The Power of (Software) Visualisation

Claire Knight and Malcolm Munro
Visualisation Research Group
Research Institute in Software Evolution
Department of Computer Science, University of Durham,
South Road, Durham, DH1 3LE, England.
Department of Computer Science Technical Report 01/00, January 2000
{C.R.Knight, Malcolm.Munro}@durham.ac.uk

ABSTRACT
The use of three dimensional software visualisation techniques has the power to transform the way that knowledge gathering takes place during many software engineering activities. The focus of this work is the understanding of software from a program comprehension perspective and therefore it is the visualisation of software artefacts that is of greatest interest. It is not just the use of three dimensions that can allow the possibility of such power, indeed a badly designed three dimensional visualisation is worse than none at all, but the extra dimension afforded by such visualisations can open a whole new world of possibilities.

Keywords
Software Visualisation, Program Comprehension, Software Engineering

1 INTRODUCTION

It is a well-known and documented fact that software is large and complex and that despite many software engineering advances this is still the case. For many this size and complexity is seen as a good barrier to hide behind with comments such as "inherent in software" and "unvisualisable" thrown out. Whilst software may remain complex, and indeed this may be unavoidable with the demands placed on the systems of today, the hiding behind these sorts of comments is not a good thing for the field. This is for many reasons, and in the recent past a very good example of this was the year 2000 and the many related issues.

The work documented in this paper and the further ideas that stem from it were originally driven, and to some extent still are, by the need to find alternative ways of visualising software. The relationship between graphics and the program comprehension field, indeed the software engineering field, is a long one but it is precisely these fields that have been slow to embrace new and novel techniques. This is despite documented shortcomings in the oft-used representations that centre on two-dimensional graphs. Software visualisation has to overcome many problems, not least in that there is usually a large amount of data to be presented and that those data items are then related in many complex and varied ways. Software is also inherently intangible and this therefore requires that the field consider representational issues, a situation that does not hold true for all visualisations. As an added problem, software systems also evolve over time and this affects the the software engineering visualisation field to the state that the information visualisation one already finds itself in.

2 VISUALISATION

Software Visualisation is a powerful way of viewing the many and varied artefacts that composed together form a

computer system. It also has much power and use in the representations of the complex interrelationships between these artefacts.

Three-dimensional visualisations enable a whole new world of visualisation styles and representations to be explored, and the use of an extra dimension **can** provide (but does not guarantee) powerful visualisations. Proponents of graph based displays (the predominant two dimensional visualisation used in the software engineering field) have been known to discredit three dimensional work purely because of the lack of empirical evidence as to the benefits of these types of visualisation. Whilst such evidence is in short supply in the various areas of software engineering where such techniques are used there is much psychological evidence to be found and reused from studies relating to graphics, colours, navigation and information visualisation.

Visualisation Power

Based on the view of Gelernter [4] the potential success of software visualisation can be characterised by describing a good visualisation as one that has powerful mechanisms to be an intelligence amplification tool combined with simplicity of interaction. In more detail, this can be described by suggesting that having simplicity of interface, interaction and browsing combined with powerful mechanisms for intelligence amplification and knowledge acquisition provides a sound basis for the development of three dimensional visualisations.

This is not to say that intelligence amplification power and a simple interface guarantees a good three-dimensional visualisation (which should have both visual beauty and machine elegance) but that the two components are major players in facilitating such visualisations.

Metaphors

From a visualisation and virtual reality perspective *metaphors* act as a mapping from the concepts or artefacts required to be displayed in the world to their graphical representation. Pettifer and West [6] suggest that the potential power of virtual reality comes from the strength of the metaphors used, and that the use of a three dimensional space enables a form of interaction that is closed to "normality" that with many other forms of computer systems.

The interaction that is possible (although admittedly not always present) with these three dimensional visualisations has the benefit of using the perceptual and spatial skills learnt and used all of the time in the real world within the virtual space. This can then play a large part in both the design and use of the visualisation, not least because it can enable the cognitive effort of a user who has had some exposure to the system to concentrate on the data under investigation rather than the visualisation interface and/or interaction mechanism.

