Multimodal Modelling Languages to Reduce Visual Overload in UML Diagrams

Kirstin Lyon, Peter J. Nürnberg
Department of Computer Science, Aalborg University Esbjerg
Denmark

(kirstin, pnuern)@cs.aue.auc.dk

Abstract

It is normally easier to build a program when the design issues are fully understood. One way of doing this is to form a "mental map" or visualisation of the problem space. Visualisation is an increasingly important method for people to understand complex information, to navigate around structured information and to aid in software design. Software visualisation techniques are usually tackled from a graphical point of view. Some similarities exist between spatial hypertext applications and software visualisation techniques. A limitation of both of these is that when the amount of information becomes large enough, users may experience visual overload. This paper suggests using multimodal cues to reduce this.

1 Introduction

Software visualisation deals with animation of algorithms as well as the visualisation of data structures. One of its main goals is to achieve a better understanding of complex programs, processes, and data structures through diagrams. Graphical representations allow for a quick overview of information. In such representations multidimensional conceptual structures are reduced into a plane. This can cause several design issues. Users may also experience difficulties in understanding a diagram that they themselves created previously due to its complexity.

Organising large amounts of unstructured information has similar difficulties to that of software design modelling. A difficulty with present modelling systems is that the structures they provide are often too rigid, and are often fixed for/by the user before the problem space is fully understood. Even when a problem space is fully understood, conditions may change through time. Entities may have various visualisations depending on the contexts in which they are being used. Changes in visualisation may force users to rethink structures. Some software and knowledge visualisation tools share similar principles, and therefore similar benefits and limitations.

When the amount of information to be organised becomes large enough, users may experience information overload. For example, a user has 10 emails to organise. This is a relatively easy task as many of the details from each page can be remembered. If this number is increased to 10,000, remembering each email becomes unrealistic. Now the emails need to be organised in some way, for example by category, date, size, sender etc. A user is able to remember a collection of categories rather than individual mails. Without this structuring of information, users become overloaded with information. Visual overload occurs when users are presented with many graphical elements simultaneously, for example, in a UML diagram.

Some hypertext tools allow users to organise their information spatially without knowing the structure beforehand, and allow users to model changing spaces. With these tools users create work areas and modify objects within this space with the organisation of information occuring over time. However, difficulties occur when trying to recreate highly complicated structures in a linear/planar representation e.g. serialising a data structure is known to be simpler than parsing it. Some suggestions as to how to deal with visual overload exist, e.g. increasing dimensions (Robertson et al., 1998), using different types of visual cues or increasing modalities. This paper focuses on the latter option.

The remainder of this paper is organised as follows: Section 2 compares spatial hypertext and software visualisation tools. Section 3 outlines some possibilities of modality use.

Section 4 describes a scenario. Section 5 provides some future work and Section 6 some conclusions.

2 Spatial Hypertext and Software Visualisation

In this section, we look at how spatial hypertext could improve software visualisation tools. We first consider some tools are available at present for software designers. We then consider some spatial hypertext systems.

2.1 Software Visualisation Tools

The Object Oriented Analysis and Design (OOA&D) methodology considers programs to be simulations of reality. In order to build an effective program an efficient model must be created first. The tools used for this are graphical in nature e.g. rich pictures, class diagrams, use case diagrams etc.

Tools such as BlueJ(Sanders et al. (2001)) converts users' class diagrams into classes automatically. A more complete suite, Rational Rose(Rose), offers a more sophisticated implementation of UML diagrams that also include re-factoring and full cycle capabilities. JBuilder(JBuilder) and similar products provide a fast and visual method of creating a user interface prototype.

2.1.1 Modelling Languages

Modelling languages allow users to model the real world graphically. One such language is UML. UML provides the user with a set of graphical elements, and allow the user to structure them in a way that is appropriate for the task. Modelling languages organise a subset of knowledge that is important to a particular software design problem. UML is currently the most widely accepted standard for object-oriented software diagrams. However the notation set is large, and can be customised (Wieringa, 1998).

Other modelling languages include Organisational Requirements Definition for Information Technology Systems (ORDIT) and SCIPIO.

ORDIT is based on the idea of role, responsability and conversations, making it possible to specify, analyse and validate organisational and information systems supporting organisational change(Dobson and Strens (1994)). The ORDIT architecture can be used to express, explore and reason about both the problem and the solution aspects in both the social and technical domains. From the simple building blocks and modelling language a set of more complex and structures models and prefabrications can be constructed and reasoned about. Alternative models are constructed, allowing the exploration of possible futures.

The SCIPIO method identifies three ways of modelling a situation, to support strategic planning decisions as well as tactical design decisions(scipio).

- Business relationship modelling provides a view of the network of collaboration agents and their oblications to one another.
- Business rule modelling provides a view of the rules, policies, regulations and other constraints imposed on the business.
- Business asset modelling provides a static view of the resources of the organisation.

Both languages are graphical in nature. They have similar problems to other graphical modelling languages.

