

“Rugplot” Visualization for Preliminary Design

Nathan Eng, B. Eng.
Ryerson University
neng@ryerson.ca

Prof. Filippo A. Salustri, Ph.D., P.Eng.
Ryerson University
salustri@ryerson.ca

This paper describes the extension of a multivariate scatter plot for qualitative use. The plot combines effective diagramming practices and modified axis dimensions to cohesively summarize qualitative analysis in the early, unstructured stages of design. The rugplot is a flexible tool that improves understanding of the “hard” limits of the problem, self-imposed “false” constraints, and directions for concept modification and recombination. An example is given with Product Lifecycle Management (PLM) software that makes the case for the introduction of a new class of tools. Improvements are still needed to streamline data collection and to automate graph generation.

1. Introduction

The *rugplot* visualization presented in this paper was developed to augment the understanding of characteristics of tools in the area of Product Lifecycle Management (PLM). It enabled the characterization and comparison of a wide set of tools while clearly highlighting incomplete features of the entire studied toolset. The process also provides a basis for establishing theoretical limits of potential tools.

The driving research was the creation of a navigation system for Opensource Product Lifecycle Management. There was a need to specify the existing “functional space” occupied by current software and to determine where development was required. It was also intuitively inferred that there may be some limitations to tools inherent to the types of desired functional combinations. Since the research began with a focus on visualization, it seemed natural to apply that research to the initial design specification problem.

1.1. What is Opensource Product Lifecycle Management?

Through previous research, the authors have defined Product Lifecycle Management as “a comprehensive approach to managing a product throughout its life, from initial design through to final disposal.” Among many other things, it is intended to enable faster time to market for new products, improved design revision control, better design for sustainability and re-use of work from previous projects.

Most PLM technologies are targeted at large enterprises. They require substantial investment in

proprietary software that is too costly for most Small and Medium-sized Enterprises (SMEs). The Opensource PLM (OPLM) project aims to provide SMEs with the ability to do effective PLM without the prohibitive resource and finance requirements of the larger systems.

PLM applications deal with complex, massively interconnected information. The authors have found that PLM requires better methods for information interaction than are currently available. The thousands of disparate information elements associated with all parts of a product's lifecycle need to be organized and understood. This understanding requires the presentation of information, context and rationale so the user can think at a higher-level about the many opportunities within a design. These missing parts in existing PLM systems stimulated the search further a field for other potentially useful concepts. The OPLM “visual navigator” is the main development thread at this time.

1.2. The power of visual tools

Computers provide unprecedented information and data management capabilities but they cannot make wise decisions. The human mind remains the best tool for making decisions in the early, informal phases of design. The navigator must somehow connect the information access capabilities of computers with the thinking abilities of the designer. Visualization is the key because it enables the highest density communication between these two types of processors.

IBM, which already has a proprietary PLM system, does not have the kind of diagrammatic visualization systems prompted by this 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.” [1]

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

The *rugplot* is an example of the kind of visualization that could be used for those breakthroughs in OPLM

2. Designing the rugplot

The original description of the rugplot scatter plot was found in E. R. Tufte's book *The Visual Display of Quantitative Information* [2]. The following discussion on visualization practices derives from that reference as well as Tufte's other books on graphical design [3, 4]. These books provide a wide range of refinement directions for all kinds of visualizations. Key concepts are presented here as they apply to this kind of rugplot. They address issues like formatting, legend, juxtaposition and context.

2.1. Format

It is important to select the correct format for a given visual. Presentation formats such as tables, graphics, and text are each suited to convey particular types of information. The example in this paper shows the progression from text description, through tabular summary and graphical representation. A table of graphs or text paragraphs within a graphic can also achieve things that the separate styles could not. Many document writing styles encourage the separation of these elements, but the key is to convey understanding of the specific problem. Experiment as much as possible.

Format can also mean the target media of the graphic. Resolution and colour depth can have a great impact on any graphic. The visual in this paper was originally designed for use in colour but line styles and careful text layout allow it to work in gray scale as well. In terms of format change, it would help to have fully computerized graphic generation for quick

revision and free-form investigation of the datasets. Unfortunately, the tool is not yet that developed.

