Published in Universal Access for Human Computer Interaction 2001 (UAHCI 2001) as part of HCI International 2001

August 2001, New Orleans, LA, USA.

Towards Automatic Adaptation of Data Interfaces

C. Knight and M. Munro

Visualisation Research Group, Research Institute in Software Evolution, Department of Computer Science, University of Durham, Durham, DH1 3LE.

Personalisation and user interface technologies are now at the stage where user preference can be inferred with some degree of success. Such approaches are not necessarily always the best approach as user autonomy is important for some. There is also potential for component based data interfaces. Such interfaces can be loaded and applied on demand with (distributed) component technologies and running code can be replaced on the fly. This theoretically enables multi-metaphor data interfaces to be taken a step further from the direct customisation level at which most currently work. This paper examines these future moves from the point of view of a directly customisable multi-metaphor visualisation tool. It currently allows visualisations and data sources to be linked and provides much of the connectivity for the user; enabling non-expert use. The automatic adaptation of such multi-metaphor tools can stem from this work. This can only help towards the goal of information access for all.

1. INTRODUCTION

The goal of an Information Society for All requires that interfaces to software are changed to be able to accommodate the wide range of expected users. Because no two users are alike, and all users have their own preferences and ways of working then flexible interfaces that provide choices in the representation of data, and allow for expansion, provide a way of overcoming the subjective nature of interfaces. Unfortunately this goal of having an easily personalised and customised user interface is easier to perceive than to create. Many issues exist with technology constraints, and also with what should be expected of a user. Obvious changes such as the colour scheme used can be inferred from the windowing system defaults; indeed these may be enforced by the window managing software. What about the case where such defaults are not required, or cause problems with some other information display? Something that seems trivial has many possibilities.

There is also the maintainability concept where users all have their own displays, layouts, and favourite ways of using the software. There has to be some way in which this can be reconciled for software updates and versioning; and all the better if these can maintain user preferences through the change process. Software is not a trivial object; it is complex on its own, and that complexity increases with the number of interactions that it participates in.

The research presented in this paper puts forward a framework in which the ideals of an Information Society for All, and therefore catering for a wide range of diversity, can start to be achieved. It has been reached through the desire for more customisation and flexibility with visualisations of large abstract data sources, and these provide examples of the concepts. Future work that derives from this is the automatic inference of aspects of the interface. This can help both novice users who need an entry route to the system, and advanced users who already have ways of working with the system and expect changes in interfaces to reflect this prior experience.

2. PERSONALISATION & CUSTOMISATION

Personalisation is something that is hard to quantify because the term is used to represent many aspects of a customisable and flexible interface (Kramer et al.). It is important to remember that the key aspect of a personalisable interface (whatever that may mean!) is that it needs to bring some form of *value-add* to the user of that interface. There is always the problem, especially with new technology, of falling into the feature trap so beloved of software applications developers; the addition of fancy features, mainly to act as selling points, or as something that their competitor's products do not have. It is necessary to ensure that this does not happen with any interface, but personalisable features should be included for a justifiable reason.

An interface will be deemed successful if it is able to support the user in achieving their current goal. This means that the personalisation aspects should be related to the tasks expected to be supported. Obviously domain knowledge is also of importance with any personalisation. If something is always constant in a given domain, making that a frequently configurable parameter is likely to cause users, particularly experienced ones, irritation. Conversely, some editing of "constants" is necessary for the once every x years that the value changes.

There is also a need to empower users through supporting them in the tasks and activities that they need to carry out, but not to restrict them too much in the way this is done. A flexible interface is much more likely to be able to adapt to slight changes in working practices than a rigid one. Clearly large changes may force some interface redesign, but in cases like this there is usually a system redesign as well. The impression of empowerment is an important one; it supports and promotes the view of a usable interface, and it allows for the creation of intelligence amplifying tools. Tools that help with the completion of real tasks become pervasive (at least in the minds of the users) and can therefore be considered to be a success (Karat et al.). Intelligence amplification does not seek to replace users in the way that much of artificial intelligence does. Instead it looks to support users through aiding them with tasks that a computer is more than capable of doing, and then allowing the insight and experience of the user to be utilised to actually achieve the end result. Intelligence amplification also supports the view the tools should support the humans, and not control the activities that they are able to carry out.

Personalisation in many ways means that an interface feels like it was designed for that user (or at worst, that class of users). Customisation deals more with the configurable items that an interface has to enable any user to fine-tune certain aspects of it. With current technologies and algorithms there is some cross over between these two concepts, but this distinction is appropriate for this work.

