UNIVERSITY OF GOTHENBURG

3D Visualization of Complex Software Models: Practical use and Limitations

Joacim Alvergren
Jonatan Granqvist

Bachelor of Software Engineering and Management Thesis

Report No. 2010:019
ISSN: 1651-4769

University of Gothenburg
Department of Applied Information Technology
Gothenburg, Sweden, May 2009

3D Visualization of Complex Software Models: Practical Use and Limitations

Joacim Alvergren[*]
IT University of Gothenburg

Jonatan Granqvist[†]
IT University of Gothenburg

Abstract

Model Driven Development is becoming a popular methodology when developing large software systems. 3D visualizations has been proposed as an aid for the developers trying to comprehend large amounts of complex diagrams present in model driven development. But how effective is it in practice, and how will it scale when applied on large complex system? The purpose of this study is to validate if 3D visualization increases comprehensibility of large, complex diagrams and to explore what limitations there are to the technique. A tool that can visualize diagrams was constructed and validated through a technical validation and a qualitative interview study at Ericsson AB. The results show that 3D aids the user to get a better understanding of the structure and component relationships of the diagrams, something that is a highly complex task when using the present 2D tools. The 3D visualization makes it more intuitive for the user to see how different parts in the model relate to each other. The study also shows that 3D has potential to be used as an alternative to the file tree-browser present in most modeling tools: the user can find diagram parts easier, see interconnections and navigate faster between different sections of the model. Several limitations that are needed to be solved before the technique can be put in to use in production were identified and these include issues relating to information overload and problems distinguishing relationships between hierarchies.

Keywords: 3D Visualization, UML, RSART, Model Comprehension, Software Visualization, MDD

1 Introduction

Model driven development (MDD) is a software development methodology where models represented as visualized abstractions are used to develop software systems. This methodology is normally applied so that the developers create diagrams using the Unified Modeling Language (UML) with connoted model data and then a modeling tool generates parts or all code for the software system. MDD combines the software design and software implementation activities and aims to increase productivity and make the software more comprehensible (Selic 2003).

However, with ever larger and more complex software systems the present visualization of the model diagrams have been found to have problems with comprehensibility. When developing large software systems it is vital to have a mental model: a comprehension of the structure of the system's different components and their relations to each other. Working with the complex hierarchical models present in MDD, gaining such a mental model has become a highly challenging task (Krolovitsch & Nilsson 2009).

The way the present MDD modeling tools allow for the user to comprehend the structure is by traversing between the different components in the model represented by 2-dimensional diagrams. This is a complex task that demands memorizing large parts of the model to gain an understanding of the hierarchic structure and that puts a high cognitive load on the user trying to comprehend the model.

To increase the comprehensibility and reduce the cognitive load of the user, a suggested solution has been to also visualize the models in 3-dimensional diagrams (McIntosh et al. 2005, von Pilgrim & Duske 2008, Krolovitsch & Nilsson 2009). The increased spatial expressiveness in a 3D visualization compared to 2D allows the user to view and explore large amounts of information at once and could therefore increase the comprehensibility of the model, something previous studies in this area have found evidence for (Lange & Chaudron 2007).

However the studies that have made functional prototypes in this area have not validated the prototypes against a large real-world industry model. Testing the technique on a large complex model naturally increases the complexity of the visualization and could expose challenges, weaknesses and limitations not visible in small models.

The purpose of this paper is to test if the ideas of increased comprehensibility using 3D visualization can scale to a large, complex, real-life software model and uncover what limitations there are. It will also investigate how a tool can be constructed to automatically generate an aesthetic 3D visualization of hierarchical UML models.