2.2. Legends and multifunction graphical elements

Given the limited space allowed for any visual, it is important to use multifunctional graphical elements. This facilitates reasoning and maximizes the value of a graphic for a given amount of space.

When users are required to remember a legend, they are not thinking about the problem at hand but about the tool used to represent the problem. It follows that their problem-solving ability is impaired to some degree. As a rule then, it is probably best to *eliminate the need for a legend*. Use full text instead of abbreviations and repeat words where there is room. The rugplot uses the layout of words, repetition and consistent line styles to indicate the meaning of each datum. It may seem redundant to repeat words but in this case, the aim is to reduce cognitive load.

2.3. Finding the right context

Time is not always the best context. Tufte argues that “... descriptive narration is not causal explanation” [5]. Consider, instead, that there are many classes of information distinction. Items can be distinguished nominally, ordinally, or quantitatively. “Apples” and “oranges” are nominally different. They are simply two separate classes. “Right,” “OK,” and “wrong” might fit an ordinal scale. There is a direction but no meaningful quantitative difference that allows expressions such as “2 X wrong = right.” Quantitative distinctions are what most people are used to seeing on graphs. Like a time scale, they provide a nice mathematical consistency but they may fail to clearly resolve meaningful differences. The other types of scales rely on clear conceptual differences.

This exercise in varying context leads to one of the key concepts of this rugplot tool. As long as all of the concepts can be measured against a given “scale,” there is a chance to find interesting correlations or gaps before investing significant resources into precise quantitative analysis of different concepts in question.

2.4. Parallelism and juxtaposition

Parallelism in space (or juxtapositions) is useful for direct comparison. Tufte contends that they are usually better than animation because users can quickly and arbitrarily consider aspects of objects simultaneously within their “eyespan” [6]. This may be another facet of the rugplot's potential as it allows the user to

experience the transformation of the problem space across different contexts while returning along their path to examine all of the subtleties of the change.

2.5. *Density and empty space*

While it is important to maximise the use the limited space of a visual, this does not include elimination of all empty space. In the rugplot, for example, spacing provides clear structural grouping of the graph elements. Empty space in the graphs themselves is specific information about the analyzed concepts or the plotted dimensions. The graphical “0” is just as important as anything else on the scale.

In most cases a visual is using space that could be occupied by text. Given this, it would be nice for it to provide at least as much information as the text it replaces. This concept is especially important when designing computerized graphics because every bit of the display counts.

3. **Creating a rugplot for early design**

Given the preceding graphical considerations, the creation of a rugplot separates cleanly into five linear steps that also provide a framework for considering specific sub-components of the design problem as well as details of solution concepts.

First, select various dimensions of interest from the design requirements. Within each of these dimensions, create a well defined scale to categorize the concepts. The clearer the definitions, the easier it will be to categorize the tools.

Second, assign each tool a value or range of values within each dimension. Also note the characteristics of the desired design.

Third, select the dimensions which reveal relevant differences between the concepts. This is best determined by placing a results summary in a table. If there are few variances in a given dimension, plotting it will not show any differences between the concepts and that probably contributes little to the analysis.

Fourth, plot the results on a rugplot in various combinations. Note the desired ranges directly to the axes to highlight the needs of the design problem in parallel with the results.

Lastly, consider the results. Look at overpopulated and sparse areas. If tools are missing from areas, is there a good reason? Was this question ever asked before? Also consider correlations. Is this a real relationship that has emerged or an artifact of assumptions on the part of those doing the analysis?

4. **Rugplot example for PLM tools**

The authors will now present a detailed case study of how this tool was applied in research for the definition of requirements for an OPLM tool.

4.1. *Define Dimensions*

The following outline is a description of the axes of the PLM rugplot. All ranges in this example are characterized by an *ordinal* progression. They were selected from early requirements in the OPLM project.

4.1.1. Knowledge Level

Good PLM tools are knowledge management tools. Davenport and Prusak identify seven “dimensions” of knowledge [7]. One dimension involves how firmly one can capture it. Those authors refer to this dimension by the terms “tacit” and “articulatable.” Further division was added here to improve the resolution within they dimension. They are:

- Strictly Tacit: knowledge that is impossible to describe in a way that it can be completely passed to another person;
- Networked Captureable: non-articulated knowledge whose general form is only visible as the interconnection of concepts. It is too non-linear and piecemeal for a narrative;
- Narrative Captureable: non-articulated knowledge of a user which can be brought out and passed to another through a linear story. It represents the knowledge but may not be sufficient to transfer it;
- Articulatable: closer to information, can be codified into documents, processes and technologies in a way that is totally transferable to a user, whether or not they are ware of the knowledge.