Whilst the use of a metaphor creates a logical framework within which the artefact to graphical item is made, the mapping tends to suggest some form of analogy between the two groups of concepts. This may be true, but it is also true that the analogy can occur at higher levels of abstraction than the individual items.

There are still problems and issues to be tackled when visualising any data set in an acceptable and reasonable way. The first of these is to make available ways for the user to be able to explore and navigate the complex environments in which the visualisations are located. There is also the issue of converting suitable data to well-founded information (for the task in hand); metaphors as discussed above. Although there are many more issues, the final one presented here is the management of increasingly abstract problems and being able to communicate to users (and managers) the visualisation visions accurately.

3 INTELLIGENCE AMPLIFICATION

For clarification the term Intelligence Amplification (IA) is the use of computers to aid and enhance human intelligence rather than the Artificial Intelligence (AI) aim of trying to replace humans with machines. IA sets out to build on the skills humans have and to augment those areas where machines can usefully be employed without detriment to human ability.

There are several areas where humans can be identified to be more skilled. The first is *pattern recognition*; especially with the embedded knowledge a human has and can bring to the problem. Another is an *overall sense of context* that allows previously unrelated pieces of information to become related and therefore useful in a new situation. The final one is *evaluation*.

As a comparison consider the task of extracting information from a mass of data. To program a computer to be able to do such a task is a major undertaking and requires the employment of the many subtle mining and interrogation techniques. For humans presented with views of this mass of data it is a much simpler task because they are better equipped to make judgements and selections and to bring in apparently unrelated sources to enable them to make more informed decisions.

Walker [9] sums up the benefits of visualisation when he writes (emphasis added)

> *"The traditional interface of mouse, keyboard and screens of text allows us to work* on *computers, while techniques such as visualisation will truly enable us to work* ***with*** *computers."*

This is not to say that by using some random graphics in data presentations or in database applications will provide this type of working. The visualisations have to be well designed and implemented, and tailored to the task in hand to be most effective.

IA is of importance to (software) visualisation because in representing large and complex data sets graphically the aim is to help the user get a better understanding of the content of those data sets and to extract relevant information from the data. By aiding the user in this way, visualisation is an implementation of the IA theory. Reading through many thousands or millions of pieces of information and then summarising them in a finite graphical space would be an immense, complex, error prone and possibly tedious task. For a computer with the right "instructions" it is a simple data processing exercise.

Intelligence amplification is important because of the intangible nature of the data being visualised, as commented on by Brooks [1]. Software is based around the creation and maintenance of abstract components modelling in some way a small part of the real world and its processes. Because software is just zeros and ones, lines of program code, the task of visualising it becomes harder. Software may assume a real presence, may take on various characteristics even, but even though the interface is rooted in reality the underlying parts – the pieces of interest to software visualisers – do not have any physical form.

This intangibility is both a benefit and a hindrance. It allows great freedom in the metaphors and mappings that can be used in the visualisation process, but at the same time the wide choice of possibly appropriate metaphors can cause problems. It is for this very reason that metaphors can make or break a visualisation because of the mappings that can be made within the metaphor framework. A well thought out and well designed metaphor can be powerful in creating a usable and effective visualisation, but in the same way a badly designed ill thought out metaphor can cause many problems, to the extent that the visualisation might as well not have been created. Once the metaphor is in place in creating an effective visualisation, the intelligence amplification benefits of the visualisation can be exploited.

4 RESEARCH AGENDA

There are many unsolved issues relating to the creation and use of visualisations. One of the main ones is that of size. Software is large and complex; therefore any visualisation must be able to deal with this in some way. Knight and Munro [5] point out that one of the most difficult problems is in dealing with the possible range of data items to be visualised.

> *"The hardest problem in visualising anything is that, theoretically, the visualisation has to be able to deal with the range of items from one to infinity. This massive range means that automation and layout algorithms (both for two and three-dimensional visualisations) are hard problems. It also means that the visualisations need to be defined with this in mind; or to provide a way of dealing with very large numbers and indicating this fact to the user."*

This problem of scalability is not the only question that faces those working with visualisations. Another one of great importance for software engineering is that of evolution. The code underlying any software system goes through various revisions over time and if it is to remain useful and used the visualisation must move with the code. This then brings in a host of issues relating to how the visualisation changes over time or is regenerated and how it shows this to the user of the system. These and others are beyond the scope of the paper, and also require more research but both visualisers and visualisation users should be aware of these current limitations.