Much of computer science remains an art, despite the scientific endeavour and theory behind it. Those facets of computer science that deal with graphics and interfaces (in any combination) come even closer to the view of feeding information into a magic black box and then the perfect result appearing at the other end. Those within the discipline appreciate that this is not the case and that engineering and scientific principles can be applied. This research is an attempt to achieve this in a flexible way.

3. VISUALISATIONS

Visualisations are unusual in that there are often two interfaces involved for interaction with the user. This can even be the case even with three-dimensional visualisations located in virtual environments composed of sensors and panel displays. There is the more usual GUI that is used to start the application (if on a desktop display) and often for the configuration of the visualisation with, for example, the data to

be utilised and the representations and metaphors required. There is then the visualisation itself, which from a usability engineering (Neilson) point of view should be able to directly interacted with, at least at a rudimentary level such as navigation.

Visualisation is also subject to many of the same problems as interfaces over what is considered to be aesthetically pleasing, easy to use, easy to learn, necessary displays for tasks and so on. Because of the high level of subjectivity, and in many ways more so than for GUIs because of the freedom of display dimensions and metaphor, the ability to personalise and customise visualisations is seen as vital in ensuring their acceptance in standard interfaces and from usability engineering perspectives.

It has been suggested that representation (at an interface level rather than for visualisations, in which case the metaphor is the guiding factor) ultimately affects the presentation and interaction with a service or product (Pednault). It is because the representations used limits the extent to which personalisation and customisation can be applied, and also the information that can be captured and utilised towards furthering this aspect of the interface. Some of this is obviously visible to the user through the display presented to them, and in the range of interaction devices that can be utilised with it. Other aspects of it relate to the structures and code that exist in that product or service at the programming and architectural side of things. Because so many of these issues are dependent on the code and the requirements upon which it was based, then it is hard to say what these should be for all systems. A step in the right direction would be something like XML (XML) whereby data exchange could be made common through the use of embedded descriptions of that data structure. This issue is included here because it is similar to the problem experienced by many visualisations. It is technically possible to factor and/or transform most data sets, especially given a range of interfaces to data stores, but whether that is meaningful for the visualisation is a different issue! (Knight)

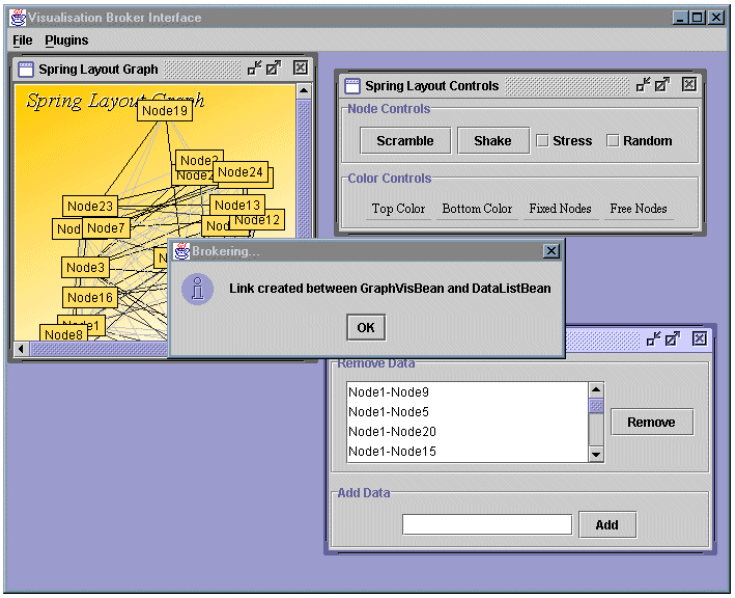

Figure 1 - Plug-in interface

Based on all of these issues the concept of a visualisation and interface framework has been developed. At the moment it relies on standard GUI technologies for the initial display and much of the control of the prototype, but the concepts still apply with smaller devices and other advances in technology. It provides a framework into which various data sources and visualisations can be registered, and attempts to broker between those the user wishes to use. This gives great flexibility and allows for an environment that can be customised (as far as the available plug-ins) for the user and task.