4.1.2. Computerization and automation

This is the degree to which the tool is dependent upon or aided by machines. It may be:

- Paper or physical only: the process is not realistically feasible inside a computer. Examples are activities that require large surface areas for free-form sketching like IDEO-style brainstorming [8] or a great deal of physical interaction such as prototyping;
- Manual: the tool can be implemented on paper and pen but can be replicated inside a computer to some degree;
- Mixed: needs both paper and computer assistance;
- Total: can use in computer alone;
- Computer Required: Not useful outside the computer except for trivial cases. Most finite element

simulation tools are like this.

4.1.3. Structure

How strictly defined is the use of the tool? The categories defined here are:

- None: generates useful work or conclusions without any application of rules;
- Some: a few rules, creativity still required;
- Refinement practices: definite direction for getting a solution;
- Partial Algorithm: Some “judgment” required to get or select a good solution;
- Algorithmic: totally defined process for getting from inputs to useful results.

4.1.4. Richness of Understanding

This dimension touches on the level of interconnection between elements and how closely they reflect the material that they attempt to capture. The better the capture of information, the more understanding the user can glean from the use of a tool. PLM information tends to be highly networked so this would infer that deep learning tends to be easier with tools that can operate on a free-form network. The ranges are:

- Cursory: Could provide a quick summary of information and knowledge. This is like a pamphlet's worth of understanding;
- Shallow: provides some discussion or decisions-making material. It allows some basic conceptual structure to be understood. This is like attending a short seminar or a topic;
- Moderate: gives understanding of the scope of a whole area of understanding as well as a means of learning all the basics of the material. This level of understanding is akin to what is reached when taking a university course;
- Deep learning: enables understanding that allows the creation of unique elements or totally unique connections.

4.1.5. Learning Speed and Complexity

The authors chose to use “learning speed” as a simple indication of the relative complexities of the various tools. Assuming the user has technical background adequate to the given product, this dimension would answer the question: “how quickly would the target user become accustomed to using the software without aid?”

Significant differences in this aspect are noted on an exponentially increasing range denoted as: nearly

instant, “5 minutes”, “1 hour”, “half-day”, whole day, a week, a month, a year and several years.

4.1.6. Cohesion

How difficult is it to use the tool in everyday operation? When performing work with the given tool, this measures how much effort is spent using the tool vs. generating significant work on the product. The distinctions are:

- Manual: uses different tools on different media that are not co-located. The task is mostly handled in the user’s mind and through their physical manipulations;
- Connected: different interfaces are used for each part, but a user can connect parts in a meaningful way. This describes most real-world manufacturing projects where there both real objects and data files to be managed. One can manage parts in a database on a computer, but recalls and rework don't really happen “at the push of a button”;
- Compiling: All parts of the tool eventually captured on one platform. This could be a case where all the tools on a connected project are digital, for example, and a computer could be used to manage all of them.
- Integrated: all in one platform, interface and application. Some larger PLM [9, 12] packages attempt to do this.
- Transparent: This is a “holy grail” in design tool requirements. It would let the user remain task-focused with minimal awareness they are moving between parts of the system or interfaces. This is conceivable using some sort of ubiquitous portable computing solution but this would need some sort of novel capture mechanism to be truly transparent.

4.2. *Rating each tool*

For each tool enumerated, a short introduction is given and comments are made on dimensions of interest. These ratings are fairly general because they are based on the basic media available from each manufacturer. This is typical of the level of information one has in the early design phase. Most of the work in this analysis should be focused on understanding the problem.

4.2.1. Desired Tool

The purpose of the OPLM project is not simply to copy what currently exists but to look at the overall problem and to design a cohesive solution. As such, many of its desired aspects may seem over idealized but that only emphasizes the true scope of the OPLM navigator problem.

A complete knowledge system must be able to capture, convey or facilitate the exchange of all types of knowledge. It must provide the full range of learning, cursory to deep. This is a challenging goal but it is also most relevant to wise decision-making by a range of users.