Von May et al. [8] suggest that the comprehension of complex software systems is one of the most challenging issues facing the software engineering community. Data mining and summary techniques can only help in certain circumstances as it may be the case that for interpretation the domain knowledge, experience and context that a human has is the only way to be able to properly identify patterns and anomalies. This could be for many reasons but often occurring ones are the subtlety of the data or the level of complexity in data relationships.

A recent paper by Feijs and De Jong [2] addressed the use of visualisation techniques in three dimensions for the viewing software architectures and relationships. Whilst this paper acknowledged that they have more research to do the work does not seem to address many of the issues and limitations of the use of three dimensions. The creation of their visualisations relies again on nodes and arcs but using an extra dimension. This does provide a greater degree of flexibility than the two-dimensional form of such structures but again does not scale or evolve well; two very important issues as highlighted above.

Achieving these Aims

In order to be able to create these powerful visualisations

Table 1 - Actual mappings from Java code to graphics

Visualisation Level	Code Element
World	The software system as a whole.
Country	Directory structure, which maps to the packages in Java.
City	A file from the software system.
District	Class (contained within the specific file and hence city in the visualisation).
Building	Methods.

that can deal with such important issues as scale and evolution (as highlighted above) it is useful to consider the basis the field has and where it should be heading.

There are other levels of dimension before the third; one-dimensional text and two-dimensional flat graphics. Three dimensions can be seen as a flat graphic, so the definition used here is that three-dimensional is the illusion of reality. Each of these levels has various attributes to a greater or lesser degree:

Figure 1 - A close view of part of *Software World*

- Abstraction
- Navigation
- Orientation
- Scalability

In order to characterise the dimensions the following notation can be used:

1D()

2D(REPRESENTATION)

3D(METAPHOR, REPRESENTATION)

The use of the representations and metaphors that are used to parameterise 2D and 3D provide differentiation between the three dimension levels over and above the differences in the previously identified attributes. Examples of 1D can be found in the Fletton and Munro work with hypertext [3], 2D in views employed by Storey et al. [7] and finally 3D in the representations explored by Knight and Munro [10]. These are by no means a complete summary of the field but provide an example of each category in the context of program comprehension.

From the above definitions it is possible then to represent virtual reality as:

VR = ELECTRONIC(3D(X,Y))

And to then be able to characterise virtual environments as:

VE = VR + MULTI-USER

Since this relationship contains VR and VR is explicitly electronic it does not include the virtual realities and environments that MUDs and similar technologies create purely through the use of mental imagery.

Based on these definitions the current state of the art in software visualisation can be written as:

$$SV = 2D(z_1) \mid 2D(z_2) \mid \ldots \mid 2D(z_n) \mid 1D()$$

where example instantiations for z can be *nodes and arcs* or *trees*. So, for example, the conventional nodes and arcs can be characterised in 2D as

2D(NODES AND ARCS)

This is not a good position for the field for the simple reason that many of the powers that visualisation can afford to the tasks to which it is applied are not exploited.

The future of software visualisation is more than this and can be better shown by the next two relations. The first is the current state of the art and a point that the authors have reached in current research.

$$SV' = SV + VR$$

for any suitable x and y instantiation of the metaphor and representation.

To be able to support the important activities to which the software visualisation techniques can be applied, then the research direction must be characterised by the final relationship:

$$SV'' = SV + VE$$

It is only when this step is made that the full power of software visualisation can be both seen and exploited. It is important to make clear the impact SV″ has. The electronic aspect of the three dimensions that the VR aspect brings in does not in itself enhance the attributes that three dimensions has. It merely restricts what is meant by 3D in this context and therefore allows certain assumptions to be made. In using the extra dimension for display the levels of abstraction, the navigation and orientation features that can be included and the ways in which the scaling issues can be addressed are all enhanced through the added variability the extra dimension affords. The real impact of SV″ is that it the multi-user facilities that the visualisation must then provide impact on all of the related components; metaphor, representation, abstraction, navigation, orientation and scalability.