2 Research Approach

To be able to validate if 3D visualization indeed increases comprehension of large, complex models we have created a

[*]e-mail: joacim.alvergren@gmail.com
[†]e-mail: jgranqvist@gmail.com

prototype of a visualization tool that visualizes IBM Rational Software Architect RT (RSART 2010) structure diagrams in 3D. The development was structured following the design science methodology, which is characterized by its problem solving nature. In theoretical research, for example quantitative studies, the researcher studies an artifact that already exist in the world, analyzes it, and tries to understand and explain it (Creswell 2008). Design science, on the other hand, is used when there is a problem that is observed in a specific environment and there is no apparent solution designs for solving it (Holmstrom et al. 2009). It is the researcher's objective to create the new and innovative artifact that solves the problem, and to move from the present state to the desired state in the best possible way. What further differentiates design science from theoretical research is how the concept of truth is valued. In for example positivist research, the researcher set out to unearth an objective truth using large amount of data that is processed in a systematic way. On the other hand, in pragmatic research — which design science is a descendant of — truth is measured by what works (Creswell 2008, Hevner et al. 2004).

As a part of our scientific contribution we have validated the artifact and generalized the knowledge gained. The method used for validating our artifact was a semi-structured interview with an employee at Ericsson AB. The test person's feedback has subsequently been a basis for our evaluation of the prototype and for answering the research question. The knowledge gained was then generalized by synthesizing the results from the interview with related research and the experiences from developing the prototype.

2.1 Process

The research was divided into six steps (see Figure 1). We started with a literature review to have a basis for the development. The next step was rapid prototyping of our tool so it could provide basic functionality. The tool was then tested during a technical validation and refined during the following step. During the fifth step we validated the functionality of the prototype in regards to the research question. In the final step we analyzed our experiences and the data from the validation.

The user tests has been conducted at Ericsson AB at a department involved with development of communications technology. The department is practicing model driven development on a large scale in complex projects.

2.1.1 Literature Review

To gain an understanding of the field of 3D visualization we first did an extensive literature review. The aim was to see how the problem with visualization of 3D models had been approached before, both theoretically and practically.

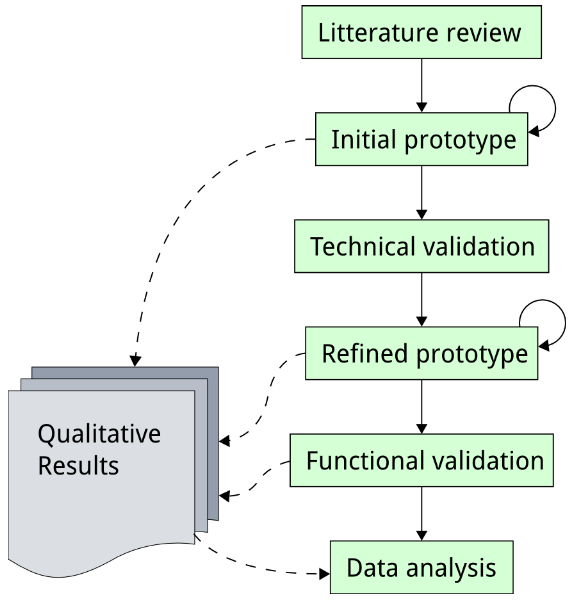

Figure 1: Research process

2.1.2 Initial Prototype

Based on the findings of the literature review, a prototype of our 3D visualization tool was constructed. The prototype was developed during several iterations, starting with basic functionality and then gradually adding new features. Notes about experiences and design decisions were taken during the development.

2.1.3 Technical Validation

A technical validation of the prototype was conducted to test if the technological solution was sufficient when using large and complex models. The test was done on location at Ericsson AB using a production model.

2.1.4 Refined Prototype

The prototype was refined to accommodate the limitations found during the technical validation.

2.1.5 Functional Validation

Thereafter, a functionality validation of the revised prototype was conducted with an Ericsson employee that has several years of experience working with model driven development. The data from the validation was collected using a qualitative, semi-structured interview and by doing observations (Creswell 2008). The interview was performed using open-ended questions following an interview protocol, with the possibility for additional questions when needed. Observations were made on the informant's usage of the tool and his reactions.

2.1.6 Analysis

During the last phase, the interview material was analyzed. The aim of the data analysis was to elicit the participant's opinions about the tested tool and to find any differences in model comprehension compared to their normal way of working. The answers from the interview were transcribed and then organized for analysis. Common themes were identified and interpreted.

