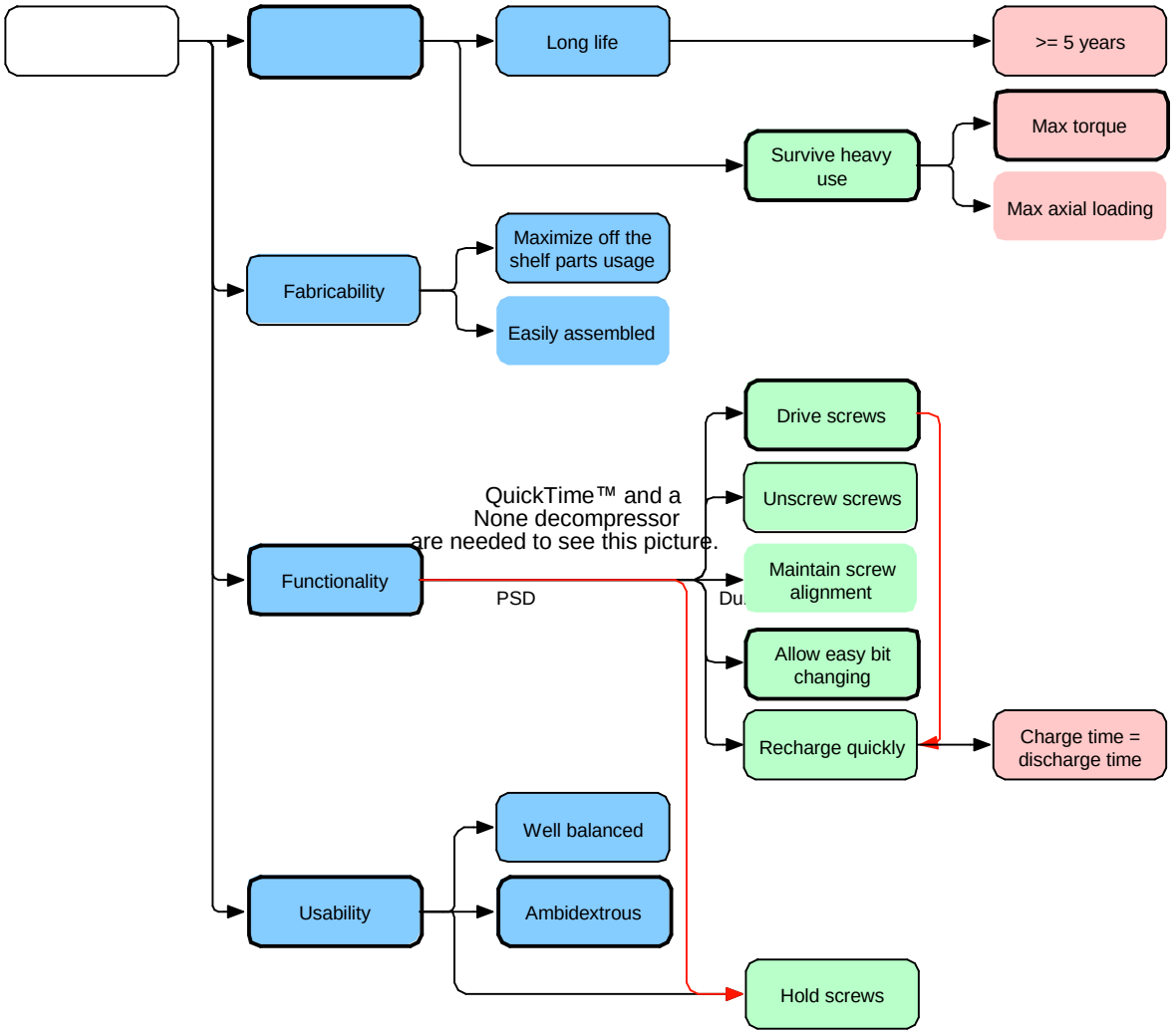

Fig. 7: A partial product design schematic for a powered screwdriver.