In terms of learning rate, the utility of the tool, or any of its subcomponents, is considered to be greatly diminished if a new user cannot create useful new work within minutes. SMEs who cannot be expected to embrace a system that requires both significant changes in practice and a notable amount of effort to learn before it produces valuable results.

All ideal technologies are transparent to the user. This is a general direction, not a requirement, since it is very difficult to achieve and it may not be absolutely necessary for a comparatively good tool.

4.2.2. Computational Fluid Dynamics (CFD)

The following is based on the experiences of some of Eng's colleagues in learning to use CFD tools in aerospace research projects. The subject of these experiences is Fluent [10] although similar experiences are expected with other, similar software. It is also important to note that this software is being measured in terms of its ability to handle a CFD problem, not PLM. It is a "transplant" from another field.

A CFD program's interface can have defaults and instructions incorporating knowledge on how to best achieve a solution (articulatable). The algorithm itself has elements that streamline computation without significantly affecting results. Generally, CFD can reveal all the fluid mechanical effects on the system. As a computerised tool, by definition, anything it handles can be captured (up to narrative).

There is no structurally perfect method for all finite element analyses nor are their models complete. Inputs and results always need some "tweaking" and interpreting to insure precision, so it stops short of being algorithmic because most real problems require some manual adjustments.

CFD represents the most complete model that humans can make of a system, so it has the potential for deep learning. This is especially true when the models successfully demonstrate the intricacies of real systems. However, simplifications to the models due to computational limits mean that a moderate understanding is more likely achieved. Cursory learning may be difficult because of the involved nature of fluid mechanics.

In terms of learning, CFD users require time to become proficient. Learning can take months.

Given the right software, it is conceivable that data can be integrated with other applications. This includes

importing CAD files and exporting force data to structural modeling software. This is the ideal situation but in the real world, there is usually some kind of problem when converting file formats between different manufacturers and file schemata.

4.2.3. UGS TeamCenter

This PLM package integrates social network building tools with CAD software and a great deal more [9].

This tool, being based on existing standard document types, seems restricted to describing narrative "capturable" and articulatable knowledge.

The document management system is highly algorithmic, enforcing appropriate input and outputs in all areas and fully structuring the data according to time of change, user privileges, etc... It does, however, have a free-form commenting function for sketching on CAD documents that adds to its flexibility. This is an example where the evaluated tool may not occupy a continuum on the rugplot.

This tool appears to provide cursory to moderate learning. The multi-user nature of the tool allows many viewers but it is restricted from "deep" understanding because it does not clearly articulate functional connections from requirement specification through to manufacture and during troubleshooting.

Learning this software does not need to be complicated, a quick use of some parts can be done in minutes. It is expected, however, that full understanding of a CAD or project management module by a user with sufficient background could take a week or more.

This package is fairly cohesive. It reaches the "integrated" level if a company uses all UGS or UGS-compatible applications in its daily activities.

4.2.4. IHMC CmapTools

This tool for generating concept maps [13] occupies a large range of space because concept maps are uniquely flexible, simple tools. Larger maps, however, become bulky to handle and there are limited auto-layout options in this program.

As a network editing tool, it can handle network capturable through articulatable ranges of knowledge.

It has minimal enforced structure, providing a useful sketchpad for free-form mapping or grouped layout. There is no set way of doing things that gives best results but larger maps are more useful with some refined layout. The use of concept maps is probably best described as having refinement practices.

It is conceivable that a sufficiently large map could completely describe any field and all its

interconnections.

The program is simple; most of the key functions can be learned in five minutes. More time is needed, however, to learn to think in terms of maps, which in the authors' experience takes about half a day.

Integrated drag and drop hyperlinking mean that this tool is has a "compiling" level of cohesion. One could easily use CmapTools to generate a web-based report or portal describing a whole product.

4.2.5. Dassault Systemes 3D PLM

This company's PLM software system [12] is similar in scope to UGS's PLM system in that it integrates a wide range of element such as CAD and project management.

The tool is also based on existing standard document types but includes a module for 'VPM navigation' [13] which seem to reflect network thinking at the levels of product function, logic and structure. This infers some "network capturable" knowledge functionality.