Then, this data was manually analyzed with an aim to classify the opinions of the participant. This was done by trying to identify and isolate opinions and then grouping them to be able to generalize and draw conclusions.

3 Result

In this section, data collected during the development and validation of our prototype is presented. First, the technical aspects and design decisions relating to the prototype are presented. That section is then followed with the results from the validations.

3.1 Prototype

The intention of the prototype developed in this study is not to provide a replacement of the standard 2D view that is present in the modeling tools today but instead it is intended as a complement: editing and analysis of details in the model are still done in the traditional 2D view while the 3D view is supposed to provide overview and understanding of the structure and relationships among components in the model.

The UML diagram standard is designed to be visualized in 2D, so the tool needs to transpose the components of the diagram into a 3D representation. This is done so that components in the diagram are represented by 3-dimensional boxes positioned in a 3D environment where the nesting hierarchies in the model are shown as offset in the z-level (top to bottom) and the interrelationships between components are shown with line-connections between them, see Figure 2. To keep the visual coherence between the 2D and 3D visualization, the tool inherits the layout of some of the components from the standard 2D view such as the relative sizes of the boxes.

The diagrams are presented in an interactive 3D environment where the user can navigate inside the model using the type of navigation common in most 3D CAD software, i.e. zooming, rotating and panning. The components in the diagram are selectable with mouse clicking and allow for interaction with the model. The prototype also implements search functionality where components in the model can be searched by their name and matching components are highlighted in the 3D-view.

The tool retrieves the UML model data from the modeling tool using Eclipse Graphical Modeling Framework API. The 2D UML data retrieved from the modeling tool is then passed to a layout algorithm that calculates the new positions of the individual diagram components in the 3D environment. This layout is then rendered using a 3D engine framework. The 3D engine used in this prototype is Panda 3D — a free open source engine mainly used for game development.

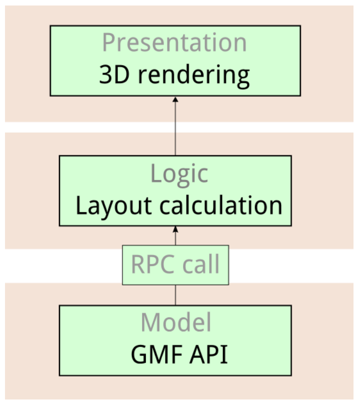

Figure 4: Simple architectural view

The first iteration of the prototype retrieved the model data by reading the model's XMI file, wich is basically UML in XML format. This way of retrieving model data was deemed to provide inadequate performance when working with very large models. Instead the model data is retrieved using a connection to the modeling tool's API — this provides a more efficient way of gathering and updating the model data. Using the API also allows for better integration into the modeling tool which makes programmatic interaction between the 3D view and the modeling tool easier.

3.2 Technical validation

For the final validation, the prototype should be able to load and visualize a large and complex model. However, during the development of the prototype we had limited access to such models. So, when we had a version that worked with our smaller test models we performed a technical validation of the functionality at the Ericsson labs. It was done using a large and complex model used in production at Ericsson.

The test revealed two limiting factors when it comes to visualizing large models in 3D that was very valuable for the further development of the prototype. Firstly, the load time of a very large model is considerable so a closer integration with the IDE Eclipse was needed. That way, the visualization tool could read information already loaded in memory thus loading faster. Secondly, differences between model file structures made it hard to be generic and support different kinds of diagrams. This was also solved by the integration with the IDE.