Inputs come from the left, and outputs exit to the right. (One could reverse this convention, or even arrange it vertically, without loss of semantics.) Furthermore, we wish to distinguish inputs that are "consumed" by a product from inputs merely used by it. In IDEF, the latter inputs and output go into the top of a node and exit from the bottom, respectively. We have not adopted this convention in design schematics because it would make the diagram more cluttered and thus more difficult to understand. (For example, consider if the "Bit" in Figure 6 exited from "Drive Screws" downwards.) Instead, in DS we use color to distinguish consumed from non-consumed inputs.

Finally, one can use an FS to develop a *product architecture schematic* (PAS). Based on diagrams of the same name in [22], our PASs address top-down design rather than the bottom-up approach described in [22]. A PAS for the powered screwdriver is shown in Figure 9. Again, our goal here is to show that a PAS can represent important design information diagrammatically, and not necessarily to determine a complete or definitive PAS for the powered screwdriver. Systems are noted in the PAS as grey nodes enveloping the functions they provide. Note that the function "Drive screws" spans the "Bit Change System" and the "Drive System." This indicates that there is a functional (but not necessarily structural) interface between the two systems. In light of heuristics such as *dominant flow* (per [19]), one can see that the systems could correspond to product modules, and that one can envision that software could identify possible modules/systems from a graph representation of the PAS.

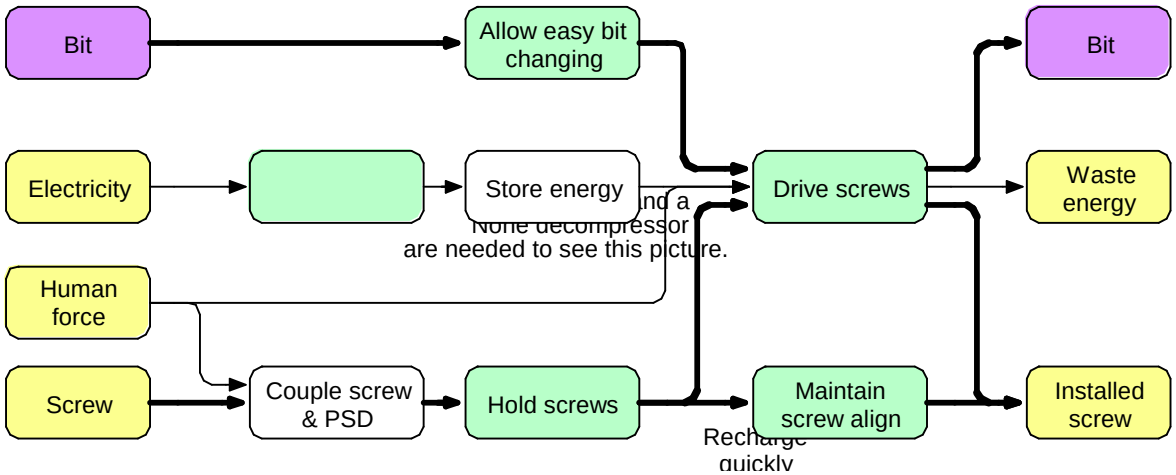

Fig. 8: A partial function schematic for the powered screwdriver.

We note again that design schematics are not prescriptive – they do not *force* a designer or design team to do anything. Design schematics are *descriptive* – they embody a language to capture and communicate information. A designer will use design schematics as a reasoning tool.