1D()

2D(REPRESENTATION)

3D(METAPHOR, REPRESENTATION)

VR = ELECTRONIC(3D(X,Y))

VE = VR + MULTI-USER

$$SV = 2D(z_1)|2D(x_2)|...|2D(z_n)|1D$$

$$\Downarrow \qquad SV' = SV + VR$$

$$SV'' = SV + VE$$

Figure 3 - Relationship Summary

This is an area where much research is still to be done but it is the future of software visualisation. It is only through the power and intelligence amplification enhancing features that software visualisation can be used to address many of the current shortfalls with the visualisation systems today.

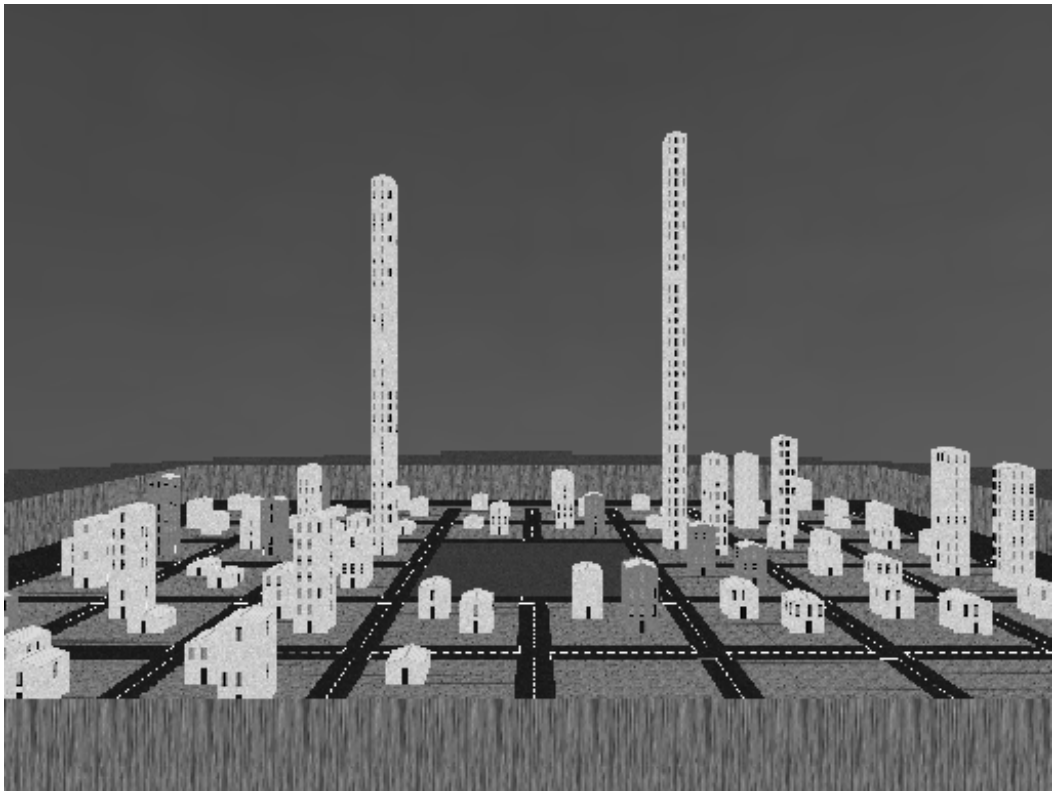

Figure 2 - A longer range view of part of *Software World*

5 VISUALISATION; CURRENT STATE

The following example and sample views are provided to show that the software visualisation field is capable of reaching SV′ and has indeed reached this point in some research. No particular examples are provided for one and two-dimensional work for the simple reason that everyone knows the views that this form of visualisation can take. This example shows how three-dimensional visualisation can be used for intelligence amplification tasks in the domain of software engineering.

