Visualisation for Informed Decision Making; From Code to Components

Stuart M. Charters, Claire Knight, Nigel Thomas and Malcolm Munro
Visualisation Research Group
Research Institute in Software Evolution
Department of Computer Science, University of Durham,
South Road, Durham, DH1 3LE, England.
{S.M.Charters, C.R.Knight, Nigel.Thomas, Malcolm.Munro}@durham.ac.uk

Abstract

The problem of trying to view and comprehend large amounts of data is a well-known one. A specialised variant of this problem is the visualisation of software code and components for the purposes of understanding, decision-making, reuse and even integration. In particular the visualisation of software components, at a much higher level than source code, has received very little research. Visualisation is a powerful tool in situations such as this. This paper presents the application of real world metaphor based visualisations that address this problem. The application of visualisation to selecting software components is especially novel. It seeks to decrease the effort required by system integrators when locating suitable components in what is an increasingly crowded marketplace. Accurate information and understanding are vital if correct and informed decisions and judgements are to be made.

1. Introduction

Visualisation is necessary in the current arena of mass information generation, providing one way of filtering this volume of data into something more manageable. Virtual reality and three-dimensional techniques provide one way of enabling these visualisations and afford many benefits that more traditional visualisation systems lack.

SEKE '02, July 15-19, Ischia, Italy.

Through the use of three dimensions and suitable metaphors, information spaces or *information landscapes* can be created in which users have the freedom to explore and browse at will.

Such techniques are useful for many types of information visualisation and this paper documents the use of three-dimensional visualisation for a subset of information visualisation; software visualisation. Like information visualisation, software visualisation covers many aspects. In particular, this work focuses on software visualisation for the purposes of aiding the program comprehension process at the code level and on understanding black box components for use in component selection and reuse. It is an important area, not least because it is a major part of the maintenance and evolution of software systems. Very rarely is a system coded completely from new, which means existing code fragments (if not more) need to be understood before any effective additions and alterations can be made.

Software Components are units of executable code that provide functionalities through specified interfaces that can be composed together to provide a larger component or a software system. Components can take many forms; the main ones being controls, containers, command packages, libraries, frameworks and business components [1]. The software component marketplace is an expanding one, it is expected to grow from US$7 billion in 2001 to US$64 billion in 2002 [8]. As the marketplace expands the number of components and component repositories available can also be expected to increase. This increase will result in traditional techniques for component selection becoming more time consuming and less effective, making the job of the component purchaser more complicated. Hence, new and novel methods for the selection of components are

required to ease the burden on the component purchaser.

This paper examines the use of a real world metaphor visualisation, a cityscape, and its development from representing source code for program comprehension to components for decision support. The first section explains the use of metaphors whilst the next looks at metaphors in the context of visualisation. The two visualisations, *Software World* and *Component City*, are then presented and discussed. Areas of further work are also provided before summarising the contribution of this research.

2. Metaphors

A metaphor is a mechanism used to assist in the communication of meaning within a visualisation. Metaphors generally make use of knowledge that the user will possess. For most metaphor visualisations that means using a real world metaphor such as a cityscape. In certain highly specialised domains other metaphors can be utilised but this may restrict access to the visualisation to those with a high level of domain knowledge.

The selection of the correct metaphor for use in visualisation is one of the most important tasks, as an incorrect selection can result in a visualisation that is hard to interpret, interact with and navigate within. The end result being that the visualisation hampers rather than aids the user. In making the choice of metaphor the type and form of data needs to be considered along with the type of representation required. According to Benford et al. [2] the use of natural metaphors can aid the usability of virtual reality (and therefore visualisation) environments.

> *"… an attempt to exploit people's natural understanding of the physical world, including spatial factors in perception and navigation, as well as general familiarity with common spatial environments…"*

The research being described in this paper has tried to harness natural perceptual skills in this way, through the use of a very strong real world metaphor.

Pettifer and West [3] suggest that the potential power of virtual reality comes from the strength of its metaphor, and the fact that it is closer to natural interaction than many other forms of computer system. They also identify the benefits of natural metaphors, and making use of perceptual and spatial skills learnt and used in the real world in the virtual environment. It is obvious that the design of the metaphor used in the virtual reality can play a large part in the usability of that system, both in terms of human computer interaction, and in terms of enabling the user to carry out the required tasks. In this case the comprehension of the software through the use of visualisation.