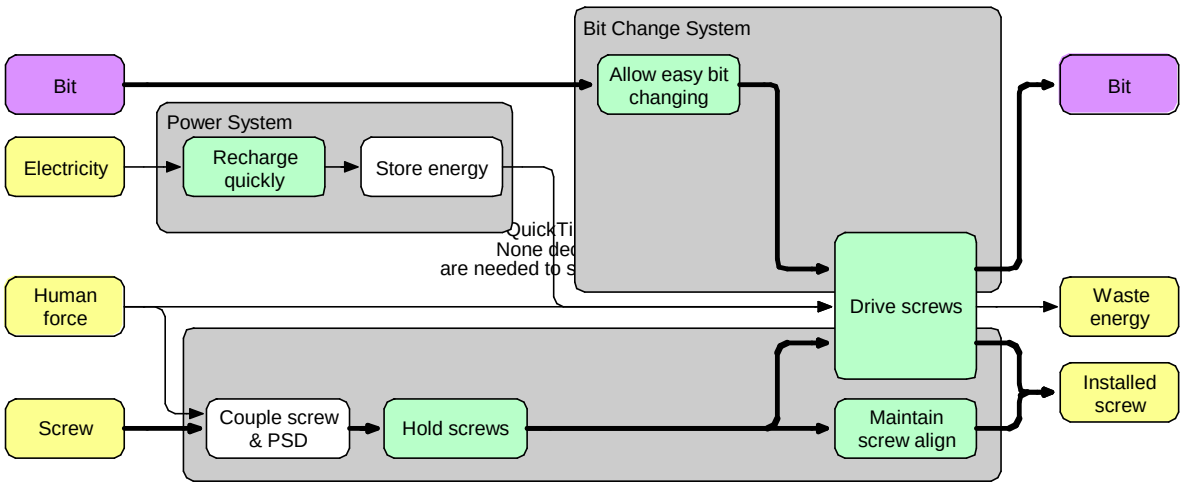

Fig. 9: A partial product architecture schematic for the powered screwdriver.

Other representations of systems are possible in DS. One example involves the simultaneous presentation of a product's basic functional, systemic, and physical structure. To demonstrate the advantage of such a presentation, we consider first Figure 10, which represents graphically a planetary gear system. An undergraduate student familiar with planetary gears but not with DS drew the figure; we think of it as a "naive" diagrammatic representation. Since the overall topology of a DS is not known a priori, one might start by simply grouping similar kinds of entities: functions and components. Shades of grey are used to distinguish the two kinds of information. Different kinds of links are supposed to show relations between components, between functions, and between both.