The system is highly structured. It is not clear if it has a commenting function. It does have the VPM function which may be more structured.

Gains in understanding may be cursory to deep. Many views for any category of user and descriptions of product in terms of function from the beginning of design through manufacture allow for these possibilities.

Learning speed is similar to that of TeamCenter. A quick view of some parts can be done in five minutes. It is expected that full understanding of a CAD or project management module could take a week if the user has the appropriate theoretical background.

Again, as with UGS, the overall system would seem to have an "integrated" level of cohesion if a company uses all Dassault PLM software.

4.2.6. Matrix10

Since this study was originally written, Matrix 10 has been merged [14] with the Dassault Systemes product. It seems to have retained most of its original functions [15] so the information is left as it was.

The file system is based on standard document types, seems restricted to describing narrative capturable and articulateable knowledge.

The overall system is highly algorithmic, enforcing appropriate input/outputs in all areas and fully structuring the data according to time of change, user privileges etc., but unlike UGS it would not appear to have a free-form commenting tool.

Richness of understanding probably ranges from cursory to moderate. It is restricted from "deep" understanding because it does not clearly articulate

functional connections from design to manufacture or during troubleshooting. It has a "Materials Compliance Central" module [16] but it is unclear what parts of the design are addressed by it. External legal constraints on materials are only part of the design.

As with previous wide-ranging solutions, Matrix 10 seems to provide a quick view and standard windows interface with some parts learnable in five minutes. Since it only connects with other complex packages (such as CAD), the learning curve associated with those components is not considered here. If a company uses compatible applications, users will have to use many other interfaces to access all PLM functions so it is merely "connected."

4.2.7. MS Windows XP

Without realizing it, the average small company probably does most of their PLM through the standard operating system on their computer. It is considered here in a minimal "out of the box" configuration with only WordPad, Internet Explorer, Outlook Express, etc. This system can handle any type of basic file management but fails to provide the database functions of commercial tools.

The knowledge handling range is narrative to articulatable. It will edit pictures, sketches, sound and text files. Its structure could most accurately be described as having refinement practices. System files aside, there are no set ways to organize a user's files. Whether they get lost, duplicated or outdated is entirely dependent on the user.

Windows enables a cursory to shallow level of understanding in the PLM context. The file system will organize hierarchically but it would be difficult to make windows shortcuts do the work of a database.

In terms of learning, the Windows GUI is fairly straightforward and consistent. It profits from ubiquitous use so learning is usually already done. It is the applications setup and system troubleshooting that can be tricky but generally do not require significant amounts of thinking.

The Windows system has a "connected" cohesion. Each file type launches a separate application that uses the same general interface with many small specific changes. Integration of PLM data parts beyond putting files in the same directory is done in the user's mind. Object linking and embedding allows some cross-referencing.