Using three dimensions for visualisation adds an element of familiarity and realism into systems. The world is a three-dimensional experience and by making the visualisation more real world means there is less cognitive strain on the user. This in turn makes the system easier and more comfortable to use because the experience and knowledge the user has built up elsewhere. In using three dimensions the depth cues that make the world, and the visualisation, appear three-dimensional can be used as part of the visualisation. This means that the aim of the visualisation to aid the comprehension of complex phenomena can be achieved without adding unnecessary complications because of the visualisation used. What is also of benefit is that in using three-dimensional environments some of the cognitive processing needed for navigation and visual interpretation can be shifted to the sub-conscious as these are activities that are carried out daily with no real conscious thought.

3. Metaphors and Visualisation

Knight and Munro [8] define Software Visualisation as

> *"Software Visualisation is a discipline that makes use of various forms of imagery to provide insight and understanding and to reduce complexity of the phenomena under consideration"*

This definition provides an overview of the purpose of visualisation, to make understanding easier. Visualisations achieve this reduction in complexity in many ways, by the use of shape, colour, texture and by using real world knowledge to assist in interpretation. Visualisations can take in the main two forms; abstract representations using shape and colour to represent data or metaphor visualisations that use existing world knowledge to aid the representation of data by imposing a structure and rules upon it.

Visualisation presents one possible solution to the problem of component selection. Over the past decade the research area of visualisation and in particular that of software visualisation has matured greatly [7]. Whilst visualisation is not a 'silver bullet' to the problem it has many advantages that can be brought to bear on the problem of component selection. These advantages come in part through the considerations visualisation developers make when designing and construction visualisations. These are [7]:

- Navigation, Interaction, Orientation
- Scalability

- Evolution
- Automation

When these considerations are taken into account and combined with the power of visual imagery, visualisation presents itself as an ideal candidate for the task of aiding software component selection. One particular area of visualisation that lends itself particularly well to this area is real world metaphor visualisation; essentially the application of a real world model to represent otherwise intangible data.

4. Software Visualisation; *Software World*

Software World was developed to provide static visualisations of Java source code for the purposes of increasing understanding of that code. This understanding then aids any future development and maintenance and thus provides a mechanism in which informed decisions can be made. Without this information decisions made about the changes could have far-reaching implications, including reduced code stability and reliability.

The metaphor details are presented in the next section. This is then followed by some examples. This work was also a means to explore visualisations and metaphors in information provision. It has proved a good basis for further work in this area, as Section 5 shows.

4.1 *Software World* in more detail

In order to be able to coherently create and use the virtual reality environment for the purposes of program comprehension a metaphor that creates a constrained environment for mapping the data to be visualised is required. The mapping needed to be complete and logical to enable full utilisation of the visualisation process. As an overview, the mappings used for *Software World* are:

- **World**; flattened, overview picture (atlas style) not necessarily countries as would be known in standard geography but shows different elements of the visualisation at a very high level and the relationships between those elements.
- **Country**; each element shown in the world view is a country. It provides a way of splitting the items in the world down one level without the detail that is provided by the next level down.
- **City**; shown within countries, as the next level of granularity. These cities are composed of sub-areas but try to ease the navigation burden through the use of standard urban navigational aids.
- **Districts**; there can be from 1..n of these in a city, the number depending on the information to be represented in the visualisation. They group together related aspects of the software and provide groupings to be used when moving from a higher level of abstraction to a more detailed level.
- **Streets/Buildings/Gardens/Monuments**; these show the detail of the visualisation and provide the next level of abstraction down from the districts. They also act as legibility features and landmarks.
- **Inside Buildings/Gardens**; this is the finest level of detail, where detailed direct mappings from the code to the visualisation can be made. The use of walls, for example, also provides a way of displaying extra information to the user such as text. Text needs this sort of context in a three-dimensional space to be understandable and visible from many directions.

The mapping that was created within the confines of the metaphor became known as *Software World* because of the representations chosen and because within its mappings it had the ability to encompass the entire software system under consideration. Java source code attributes have the nice feature that they can be organised into a clean hierarchical structure and all attributes must be contained within this structure. The metaphor definition relies on this to make logical sense to those using it and has been created with this fact in mind. More details of the mappings chosen and the rationale for these can be found in [6].