2.2 Spatial Hypertext Tools

Hypertext research looks at how to present and represent material electronically that is too complex to represent on paper (Nelson, 1965). Spatial hypertext is a structure discussed in the hypertext domain. It takes advantage of our visual and spatial intelligence. These tools allow users to organise information spatially, and allow structures to emerge over time.

In general, spatial hypertext tools are similar to Visual Knowledge Builder (VKB) (Shipman et al., 2001). These tools are good for information-intense activities, such as analysis or research. How information is presented is decided upon by the user, with no specific structure being decided for them beforehand. As new files are added, structures will inevitably change. Files are given various attributes which allow the user to see what items are related i.e. colour, proximity, shape and size.

2.2.1 Modelling Language

Spatial hypertext tools provide the user with a modelling language inside the application that performs a similar task to that of Unified Modelling Language (UML) for software designers. The visual layout of these tools and UML may be viewed as visual languages. UML diagrams can be seen as similar to spatial hypertext. Both interfaces share visual properties.

2.3 Comparison

When comparing spatial hypertext and UML diagrams, it is noticable that in both, some form of spatial organisation exists. Shapes of objects are also important. UML has a predefined set of shapes that have specific functions and semantics, whereas the language used in VKB is decided upon by the user.

Some limitations for both languages exist e.g. diagrams in both languages can become complex and difficult to interpret and remember through time. It is also difficult to categorise information. There is also a difficulty when the amount of information becomes large, users become inundated with information. If we consider a UML class diagram to be similar to a spatial hypertext diagram, we can use some solutions suggested by that community to improve this.

3 Multimodal Modelling Language

This section first of all describes a possible language, and then considers what devices could be used.

3.1 Modelling Language

Multi-modal interfaces use more than one modality(sense) to convey meaning i.e.vision, audio, haptic etc. It is possible to mix various modalities e.g. visual and audio, tactic and audio etc. Research has considered various combinations of modalities and how to create an interface. Guidelines exist on how to create multimodal interfaces Heller and Dianne Martin (1995). One method is to use them in a way you would naturally. This approach has several benefits. Each sense has a particular set of characteristics. If this is exploited correctly, user interfaces become easier to use. This paper considers increasing the number of modalities to include visual, audio and haptic (sense of touch).

When considering what information to transfer to what modality, it is necessary to understand the strengths of each sense. This is summerised in tab. 1.

Vision is best suited to high-detail work. It can attend to anything in sight at that time. However, when the amount of information increases it becomes difficult to maintain and overview and focus at the same time. Visual attributes include proximity, colour, shape and size. Audio is better at maintaining an overview. It has less detail than visual cues, but

Quality/Ability	Audio	Visual	Haptic
Transmission	Broadcast, linear	Broadcast	Narrowcast
Information content	High, but less detail than complex images.	High and flexible	Surface properties

Table 1: Comparison of audio,visual and haptic cues.

may be useful in providing spatial or density information. It may also be useful to describing relationships between objects. This is difficult to do with visual cues, as diagrams can become complex when all relationships are described. Its attributes include pitch, rhythm, volume, etc. Tactile (touch) information is limited in the amount of information it can hold. You are limited to how much you can touch at the same time. This could be useful for highlighting areas of diagrams. It's attributes include roughness, temperature etc. Sutcliffe (2003)

This paper suggests using visual cues for high detailed areas, audio for maintaining an overview of the relationships between objects, and tactile for highlighting objects that are of interest to the user.

3.2 Devices

Difference modalities require different devices, for example, as most information presented by a computer is through visual means, all computers are provided with a monitor. Audio cues require either headphones or speakers. Both of these come as standard with many computers. It is not so easy to use tactile cues.

There are two main types of haptic devices:

- Glove or pen-type devices that allow the user to "touch" and manipulate 3D virtual objects.
- Devices that allow users to "feel" textures of 2D objects with a pen or mouse-type interface

WingMan Force Feedback Mouse and the iFeel mouse are examples of 2D devices(ifeel and wingman). Examples of 3D devices include the Phantom(phantom) and CyberGrasp(cyberglove), a glove style interface.

Before specifying what tool should be used, it is necessary to consider exactly what kind of cues the user will receive, and if the user should input tactile cues themselves.

4 Scenario

In this section, we consider a scenario of a UML class diagram use. We base this scenario on traditional software design methodology, i.e. proto-type is designed then implemented, testing and evaluation is carried out and the prototype is modified with information from the evaluation.

Introduction. *David and Grace are programmers. They have been newly employed by a university to modify the payroll system for staff. First they look at the original designs of the program. Then they decide what needs to be modified through checking the new tax system, and how the salaries of staff will change. Then they create a new design. Once this is established they implement the modifications and check that all classes are changed if neccessary.*

Diagram familiarisation. *Initial challenge is to learn how UML is being used in this case so that standards can be maintained.*

David and Grace would consult either other programmers at the university who are familiar with UML, or consult the documentation that describes how UML is being used in this case. Some time is required to become familiar with the university's usage of UML

Design familiarisation. *Next they must become familiar with the original design.*

They consult the program's diagrams. As this is a large program, the diagrams are large and complex. Some time is required to form an overall understanding of how the program works.