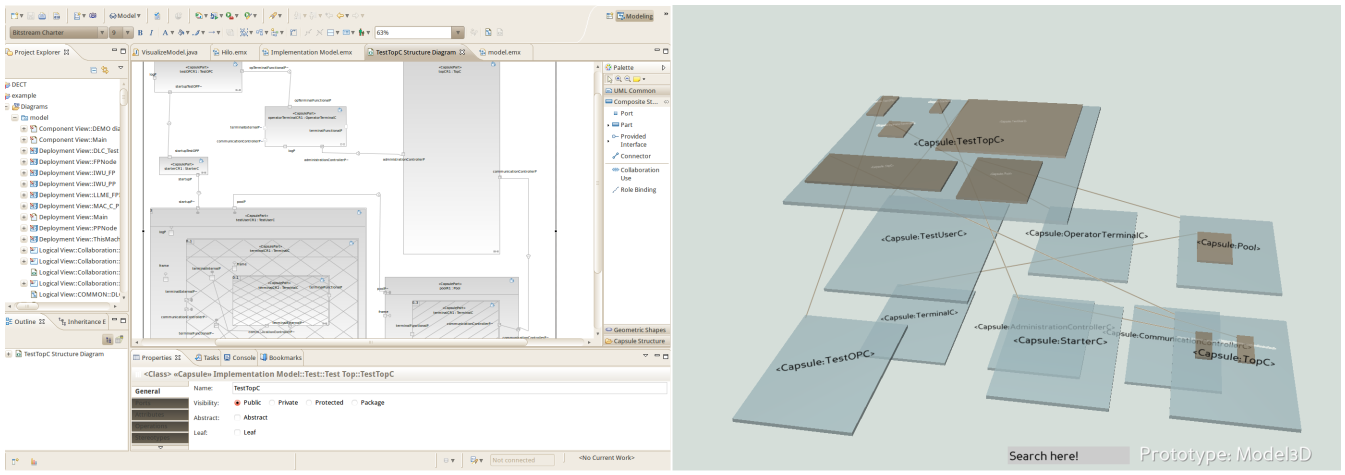

Figure 2: Comparison between 2D and 3D view

3.3 User validation

During the user validation we identified several limiting factors of the 3D visualization technique in software modeling of large complex systems — and what steps are needed to take before the technique can be truly useful in everyday practice. We also identified in what areas of practice the technique have the biggest potential to have a positive impact.

3.3.1 Visualization of abstract information

The 3D visualization is seen as something that is useful mainly to people working with development. The use for non-technical people that are involved in the development process is seen as limited but could provide a visual experience of the size of the system that is otherwise hard to convey. Seeing something as a picture is seen as generally positive and the informant gives an example relating to the DNA-spiral: people had been working with DNA for years and knew it was there but until someone made a visual representation of it people did not share an image of it. The same is certainly true about something as abstract as software, but it also poses a challenge as there is no natural link between the 3D visualization and the artifact that is created. Software do not have a shape, form or color compared to for instance hardware modeling where the link between what you visualize and the artifact you create are much more natural.

Since many people are acquainted to the UML 2D diagrams, switching to see the software visualized as a 3D image would naturally take some habituation. An important step to simplify this and to avoid confusion between the 2D and 3D visualization is to minimize the need to switch between the different types of visualization as the switching between views would put a strain on the user having to readjust his thinking to fit with the visualization type.

3.3.2 Information overload

One major limiting factor for the scalability of visualization of complex models identified in the user validation is information overload. When trying to interpret complex structures, getting too much details presented at once severely affects the comprehensibility. The informant expressed that the process for the investigation of a complex model should be to go from a simple overview with limited detail level. Then from this view identify the section of the model which is of interest depending on the task and then focus on this subset and build up a view of the model with the specific details needed.

The way the informant currently is working, the only possibility to get an overview of the model is to traverse between many fully detailed views showing small components of the model. These views provides information that are important when doing detailed implementation work in a specific part of the model but become too expressive when the user is trying to get a grip of structure and to assess how changes in one part of the system will impact other parts of the system. As a complement to these detailed diagrams a 3D view provides substantial practical improvement to the way of working with large complex models. Both being able adjust the level of detail shown and to have the ability to select and visualize a specific subset of the model is seen as important to avoid the negative effects of information overload.