Figure 4 - Part of the multi-user aspect of three-dimensional visualisations

An Example: Software World

The *Software World* is a representation based on a real world metaphor encompassing city items and urban developments. The main reason for this choice of metaphor was the power it afforded in scaling so that software systems of varying sizes could still be accommodated coherently by the one metaphor.

At a detailed level code artefacts are shown as buildings and districts in a city environment, whilst higher level detail is shown in a map (cartographic) style making use of overview and amalgamation techniques to fit the desired information into a finite space.

Software World is targeted at the visualisation of Java code and the levels of metaphor mapping are tailored to suit the language. The language definition of Java allows certain assumptions to be made and these have influenced the mapping decisions. Table 1 shows the high-level visualisation items and the corresponding code elements. Knight and Munro provide a more complete mapping in [5]. Figure 1 shows a view of the *Software World* in action when applied to Java source code. This view (and the one shown in Figure 2) has been automatically generated directly from the source code. The image shows several methods (buildings) in a cityscape environment laid out in blocks between the streets. Figure 2 shows the same visualisation from a greater distance, which allows the whole district to be seen at once and obvious things such as the relative height of the buildings can be seen. Figure 4 shows the design of a view that is intended to support the multi-user, inhabited view of three dimensional visualisations and is one of the many steps being taken down the route to being able to achieve SV″. It is

important to provide places in the virtual environments that appear (and act) as work areas in reality do. The office is intended to support interaction as well as allowing personalisation of the workspace. The shelves and filing cabinet can be used for storing notes, bookmarks to other areas of the virtual environment and documentation. The space itself can be used for meeting and communicating with other users, independent of the particular part of the software visualisation being discussed and away from the actual visualisation, but still supporting remote users and allowing easy reference to the visualisation.

6 CONCLUSIONS

This paper has provided a glimpse of the future of research in software visualisation, and in particular it has explained where the research effort should be focused. An automated three-dimensional prototype system has illustrated what can be achieved and some of the unresolved research issues have been outlined. There are many areas where work is still to be done both with issues identified and presented in the paper and with the many others that exist, but it is an area in which much benefit can and should be expected for the research effort put in now. The ability to enhance a users own cognition is a powerful notion, and can lead to powerful use, which in turn should benefit the work being done with the visualisation tool.

ACKNOWLEDGEMENTS

This work has been done with the support of an EPSRC ROPA award.

REFERENCES

1. Brooks, F. P. No Silver Bullet. *IEEE Computer* (April 1987) 10-19.
2. Feijs, L., and De Jong, R. 3D Visualization of Software Architectures. *Communications of the ACM, Vol. 41, No. 12*, (December 1998), 72-78.
3. Fletton, N. T., and Munro, M. Redocumenting Software Systems using Hypertext Technology. In *Proceedings IEEE Conference on Software Maintenance* (1988) 54-59.
4. Gelernter, D. *The Aesthetics of Computing.* Weidenfeld & Nicolson, London, 1998.
5. Knight, C., and Munro, M. Comprehension with[in] Virtual Environment Visualisations. In *Proceedings of the IEEE 7th International Workshop on Program Comprehension* (Pittsburgh, PA). IEEE Computer Society Press, 1999.
6. Pettifer, S., and West, A. Deva: A coherent operating environment for large scale VR applications, *Presented at the first Virtual Reality Universe Conference* (Santa Clara, California, April 1997).
7. Storey, M.-A. D., Müller, H. A., and Wong, K. Manipulating and Documenting Software Structures. *Chapter in Software Visualization*, World Scientific Publishing Co., 1996, 244-263.
8. Von Mayrhauser, A., Vans, A. M., and Howe, A. E. Program Understanding Behaviour during Enhancement of Large-scale Software. *Journal of Software Maintenance: Research and Practice, Vol. 9*, (1997) 299-327.
9. Walker, G. Challenges in Information Visualisation. *British Telecommunications Engineering Journal, Vol. 14,* (April 1995), 17-25.
10. Young, P., and Munro, M. Visualising Software in Virtual Reality. In *Proceedings of the IEEE 6th International Workshop on Program Comprehension* (Ischia, Italy). IEEE Computer Society Press, 1998.