The city metaphor has often been applied to visualisation problems but the motivating factors for its use as part of the *Software World* metaphor were:

- It fitted into the overall scheme of world, country, city, district and building used to provide a consistent but scaleable and evolvable visualisation.
- The use of abstract visualisation features has already received a small amount of attention in the software visualisation field and a real world visualisation was decided to be worth investigation.
- The use of a city as a container mechanism for a mid-level Java attribute worked well with long-term evolution, which from a software perspective is an important consideration.
- Using a city metaphor directly allowed the legibility features identified by Lynch [10] (and also covered by Ingram and Benford in [4]) to be easily incorporated to show their use in information gathering situations.

It may well be the case that other metaphors, mappings, representation, abstractions may well be suitable for this visualisation of Java source code. It is important to bear in mind that the use to which the

visualisation is to be put, and the underlying data, have a far greater influence on the appropriateness of any of these factors than is often recognised. These will govern the perceived success from both design (mappings and representations) and usability perspectives.

4.2 Software World Examples

Some examples of the views that can be seen in an implementation of the *Software World* when applied to real source code are included to show some of the points that have been presented and discussed in the previous sections of the paper.

Figure 1 shows an overview of the source code of a reasonably sized Java class, which is itself part of a much larger Java system. From this point of view the user of the visualisation is able to get an appreciation of the number of methods in the class, those methods that have a large line count, and those which are private methods thus conveying the visibility of the methods at the code level. All of this overview information provides, at a glance, useful contextual information before the user goes and investigates any of the methods in more detail. The use of a block (grid) structure for layout (blocks as in a city and roads), a central garden and an enclosing fence have been used to try and aid the navigation and orientation of the user within the visualisation system.

Figure 1 - Overview of a district; each district represents a class from the source code, with the methods shown as the buildings.

The colour of the buildings represents whether the method the building represents is private or not (private is a keyword in Java and indicates that the method cannot be accessed outside of the class in which it is declared). The height is a representation of the number of lines of code. Each building has a minimum height of one storey and a roof, but for each extra ten lines of code an extra storey is added to the building. The doors also provide information about the method. All buildings have a blue door – the reason being that a method may have no parameters and there still needs to be a logical way for a user to enter the building when exploring the environment. Parameters are shown in number and type (at a basic level) by extra doors. A door with either yellow or green paintwork shows all method parameters; the formal type of the parameter determines the exact colour.

5. Component Visualisation; *Component City*

Extending the *Software World* work from source code to components led to the development of *Component City*. This section explains the use of the metaphor visualisation as part of an on-going investigation into the representation and selection of software components. *Component City*, the name given to the visualisation developed, is a virtual environment based visualisation that currently represents a static representation of a software component repository. The aim of *Component City* is to develop a scaleable, evolvable representation of software components that provides an easily navigable environment with a shallow learning curve for non-expert users allowing them to select components based on multiple attributes.

The visualisation is generated from data detailing the functional properties of components. The visualisation allows users to examine the components available and explore the relationships that exist between those components. Within the city components are grouped by functionality such that similar components, sharing the same functional properties, appear close to each other.

5.1 Metaphor Mappings

It was decided to use the metaphor of a cityscape, a metaphor that has previously demonstrated effectiveness in software visualisation as shown in Section 4 and [6]. The cityscape metaphor is a powerful one. It is one that is easily related to as it is within most peoples experience and is relatively simple.

Any metaphor that is chosen for a visualisation may contain some inherent structure. It is important that this structure is used to the best advantage to maximise the benefits that a metaphor provides. For this reason it is important that the data and metaphor contain a similar structure. In using a real world metaphor a certain structure exists that data is required to be mapped onto. In the case of a cityscape the concepts of world, city,

district and building exist, these levels represent a hierarchical structure, increasing in detail at each stage. These mappings echo in part those used in *Software World* as above and [6], [7] to describe the levels within Java source code.

It is important to choose suitable mappings to avoid distorting either the metaphor or more importantly the data being represented. If distortion were permitted to occur it is possible that the user would begin to draw incorrect assumptions about the virtual world with which they are interacting such that the meaning of the data is altered. Incorrect conclusions are worse than having no or incomplete conclusions because of the impact this has. Any errors propagate through the process and are compounded and magnified at each stage.

Taking this into consideration and maintaining the hierarchical structure of the cityscape then following mappings were chosen for *Component City*:

- **World**; A flattened overview of *Component City* showing the distribution of components.
- **City**; A more detailed overview highlighting areas representing functionality groups.
- **District**; A functionality group, components are clustered into districts based on their functional properties and their relationship to other components.
- **Building**; Representing a single component or multiple very similar components.