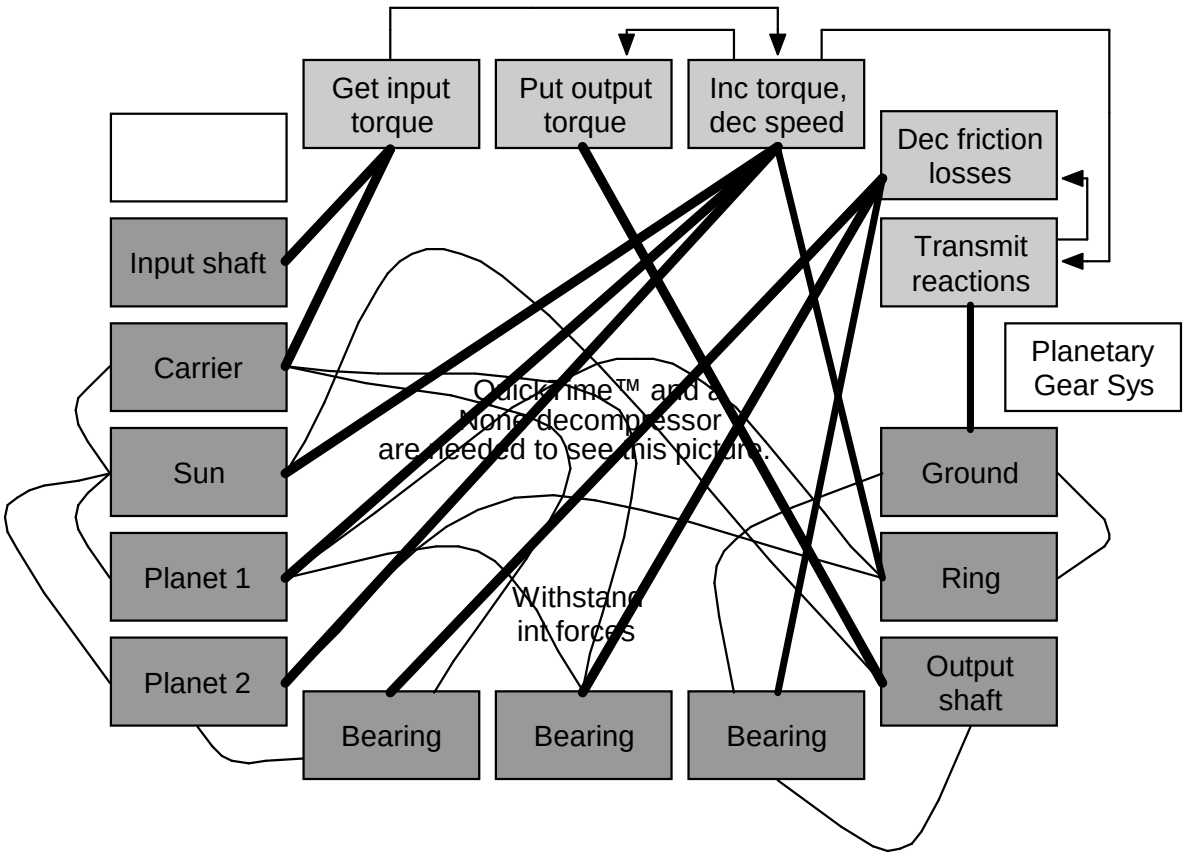

Fig. 10: A naïve diagram for a planetary gear train.

However, very little of the overall information structure is visible in this view. The links are confusing. In addition, two nodes – *"withstand internal forces"* and *"planetary gear system"* – cannot be connected reasonably to the rest of the drawing. In the first case, withstand internal forces would connect every component node to every other component node, making the diagram impossibly complicated. In the second case, planetary gear system is a label for the whole diagram. (This is why the student who drew this diagram did not connect it at all, and left it uncolored.) There is no simple way to connect it to the rest of the diagram given the modeling approach taken by the student.

Figure 11 shows the same information about the planetary gear system, but arranged according to a small set of deterministic rules using the design schematic approach. This new figure clearly gives a much better impression of the overall system. Using partially overlapping nodes has eliminated many links, and there are only two kinds of links instead of three as in Figure 10. The rules for this kind of diagram are generally as follows.

- Functions "contain" the components that provide them.
- General functions contain more specific functions.
- System interface elements reach the border of the containing system.
- Colors distinguish physical connections from functional ones.
- Links are arranged to minimize crossing lines, modulo maintaining a layout that is suggestive of the physical structure.