Program Design. *Modifications need to be designed. This may involve classes being added, deleted or modified in some way. To do this, they need to understand how the classes interact, and the constraints on each class.*

These relationships are not shown in UML class diagrams at present. They would have to use other diagrams to find out what parts of objects related to what other parts of objects. They need to know the constraints on each object too.

Update documentation. *As an ongoing task, documentation is updated so that when the program needs to be modified again, future employees may understand how the program works.*

At present, diagrams would be modified with the new changes. Difficulties still occur when it is not clear where an object belongs. A particular visualisation that best suits may be chosen. In future when the program needs to be modified once more, it may be different people that work on it, so documentation needs to be clear.

4.1 Multimodal suggestions

This paper suggests transfering some of the visual information into audio or haptic output. Different relationships can be represented by different modalities. At the moment it is difficult to represent many-to-many relationships. Audio could be used to represent all relationships between objects. Haptic output could be used to highlight specific objects, e.g. when an object is touched, the 'feel of the object changes in some way.

Increasing the number of modalities used can increase the amount of information the user can take in at one time. Each modality could contain information for a different layer of the diagram. The main difficulty of UML diagrams is that they are trying to represent information that is multi-dimensional, so using only 2D graphics is perhaps too limiting.

5 Future Work

Our next step is to build a spatial hypertext prototype that uses a multimodal modelling language. Guidelines for multimodal interface design already exist, but it is still difficult to find a useful balance between the modalities. It is also not clear what information is best represented in audio or haptic instead of vision. To resolve this, usability tests will be carried out at various stages of the analysis to find this balance. With those results fed into a prototype.

6 Conclusions

At present the main focus for modelling languages has been on creating them visually. This has limitations, for example, how to represent mutli-dimensional information in a 2D space. Users can become overloaded visually due to complex diagrams, and it can take time to become familiar with unknown designs. Similarities between software visualisation and spatial hypertext exist. Both are useful for information intense tasks, and have a visual modelling language. Some possible suggestions for reducing visual overload have been suggested from the spatial hypertext community, and should also be applicable to the software visualisation community.

References

Keith Andrews, Wolfgang Kienreich, Vedran Sabol, Jutta Becker, Georg Droschl, Frank Kappe, Michael Granitzer, Peter Auer, and Klaus Tochtermann. The infosky visual explorer: exploiting hierarchical structure and document similarities. *Information Visualization*, 1(3/4):166–181, 2002. ISSN 1473-8716.

Nelson Baloian and Wolfram Luther. Visualization for the mind's eye. In *Software Visualization*. Springer, 2001.

cyberglove. URL `http://www.immersion.com/3d/products/cyber_glove.php`.

J. E. Dobson and M. R. Strens. Organizational requirements definition for information technology systems. In *IEEE International Coference on Requirements Engineering*. IEEE, 1994.

John Domingue, Blaine A. Price, and Marc Eisenstadt. A framework for describing and implementing software visualization systems. In *Proceedings of Graphics interface*, 1992.

Jing Dong. Uml extensions for design pattern compositions. In *Journal of Object Technology, vol. 1, no. 5, pages 149–161*, 2002.

Rachelle S. Heller and C. Dianne Martin. A media taxonomy. *IEEE MultiMedia*, 2(4):36–45, 1995. ISSN 1070-986X.

ifeel and wingman. URL `http://www.logitech.com/`.

JBuilder. URL `http://www.borland.com/jbuilder`.

Ted Nelson. Complex information processing: A file structure for the complex, the changing and the indertimate. In *Proceedings of the 1965 20th national conference*, 1965.

phantom. URL `http://www.sensable.com`.

George Robertson, Mary Czerwinski, Kevin Larson, Daniel C. Robbins, David Thiel, and Maarten van Dantzich. Data mountain: using spatial memory for document management. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 153–162. ACM Press, 1998.

Rational Rose. URL `http://www-306.ibm.com/software/rational`.

Dean Sanders, Phillip Heeler, and Carol Spradling. Introduction to bluej: a java development environment. In *Proceedings of the seventh annual consortium for computing in small colleges central plains conference on The journal of computing in small colleges*, pages 257–258. The Consortium for Computing in Small Colleges, 2001.

scipio. URL `http://www.scipio.org`.

Frank M. Shipman, Haowei Hsieh, Preetam Maloor, and J. Michael Moore. The visual knowledge builder: a second generation spatial hypertext. In *Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, pages 113–122. ACM Press, 2001. ISBN 1-59113-420-7.

Alistair Sutcliffe. *Multimedia and Virtual Reality: Designing Multisensory User Interfaces.* Lawrence Erlbaum Associates, 2003.

Roel Wieringa. A survey of structured and object-oriented software specification methods and techniques. In *ACM Computing Surveys, Vol.30, No.4.* ACM, 1998.