Three types of building are currently used in *Component City* (as seen in Figure 2):

- **Houses**: Representing a single component.
- **Mansions**: Representing two components
- **Skyscrapers**: Representing more than two components, with skyscrapers the number of levels indicates the number of components at that location.

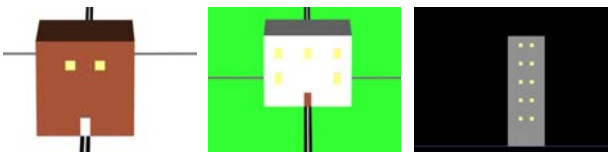

Figure 2 - A House, Mansion and Skyscraper

Different buildings were selected for representation at the component level for several reasons. The first is to provide a more distinctive landscape, the different styles of building and their location relative to each other will assist navigation and orientation within the world. A landscape where everything looks very similar can be even more confusing when a user is just beginning to use the visualisation and could prevent the rapid construction of a cognitive map of the world. The second is to provide an immediate visual distinction between areas that contain large numbers of related components and areas that contain large numbers of highly similar components.

5.2 Navigation

Navigation within visualisations is recognised as a difficult problem. The use of a metaphor visualisation aids that problem by exerting constraints on the types of navigation permitted to maintain the reality of the metaphor however additional aids are also required. Landmarks have been shown to aid navigation within visualisations [4, 9] and it is this device that has been currently selected for use within *Component City*. *Component City* contains several monuments which act as landmarks within the city. These landmarks aid the user with both orientation and navigation within the city.

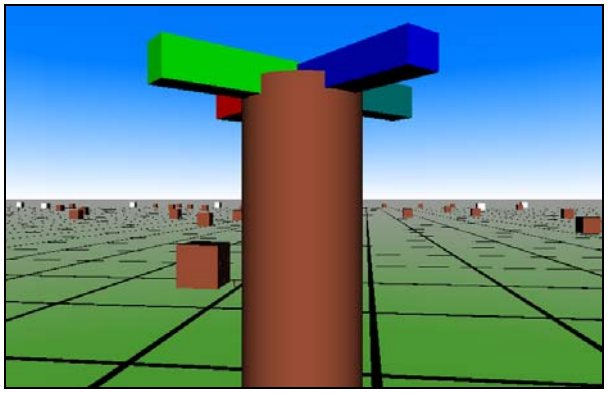

Figure 3 - A Monument within Component City

Currently monuments exist in the centre and at each corner of *Component City* aiding users in their construction of a cognitive map for *Component City*. The central monument (Figure 3) also acts as a signpost. The top of the monument has four arms each of a different colour; the colour of the arm corresponds to the colour of the corner monument to which the arm is pointing.

5.3 Implementation Details

The process of generation for the visualisation uses an XML representation of the software component repository detailing software components and their functional properties. This XML file is fed into a self-organizing map [5] that is used to group the components based on their functional properties. The self-organizing map was chosen for the grouping of components as it operates with little configuration and could later be used for automatic generation of the

visualisation. XML was chosen to describe the input into the self-organizing map as it allowed a common format for translation of data from component repositories and subsequently allows the use of XSLT for transformation to VRML to display the visualisation.

The results from the self-organizing map are then transformed using XSLT into a VRML model of *Component City* that can be viewed within any VRML Viewer or Web Browser with appropriate plug-in.

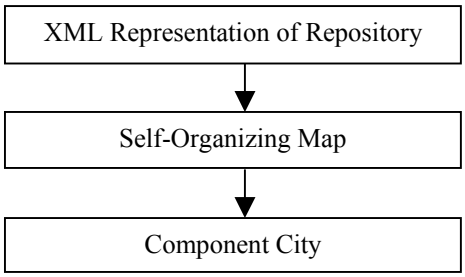

Figure 4 - Visualisation Generation Process

The use of a VRML model allows the visualisation to be multi-platform and to use standard software for the display of *Component City* making the visualisation more accessible to users. The XML output from the self-organizing map and the use of XSLT to transform this output to VRML allows rapid evolution of the visualisation to incorporate new feature and also allows the transference to another language for displaying three-dimensional worlds with less development effort.