3.3.3 Visual coherence

Having high visual coherence between the different visualization views of the model is expressed as very important by our informant. 2D and 3D views should share the same visual representation to make it easy for the user to connect an object in one view to the other. This is true within the 3D view as well. If one object exist is several places, one should be able to get a natural connection between them. Visual

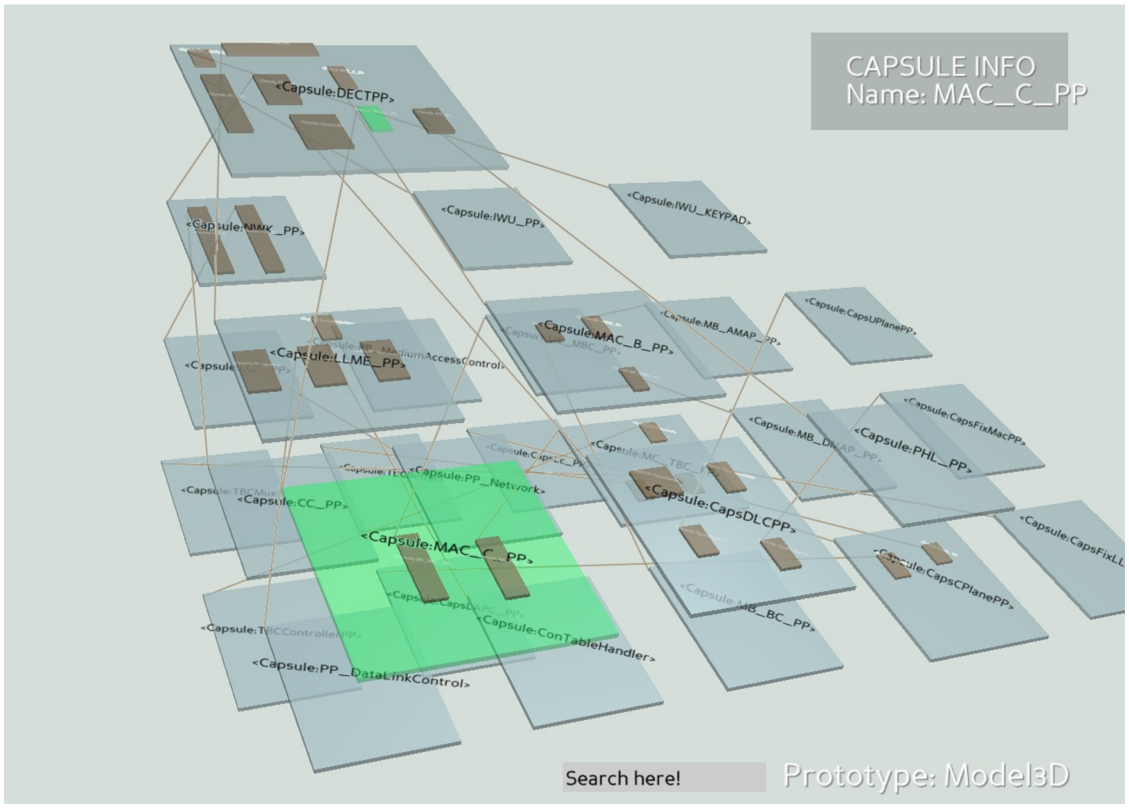

Figure 3: Structure diagram in 3D view

traits like for example color, position, shape and relative size can be used for this.

3.3.4 Dynamic documentation

One of the goals of MDD is to have "everything in the model", meaning that the goal should be to avoid static documentation that needs to be updated and instead have a dynamic information source. However, since the present modeling tools do not provide diagrams that give a full overview of a models structure, many individual developers make manual drawings, normally with pen and paper, to build up their own understanding of the system. The 3D view is seen as a potential replacement for this and since the 3D view is visualized directly from the model source you could replace a static documentation with a dynamic one.

The 3D view could also be used as a better way of conveying information between different developers. The informant gives an example of when a developer needs to communicate with another developer about how a certain change in the system would impact another part. Instead of trying to explain this with text referring to a several diagrams in the model, the developer can convey this information using a 3D view that is containing a larger part of the model; with a highlighted comment embedded in the 3D view the receiving developer would get a more efficient way to perceive wich part of the model the issue is regarding.

3.3.5 Discovering design flaws

A potential practical use of 3D visualization expressed by our informant is to identify design flaws in the system. This could for example be to be able to highlight different components in the model that share similarities. This would allow for analysis of the model for bad design choices or find that major changes in an old system have created the need for a re-design.

4 Discussion

In this section the results from the validations and the prototype development will be discussed in relation to previous research.