4.3. *Selection of Dimensions*

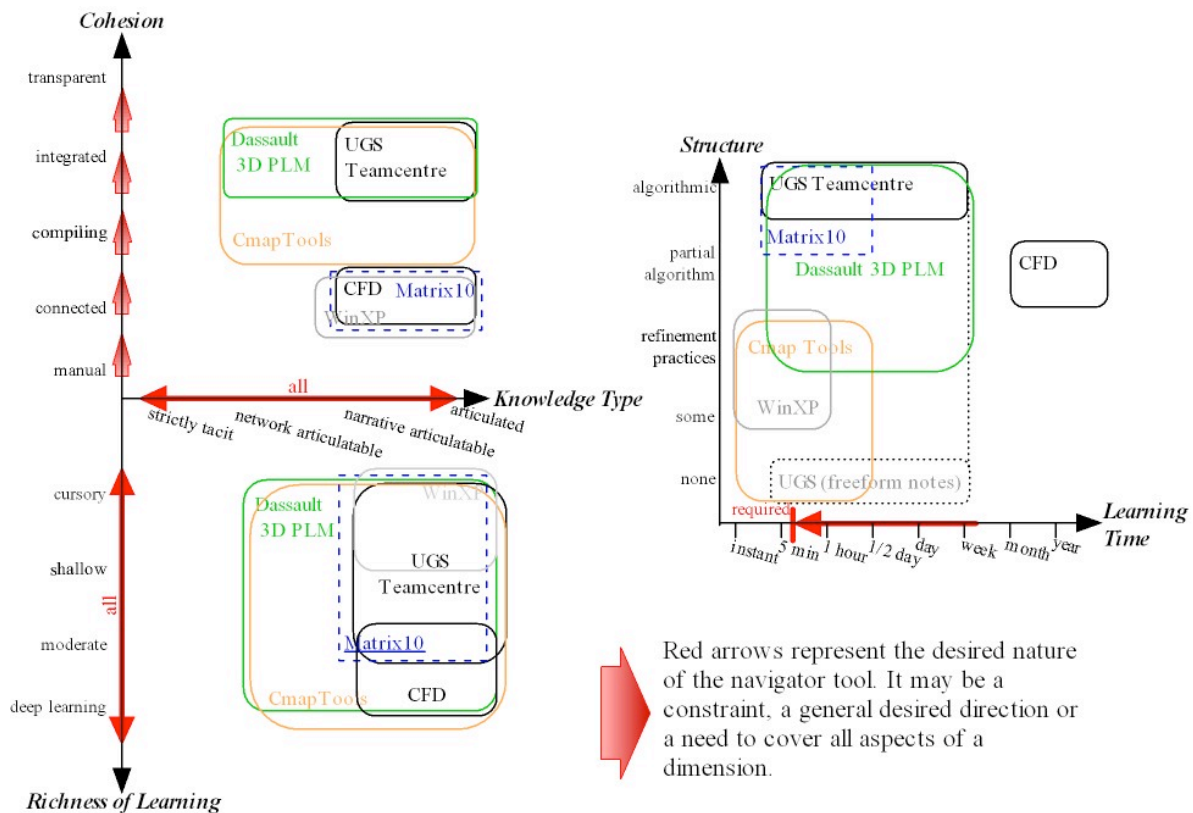
The preceding discussion neglected to mention distinctions in computerization and automation.

Referring to the computerization row of Table 1, it appears that only concept maps varied as to the necessary level of computerization. All of the other tools required it.

While there was some minimal distinction among the analyzed tools, there was no target direction for the idea tool. This dimension was thus omitted. See Table 1 for the full summary.

Table 1. Tabulated results of the PLM tools analysis

Dimension	Desired Tool	Computational Fluid Dynamics	UGS Teamcentre PLM	CMap Tools	Dassault Systems 3D PLM	Matrix One	Windows XP
Knowledge description	ALL	Narrative to Articulatable	Narrative to Articulatable	Network to Articulatable	Network to articulatable	Narrative to Articulatable	Narrative to Articulatable
Computerization & Automation	Any	Required	Required	Manual to Total	Required	Required	Required
Structure	any	partial algorithm	None AND algorithmic	None to Refinement practices	Refinement Practices to algorithmic	Algorithmic	Some to Refinement Practices
Richness	Cursory to Deep Learning	Moderate to deep learning	Cursory to Moderate	Cursory to Deep Learning	Cursory to Deep Learning	Cursory to Moderate	Cursory to Shallow
Learning Simplicity Speed/ Complexity	5min to 1h	year	5 minute to 1 week	5 minute half-day	5 minute to 1 week	5 minute to half-day	5 minute to 1 hour
Cohesion	Transparent	Connected	Integrated	Compiling	Integrated	Connected	Connected



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

Figure 1. Rugplot results for PLM example

4.4. Plotting Results

Using the data from Table 1, one can now construct a visual using any vector-based drawing program. A vector-based program is recommended because this makes rearrangement easier. A fair amount of learning comes from experimenting with the graph. The authors used SmartDraw 7 to generate the graphic in Figure 1.

4.5. PLM Results Analysis

The densest area of the resulting graphic is the “cursory, articulated” region where five out of six tools overlap. This is probably due to the relative simplicity of fulfilling those extremes of the individual axes more than a special nature of that point combination.

The sparsest area is any point in the “strictly tacit” zone. This is probably due to the dataset. Types of knowledge usually considered strictly tacit can be manual crafts or social skills. All the tested tools were computer based so their use here is limited. This conclusion highlights the need for non-computer elements in a PLM system. If a compromised dataset were not the case, the map would clearly highlight a set of requirements missing from the existing solutions.