5.4 *Component City* Examples

To demonstrate the ability of *Component City* an example task that may be undertaken is to locate a component that provides Document Management functionality for inclusion into an administration system. It is known that certain basic functionality is required including being able to convert to and from a variety of imaging formats. Other functionality that a component can provide in addition to the basic required functionality may enhance the chance that it is selected for inclusion within the system. To find a suitable component for this task the visualisation can be entered at a world level allowing the distribution of software components to be seen, as shown in Figure 5.

The users can then descend to district view to select an area of functionality appropriate to the component they are looking for, in this case IMAGING. This level allows individual components and groups of components to be seen more easily to again an idea of coverage for the functionality area under consideration. The user can also see other functional groupings in the area selected also shown in Figure 5, this can aid the user in locating other components or in finding components that also contain other functionalities.

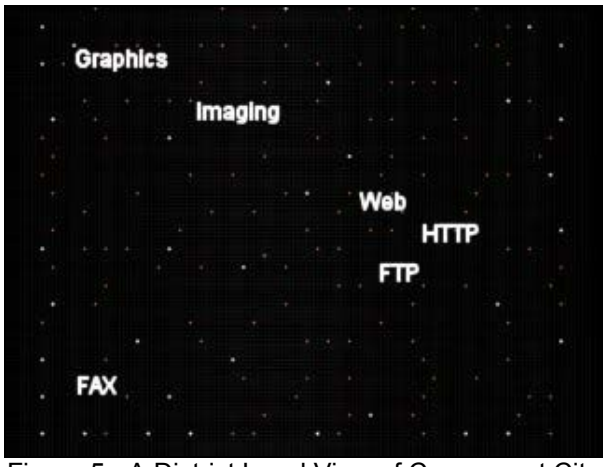

Figure 5 - A District Level View of Component City

From the district level the user can select the desired functionality and then descend to street level where individual buildings within the district can be seen (such as in the distance in Figure 3). At Street level the user can navigate around the area and the buildings examining the individual components that are present in the area.

The user can go right up to a building at this level. It is at this level that detailed information about component will be presented to the user allowing them to examine in detail the component under consideration and decide if this component should be selected as a possible candidate for inclusion within the software system to be developed. The buildings have a variety of features; the front door acts as an entry point, allowing the use to select the component, the windows act as indicators that the component has been selected by changing colour.

All of the information presentation, from the district level through to the building level aids in the process of component comparison and selection. The detail presented at the building level can prove invaluable when trying to decide between which components are most appropriate.

6. Users of Component City

Developers of software systems have a variety of needs, these needs are reflected in *Component City*. Users are likely to use the visualisation erratically, with periods of intense activity towards the start of a project when finding and evaluating components for the system.

Use of the system will decline as the development of a system progresses as the requirement to find new components is removed.

The goal of a user of *Component City* is to find a selection of components that could possibly be used in the development of their system. To this end the user has to make some choices about the core functionalities they require in their components. This choice is supported through the use of clustering which brings together those components that share functionalities allowing the user to review all the components of a similar type in one area. The overview level of *Component City* allows the user to select which area of functionality that they wish to explore.

The system therefore is interfaced in such a manner that a reasonable level of usability can be achieved in a short time and that the level of skill can be quickly reacquired after a period of non-use of the system.

7. Issues and Future Work

As mentioned earlier navigation within visualisations is difficult. Within *Component City* this issue is partially addressed through the use of monuments located in the centre and corners of the world. These act as landmarks and aid navigation and orientation within the world. However they do not solve the problem in its entirety and this is an issue that requires more investigation.

Information representation within *Component City* at its current stage of development does not allow for all information known about a component to be displayed especially with regard to non-functional attributes of the components such as quality, reliability and security. These non-functional properties whilst not forming part of the initial decision making process when functionality is most important may be required once the candidate components have been selected for evaluation. *Component City* at its present state of development has shown considerable advantages and use in the area of component selection but requires work in several areas in order that it can fully support multi-attributive decision-making in the process of software component selection.

At present *Component City* contains information on approximately 200 components for initial development of appropriate interaction and interaction. However it is envisaged that this will expand to the order of 10,000 components or more in full use. The data for these components will be drawn from multiple components repositories through the use of a virtual repository system.