4.1 3D game engine

A 3D Game engine is a middle-ware used in game development. In the beginning of game development history, games were made as isolated entities that were optimized to use the limited display hardware. But, with more advanced games and better display hardware the idea of reusing a general platform of code used to create games came to life. Choosing to use a game engine to create the 3D visualization tool was a deliberate choice stemming simply from that the game industry has lead the development of advanced 3D graphics for many years. To use a tested, optimized and stabilized code base for the rendering of the visualization also seemed as a good way not just to allow for rapid prototyping but also in regards to performance which was identified as a critical issue when wanting to visualize large complex structures in 3D (von Pilgrim & Duske 2008).

We chose the Panda3D engine (Goslin & Mine 2004) as this was a powerful open-source engine that is portable to all major platforms (Microsoft Windows, Linux, Mac OS X, FreeBSD). Apart from the 3D rendering the engine also provides many other capabilities such as collision detection, mouse and keyboard support, performance analysis etc. The engine is written in C++ but provides a Python wrapper that exposes the full functionality, and this was seen as positive as it allowed for a rapid prototyping.

4.2 Layout

One of the purposes of complementing 2D visualization with 3D is to be able to view more information in one view. This emphasizes an effective presentation of that information. So when choosing how to layout the 3D diagrams there is a prioritization that is needed between displaying as much of the wanted information as possible and at same time provide an aesthetic, simplistic and intuitive presentation. This is performed by the layout algorithm that calculates the position of the components of the diagram.

When visualizing large structures having a layout that promotes intuitive interpretation is crucial. The current layout strategy aims to create balanced square patterns positioned at the model's different hierarchy levels. To create a clearer presentation of the model it should have a layout algorithm that takes the connections between different levels in the hierarchy in to account when calculating the positions to minimize intersection between connectors and create a simpler and a more synoptic presentation.

4.3 Search and highlighting

Searching in 3D is very efficient as the results of the query can be visualized in the 3D environment, for instance by highlighting affected objects. In Rational Software Architect RT it is common to use objects that is shared and used in several places. In a large diagram it can be hard to find all places where a particular object is used without a lot of manual labor. In a 3D environment all instances of the object can be highlighted at the same time, giving the user an immediate response. For example, one could highlight all places where a certain protocol is used.

4.4 Integration

As noted during the user test, our prototype suffered from being external to the development tool. The user had to switch between the editing view and the visualization view and that forced him to readjust his perception of the diagram every time and this is an unnecessary strain of the mind. The aim should be to allow the user to do as much as possible while in the 3D view, thus removing much of the need for switching between different mental models of the same diagram. This could include smaller editing tasks and adding and removing objects. However the 2D view is still important for more detailed tasks like connecting ports.

4.5 Visual metaphors

When designing a visualization tool one must be confident that your visuals convey correct and understandable information and not at the same time overwhelm the user. An object in a visualization is basically a metaphor for something else and is often carrying an added meaning in itself. In 2D, the visual metaphor is limited to features like size, pattern, color and placement on the plane etc. But, when transformed into 3D, the semantic richness broadens enabling effects like objects having different elevation and transparency (Marcus et al. 2003).

In our prototype we plot different levels of hierarchies on planes on top of each other and use the visual metaphor of lines to show the relationships between them. This way of visualizing worked well on smaller models but when used on large models the visual effectiveness was limited; too much and too detailed information at the same time made it impossible to distinguish the individual parts.

One proposed solution to this problem is to use a more advanced filtering mechanism where the user initially view the model without any detail, just showing the general structure. The user can then select the part of interest and the view changes to show just that part, but with more detail. Basically there will be two modes, one for high level navigation and one for detailed work. This way, the user can get a mental model of the whole structure before going into detail. By having this mental model it is easier to grasp the interrelationships between the parts. This is different from just seeing the diagram in 2D making it much harder to get a mental picture of the structure.

Another solution is to use a different way of showing relationships between hierarchies, for example color coding. The objects on a plane would be filled with different colors making them stand out from each other. On the plane lower in the hierarchy, the objects that are descendants of an object higher up would have the same color as its parent. That way one could easily follow which object belonged to each other, just by its color.