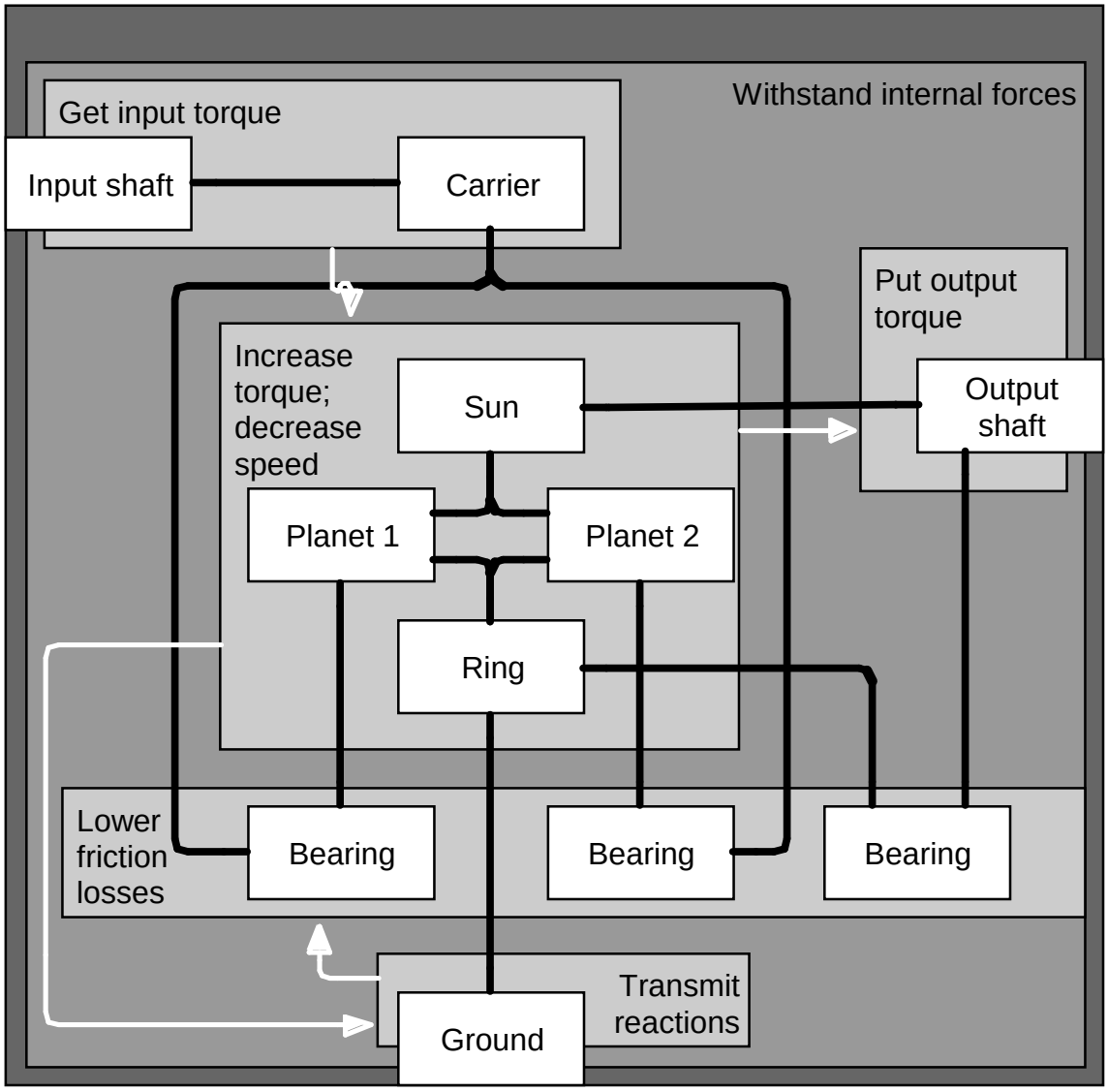

Fig. 11: A "better" product architecture for a planetary gear system.

In Figure 11, "withstand internal forces" now fits nicely into the overall description without unnecessarily complicating the diagram. Informal tests have revealed that engineers respond more quickly and accurately to questions based on Figure 11 than Figure 10. Clearly then, careful selection of diagrammatic forms facilitates understanding.

One might reasonably ask if having many different diagrams – as shown above – fails to meet our basic requirement of simplicity. This is a legitimate concern until one considers a single hypothetical diagram containing all the information shown in, say, Figures 4 through 8. We have attempted to construct such diagrams for small but non-trivial problems; the results are so complex as to be nearly unintelligible. Not all the simplicity in the world will matter if a diagram cannot be understood. Therefore, we have chosen to develop diagrams that correspond approximately to generally accepted major tasks in design engineering. We further simplify the diagrams by reusing many graphical elements that have similar or identical semantics in different diagrams; for example, once the color of, say, a functional requirement, is set, it assumes that color in every diagram type.

Salustri is also working on the computerization of the DS system of diagramming. Ideally, all graphical aspects of the rendering would be configurable at run-time, so that (a) different kinds of diagrams can be rendered using the best possible layouts, and (b) different organizations can customize the application to support whatever diagramming styles and conventions they prefer. At the moment, we are building a small application using the Python programming language and the TkInter widget set. The application will initially support conventional concept maps only, because as mentioned above, concept maps represent what we have identified as a lowest common denominator for diagramming in design settings. We will then begin systematically enhancing the application based on feedback from users, experiments, and further study of the research in interface design and cognitive science.