In an idealised world the source code to the components would be available. In many internal company situations this would hold true. Where source code and component level attributes are both available, dynamic linking between *Software World* and *Component City* is a desirable addition to this work. Developers working at the code level could identify possible groupings of packages and classes that would make candidate component solutions. By adding component attributes this can be integrated into *Component City*. The reverse situation may also be necessary. Having identified a small number of solutions, examining the source code may be necessary to determine if the low level functionality is that required when the component/code is reused. *Component City* can provide the relevant reference to *Software World* and this then would be able to display the methods and their attributes for inspection.

One such area is that of Information Representation. Improving the information that is displayed to allow rapid assimilation of multiple attributes both functional and non-function is an area that requires considerable development. New techniques for the representation of data within a three-dimensional environment may be required to overcome some of the present limitations that exist with visualisations.

Component City will be evaluated in two main respects. The first, evaluation of the visualisation, will be conducted using previously established frameworks [6, 11], which will be applied and refined where components are not adequately addressed. The second evaluation theme is the effectiveness of the system for component selection, this will be conducted by comparing the time to find candidate components using *Component City* and using more standard methods of textual query and listed responses as used often today.

8. Summary

Software World was implemented using a desktop virtual reality system using C code. This C code was automatically generated from the Java source code in a two-step process. The source was parsed in order to populate a repository, and then from this the necessary code for the visualisation was generated. *Component City* takes data from a component repository in an XML format, through the use of a self-organizing map this component data is classified based on functional attributes. The classified components are then represented using the real world metaphor of a city at varying levels of abstraction, World, District, Street and Building. Users can view the visualisation using

standard VRML enabled browsers or stand alone viewers and interact with the world.

Software World provided a good grounding for the later work with *Component City*. The basic ideas of exploiting metaphors and space to represent essentially intangible attributes have proved useful with both code and components. It is however acknowledged that in order to be of greater use *Component City* must incorporate non-functional attributes and provide a more expressive display of the information known about components.

The current status of *Component City* demonstrates the power of visualisation and the applicability of that visualisation to the task of component selection on the basis of multiple attributes. The work on *Component City* so far also demonstrates a need for further work on the use of visualisation for software components and for decision support especially in areas that require multiple attributes to be evaluated.

Acknowledgements

This work has been partly financed by an EPSRC studentship, partly by an EPSRC ROPA research grant; VVSRE, Visualising Software in Virtual Reality Environments, and partly supported by EU Grant: IST 1999-11631 CLARiFi: Clear and Reliable Information For Integration

References

[1] V. Traas and J. van Hillegersberg, "The Software Component Market on the Internet Current Status and Conditions for Growth", *Software Engineering Notes*, Vol. 25. No. 1 pp 114-117 January 2000.

[2] S. Benford, C. Brown, G. Reynard, and C. Greenhalgh. "Shared Spaces: Transportation, Artificiality and Spatiality", *Proceedings of the 1996 ACM Conference on CSCW*, Boston, Masachusetts, USA, November 16-20, 1996, pp77-85

[3] S. Pettifer and A. West, "Deva: A coherent operating environment for large scale VR applications", Presented at the first Virtual Reality Universe conference in Santa Clara, California, April 1997.

[4] R. Ingram and S. Benford, "Legibility Enhancement for Information Visualisation", *Proceedings of Visualization 1995*, Atlanta, Georgia, October 30th - November 3rd 1995.

[5] R. Beale and T. Jackson, "Neural Computing An Introduction", *Institute of Physics Publishing Bristol and Philadelphia* 1992 Reprint, 0-85274-262-2.

[6] C. Knight, "Virtual Software in Reality", *PhD Thesis, Department of Computer Science*, University of Durham, June 2000.

[7] C. Knight and M. Munro, "Visual Information; Amplifying and Foraging", *Visual Data Exploration and Analysis VIII, in Proceedings of SPIE*, San Jose, USA, January 2001

[8] C. Knight and M.Munro, "Comprehension with[in] Virtual Environment Visualisations", *Proceedings of the IEEE 7th International Workshop on Program Comprehension*, Pittsburgh, PA, May 5-7, 1999.

[9] J. Burrus, R Waters, and D. Anderson, "Locales and Beacons: Efficient and Precise Support for Large Multi-User Environments", Mitsubishi Electric Information Technology Centre Technical Report TR-95-16a, August 1996.

[10] K. Lynch, *The Image of the City*, The M.I.T. Press & Harvard University Press, Cambridge Massachusetts 1960.

[11] P. Young, Visualising Software in Cyberspace, *PhD Thesis, Department of Computer Science*, University of Durham, Octiber 1999.