4.6 3D replacing tree-browser

One of the major limiting factors of MDD modeling tools today is the navigation within the model: not being able to see a picture of the structure when navigating in the model it becomes cumbersome to find the way parts fit together. In the present tools the only overview one may get is a tree-browser view of the components in the model. When working with large complex models this view becomes very cluttered and hard to comprehend and navigate.

A 3D visualization is a potential replacement for this view, providing a simple overview that clearly show different components positions and relations in the model. This would give a more intuitive way of navigating a large model allowing quick overview and access to all parts of the model in one view. This would be very useful especially for someone trying to navigate in a unknown model or in a unfamiliar part of a model. The possibility to get an overview of the relations provides a more natural way to comprehend the model compared to the tree-browser's list of cryptic names that is normally used to name components of the model.

5 Related Work

The idea of displaying UML diagrams in 3D is well-established and there exist a common ground when it comes to notation and shapes of the objects (Gil & Kent 1998).

Research in the area of model comprehension have proposed the idea of using 3D visualization as a means of making comprehension easier. Koike (1993) showed that 3D allows the user to view several diagrams in one view thus allowing them to use the 3D model as a basis for their mental model. The usual way to display highly complex systems is to disperse the complexity in a multi-window system, but this solution can become confusing and counterproductive. Multi-window systems only allows the user to get a mental model of the separate diagrams – not the interrelationships among all the components of the models (Koike 1993).

In a case study performed at Ericsson AB in 2009 it was shown that comprehension of complex software models is problematic when working with model driven development (Krolovitsch & Nilsson 2009). In interviews with the developers at Ericsson, the interviewees expressed problems understanding new models and quickly learning a model's structure. They also had difficulties navigating inside the models and understanding the diagrams within. A static 3D representation of a hierarchical model was created based on the developers feedback and it was evaluated by a qualitative study. The study concluded that 3D visualization definitely have great possibilities to increase model comprehensibility when working with model driven development. The gains using 3D was also verified in a study conducted by McIntosh et al. (2008) where he tested whether comprehension of state machines could be increased using a 3D environment. The approach was tested quantitatively by letting users perform certain tasks with state machine diagrams while measuring the time it took. Some of the tests were made on state machine diagrams used in production, but the complexity was low (McIntosh et al. 2008).

Another 3D visualization tool is GEF3D by von Pilgrim & Duske (2008). GEF3D is an extension to the Eclipse framework that extends the standard editing framework seamlessly with a 3D equivalent. In the paper, the tool is tested with Topcased UML editor, but the focus is on implementation rather than comprehension.

6 Conclusion

The goal of this paper was to test the possibilities of automatically creating 3D visualizations of large, complex and hierarchical UML diagrams, and to find factors limiting the comprehensibility and scalability of the technique. To accomplish this, we created a prototype of a visualization tool based on best practices found in previous research. Then a technical and a user based validation was performed using a large, complex production model.

The results from the creation and validation of the prototype showed that 3D visualization of UML diagrams is very promising, especially when working with complex models. The study shows that getting a 3-dimensional visual experience of a model can be a great help understanding the structure of something that otherwise is highly abstract. A very complex diagram is inherently hard for the user to comprehend and visualize mentally, but when viewed in 3D one can more easily get a good overview of the system compared to 2D. When viewing the diagram in 3D it is also easier and faster to understand how parts relate to each other and how the parts a dispersed.

Another finding is how important it is with visual coherence in the visualization. By inheriting the same visual style as in the 2D editor the size and placing of the elements in relation to each other is important to make it easier to adapt the mindset to a new way of viewing the model.

The study also found several factors that limits the comprehensibility of 3D visualization. On one hand, it can be helpful to see a large diagram in its entirety. But, on the other hand, seeing everything at once limits the comprehensibility of the information displayed: it is simply too much information to comprehend at the same time.

Another limiting factor is the need for switching between different views. For 3D to be efficient one must be able to do the work needed in the 3D view in one session and avoid being forced to switch between two representations of the same entity. That would force the user to readjust the way of thinking and put unnecessary cognitive load on the users mind.