There is a light correlation between the structure and learning time axes. Given a few more samples, one could infer that increasing structure (and thus predictability) also forces users to think a certain way. Free-form tools let users think their own way and get using a tool more quickly.

Richness seems slightly correlated to knowledge dimensions. This may be because of an assumption about networked representation being more suitable for deep understanding.

In terms of overall complexity, the single simplest tool of the set is CmapTools. It is not a database nor does it automate any processes. It is, however, covering more of the desired problem space than any other tool. This flexibility provides a good rationale for basing the OPLM navigator’s initial development on concept maps. It represents combinations of functionalities that are generally missing from the current toolset.

5. Closing Remarks

This rugplot is a simple combination of ideas but one can begin to get a useful perspective on an early design problem, even with slightly naïve input information. It provides a framework for fine articulation of the design elements even when the problem itself is only partially understood. Mistakes along the process, like poor axis definition or bad requirements identification are highlighted by the end.

These focus efforts for further work.

Further work on the rugplot would involve the automation of the graph generation process from the results table. This could be done through a spreadsheet template or database plug-in to integrate it with existing tools. This would automate much of the “book keeping” involved in the generation of the graph.

A larger number of data points would also help to see patterns more clearly. This is only really feasible with that added automation.

More layers of data would also add to the design of the visual. One such layer is a qualitative assessment of how well each concept meets the requirements. Using a colour or shade gradient within each data rectangle, a general sense of poorly met requirements might also surface.

Developing beyond all of this, a larger scale database could be maintained to keep a continuous sense of all the tools in a given field or to track the performance on a project with respect to a set of evolving requirements. It would speed access to the many layers of thinking and annotation involved in the results table’s creation. This would mean that more time could be spent on experimenting and looking for correlations.

Finally, for future studies, it would be useful to have a nomenclature for the different kinds of gaps and the different dimensional combinations that form them.

6. References

- [1] “IBM Deep Computing Visualization Solutions for Product Lifecycle Management”. document DCS00970-USEN-00. http://www-03.ibm.com/servers/deepcomputing/visualization/downloads/dcv_solutions_for_plm.pdf. Retrieved August 2005.
- [2] Tufte, Edward R., *The Visual Display of Quantitative Information*. 2nd ed. Graphics Press, Cheshire, Connecticut, 2001.
- [3] Tufte, Edward R., *Envisioning Information*. Graphics Press, Cheshire, Connecticut, 1990.
- [4] Tufte, Edward R., *Visual Explanations: Images, Quantities, Evidence and Narrative*. Graphics Press, Cheshire, Connecticut, 1997.
- [5] *Ibid.* p.29.
- [6] *Ibid.* p.80 – 81.
- [7] Davenport, T. H. and L. Prusak, *Working Knowledge: How Organizations Manage What They Know*, Harvard Business School Press, Boston, 1998, p.70.
- [8] Kelley, Tom and Jonathan Littman, *The Art of Innovation*. Currency Books, New York, 2001. p.59.
- [9] “UGS: Product Lifecycle Management (PLM) Solutions” <http://www.ugs.com/index.shtml>, Accessed May 2006.
- [10] “CFD Flow Modeling Software & Solutions from Fluent” <http://www.fluent.com>, accessed May 2006.
- [11] “CmapTools Homepage”. <http://cmap.ihmc.us>, accessed May 2006.

[12] "Dassault Systemes: Home", <http://www.3ds.com/home> accessed September 2006.

[13] "VPDM White Paper May 2004", http://www.3ds.com/uploads/tx_user3dsplmxml/VPDM_White_Paper_May2004.pdf, p. 6. accessed May 2006.

[14] "Dassault Systemes: Press Room", <http://www.3ds.com/news-events/press-room/release/1178/1/>, Accessed May 2006.

[15] "ENOVIA MatrixOne Products", <http://www.3ds.com/products-solutions/plm-solutions/enovia-matrixone/overview/>, Accessed May 2006.

[16] "MatrixOne: PLM Solutions: MatrixOne Materials Compliance Central", <http://www.matrixone.com/matrixonesolutions/materialscompliancecentral.html>, Accessed May 2006.

ACKNOWLEDGEMENTS

The authors graciously acknowledge the support of the Auto21 NCE and the Ontario Centres of Excellence for the work presented here.