6. CONCLUSIONS

The authors have provided a brief overview of their recent work in diagramming tools and methods for the early stages of engineering design. Our goal is to develop new and useful ways of visualizing non-geometric information that is particularly essential in early design. While this work is ongoing, we have found strong evidence that there do exist general guidelines or rules that facilitate comprehension of engineering design information, and that these rules can be embedded in software.

7. REFERENCES

[1] --: Integration Definition for Function Modeling, NIST Defence Technical Information Center, 1993.

[2] Andries, M.; Engels, G.: A hybrid query language for an extended entity-relationship model, J. Visual Languages and Computing, 7, 1996, 321-352.

[3] Anwary, M; Salustri, F. A.: Results of a survey of product lifecycle management issues, Ryerson University technical report, 3 June 2005. (http://deseng.ryerson.ca/y/pub/Research/PlmSurveyReport2005.pdf)

[4] Blackwell, A. F., ed.: Thinking with Diagrams, Kluwer Academic Publishers, 2001.

[5] Bracewell, R. H.; Ahmed, S.; Wallace, K. M.: DRed and design folders, a way of capturing, storing and passing on, knowledge generated during design projects, Paper DETC2004-57165, Proc ASME Design Engineering Technical Conferences, ASME, New York, 2004.

[6] Chapin, N.; Denniston, S. P.: Characteristics of a structured program, ACM SIGPLAN Notices, 13(5), 1978, 36-45.

[7] Dearden, A.: Designing as a conversation with digital materials, Design Studies 27(3), 2006, 399-421.

[8] Dym, C. L.; Little P.: Engineering design: a project-based approach, Wiley & Sons, New York, 2000.

[9] Elmqvist, H.; Mattson, S. E.; Otter, M.: Object-Oriented and Hybrid Modeling in Modelica, J. Europeen des Systemes Automatises, 35, 2001, 1-10.

[10] Eng, N. L: Design of a visual concept navigation tool for engineering design and open-source product life-cycle management, Master's Thesis, Ryerson University, Department of Mechanical and Industrial Engineering, 2006.

[11] Hauser, J. R.; Clausing, D.: The House of Quality. Harvard Business Review, 1988, 63-73.

[12] Hicks, B. J.; McAlpine, H. C.; Culley, S. J.: The fundamentals of an intelligent design laboratory for researching the impact of tools, teams and technologies on information use and design performance, Proceeding of International Conference on Engineering Design ICED 07, 2007.

[13] Karnopp, D. C.; Margolis, D. L.; Rosenberg, R. C.: System Dynamics: A Unified Approach, Wiley & Sons, New York, 1990.

[14] Kelley, T.: The Art of Innovation: Lessons in Creativity from Ideo, America's Leading Design Firm, Doubleday publishers, 2001.

[15] Nelson, T.: A File Structure for the Complex, the Changing, and the Indeterminate, ACM 20th National Conference, New York, 1965.

[16] Novak, J. D.; Gowin, R.: Learning how to learn, Cambridge University Press, Cambridge, 1984.

[17] Pugh, S.: Total design, Addison-Wesley, London, 1991.

[18] Simon, H. A.: The sciences of the artificial, MIT Press, 1996.

[19] Stone, R. B.; Wood K. L.; Crawford R. H.: A heuristic method for identifying modules for product architectures, Design Studies, 21, 2000, 5-31.

[20] Suh, N. P.: The Principles of Design, Oxford University Press, New York, 1990.

[21] Tuomi, I.: Data is more than knowledge: implications of the reversed knowledge hierarchy for knowledge management and organizational memory, J. of Management Information Systems. 16(3), 2000, 103-117.

[22] Ulrich K. T.; Eppinger S. D.: Product design and development, McGraw-Hill, New York, 1995.