The validation also showed that when a diagram is very large, the effectiveness of having lines to show hierarchical relations becomes limited. This is something that needs to be solved before the technique can become useful in practice.

7 Future work

Following the work in this paper it is shown that 3D visualization is a technique that have potential to be used on large complex MDD projects. However, there are some areas of the technique that needs further exploration before it can be put in to real production use.

One area that needs further exploration is the way to handle the dynamic selection of subsets of a large model and how to be able to adjust the detail level in a intuitive way that is non intrusive and seamless for the user.

Another area in need for more exploration is alternative ways of displaying hierarchies in ways that scale better than using line connectors, e.g. color coding, to avoid information overload. This could also be improved by developing a better layout algorithm to handle interrelationships between hierarchies in a more intelligent way, avoiding intersections and accumulation of details in certain areas of the diagrams.

There is also the question of why the 3D technique has not come further in software modeling although the 3D technique has been around for decades. When it comes to hardware modeling, 3D tools have been standard for many years. Although the technical issues is one limiting factor behind this difference in technique adaptation, the authors feel that other reasons such as socio-technical and economic factors may be behind that the software industry have not put this technique into practice.

The use of the 3D technique as a way of analyzing software systems for design flaws in a large, complex system is something that could be really interesting and that needs further exploration.

Acknowledgements

The authors wish to thank their supervisor Lars Pareto for invaluable help and support. Also, they would like to thank Krister Bergh for his technical expertise, feedback, thoughts and ideas and Ericsson AB for graciously providing material for our tests.

References

Creswell, J. (2008), *Research design: Qualitative, quantitative, and mixed methods approaches*, Sage Pubns.

Gil, J. & Kent, S. (1998), Three dimensional software modelling, *in* 'Proceedings of the 20th international conference on Software engineering', IEEE Computer Society, pp. 105–114.

Goslin, M. & Mine, M. R. (2004), 'The Panda3D Graphics Engine', *COMPUTING, E.* pp. 112–114.

Hevner, A., March, S., Park, J. & Ram, S. (2004), 'Design Science in Information Systems Research', *Management information systems quarterly* **28**(1), 75–106.

Holmstrom, J., Ketokivi, M. & Hameri, A. (2009), 'Bridging practice and theory: A design science approach', *Decision Sciences* **40**(1), 65–87.

Koike, H. (1993), 'The role of another spatial dimension in software visualization', *ACM Transactions on Information Systems (TOIS)* **11**(3), 286.

Krolovitsch, A. & Nilsson, L. (2009), '3D Visualization for Model Comprehension A Case Study Conducted at Ericsson AB', *Department of Applied Information Technology 2009: 047* .

Lange, C. & Chaudron, M. (2007), Interactive views to improve the comprehension of UML models-an experimental validation, *in* '15th IEEE International Conference on Program Comprehension, 2007. ICPC'07', pp. 221–230.

Marcus, A., Feng, L. & Maletic, J. (2003), 3D Representations for Software Visualization, *in* 'Proceedings of the 2003 ACM symposium on Software visualization', ACM New York, NY, USA.

McIntosh, P., Hamilton, M. & Schyndel, R. (2008), X3d-uml: 3d uml state machine diagrams, *in* 'MoDELS', Vol. 8, Springer, pp. 264–279.

McIntosh, P., Hamilton, M. & van Schyndel, R. (2005), X3D-UML: enabling advanced UML visualisation through X3D, *in* 'Proceedings of the tenth international conference on 3D Web technology', ACM, p. 142.

RSART (2010), 'IBM - Rational Software Architect', `http://www-01.ibm.com/software/awdtools/architect/swarchitect/`.

Selic, B. (2003), 'The Pragmatics of Model-Driven Development', *IEEE software* **20**(5), 19–25.

von Pilgrim, J. & Duske, K. (2008), Gef3D: a framework for two-, two-and-a-half-, and three-dimensional graphical editors, *in* 'Proceedings of the 4th ACM symposium on Software visualization', ACM, pp. 95–104.