4. FLEXIBLE INTERFACE FRAMEWORK

This section describes the framework that has been developed, and is being refined through experience, for developing flexible visual analysis and understanding systems through the integration of

visualisations and data sources. Such a framework also has a place in interface technology. For the same reason that you need to give users both an easy route into using a system (through default displays) and also the ability to configure to their way of working as they gain more experience. This empowering of users means that they are more likely to accept the tool. From both an interface and a visualisation perspective this type of framework goes some way towards making the goal of information access for all a reality.

Similar research has been done for visualizations, such as the work on creating a database schemata dependent framework. *Snap-Together Visualization* (North) is a model that creates explicit interaction links between multiple views, and is based on relations and actions. Interactions with objects in one view cause the views at the other end of the relationships to also look that data up based on the action defined for that link, and thus changing an object in one view will update all linked views. This work differs in that it seeks to be more flexible. It does not rely on the presence or use of database technologies. It also aims to work at a more automated level for the linking and translation between data sources and visualizations. *Snap* requires that links are made explicitly which means that users need to be aware of databases and the use of keys to identify related data.

The framework is based on a plug-in technology that allows several types of plug-ins to be registered with the system. Once these plug-ins are known about then the system can utilise them in combination. At the moment this is under user direction, and there is very little in the way of data brokering (from a visualisation perspective) to ensure the images meet the data needs of the information. The next step is to utilise some degree of automatic personalisation, but still allow full user control should they wish. The concept of brokering can also be brought into play with users controlling actual linking of visualisations to data. This allows a given data source to know which of the registered plug-ins would be suitable for displaying it, and also for any visualisation to be aware of the actual data sources that are usable in its display.

Figure 1 shows a view of the prototype in action. This has two components plugged in. The top two windows are connected with the visualisation shown in the top left, whilst the bottom right displays the data source it is currently working with. It is appreciated that this interface is rudimentary and a standard GUI from an interface perspective, but it is theory that is important to consider. The fully specified framework will allow much more plug-in flexibility, to the extent of interacting with other plug-ins, the framework, and so on. It will also have some degree of automation to deal with personalisation and brokering of data. All of these are necessary for a flexible, personalisable interface to allow access for all.

5. FURTHER WORK

There are many ways in which this work can develop. The main direction is to incorporate linking with distributed provisions such as lookup and directory services to allow for utilisation of remote and/or third-party interface objects. There are obviously issues to do with licensing, actual abilities of the remote code, and security that need to also be addressed. There is also the concept of service provision which industrial initiatives such as .NET (Microsoft), J2EE (Sun – Java 2 Enterprise Edition) and JINI (Sun – JINI) are considering now. These are (in various guises) frameworks for developing distributed component based services and may provide mechanisms to solve some of the problems, such as security, which would be an issue for remote plug-ins.

6. CONCLUSIONS

This interface has presented a plug-in framework that allows for user freedom in the utilisation of visualisations and data sources to enable them to carry out analysis. Such a framework is necessary if the ideal of information for all is to be achieved because of the huge variability in user preferences. The flexibility to utilise different displays on different devices to suit the user and task is important. Empowering the user is a good way of achieving user satisfaction, and interface acceptance.

REFERENCES

Karat, J., Karat, C., and Ukelson, J. (2000). Affordances, Motivation, and the Design of User Interfaces, Communications of the ACM, Vol. 43, No. 8, August 2000.

Knight, C. (2000). Virtual Software in Reality, Ph.D. Thesis, Department of Computer Science, University of Durham, June 2000.

Kramer, J., Noronha, S., and Vergo, J. (2000). A User-Centered Design Approach to Personalization, Communications of the ACM, Vol. 43, No. 8, August 2000.

Microsoft (2001). Microsoft .NET Home Page, current page available on the world wide web with last known update in early 2001: http://www.microsoft.com/net/

Neilson, J. (1993). Usability Engineering, San Francisco, Academic Press Professional Publishing.

North C., and Shneiderman, B. (2000). Snap-Together Visualization: A User Interface for Coordinating Visualizations via Relational Schemata, Advanced Visual Interfaces (AVI) 2000, May 2000.

Pednault, E. P. D. (2000). Representation is Everything, Communications of the ACM, Vol. 43, No. 8, August 2000.

Sun (2001). Java™ 2 Platform, Enterprise Edition, current page available on the world wide web with last known update on 13th March 2001: http://java.sun.com/j2ee/?frontpage-javaplatform

Sun (2001). Jini™ Network Technology, current page available on the world wide web with last known update in February 2001: http://www.sun.com/jini/

XML (2001). XML.ORG – The XML Industry Portal, current page available on the world wide web with last known update on 12th March 2001: http://www.xml.org/