Developing Ambient Displays for Collaborative Work

Rod McCall[1], Benoît Otjacques, Fernand Feltz
Public Research Centre – Gabriel Lippmann
41, Rue du Brill
L-4422 Belvaux
Luxembourg
rodmc@acm.org,{otjacque,feltz}@lippmann.lu

This paper discusses a prototype system known as the Ambient Workplace (AW) that lets people monitor work patterns within a group of co-workers. The AW provides information in the users peripheral attention zone with the objective being that the user soaks up the information without needing to specifically attend to it. Information is displayed within standard applications such as the Windows Active Desktop, and in future Outlook or Word. At present the system displays various types of awareness information, for example the number of interactions a specific user or group of users have initiated. It also displays global information about the volume and type of work for the entire group. The AW uses Python and displays information using Scalable Vector Graphics (SVG). It uses a client/server architecture built upon the XMPP instant messaging protocol and retrieves information from a MySQL database. The objective was to build a system which can be easily ported to other platforms such as mobile phones and smart boards as well as to leverage existing libraries while allowing rapid prototyping of the user interface.

Introduction

The work patterns of many people have changed substantially since the rise of the Internet, with many CSCW (computer supported for co-operative work) tools emerging. Examples of such tools include Netmeeting, Skype, Blogs or on-line forums. The rise in the use of such tools combined with the changing working practices has altered the way that many people work together. For example closed-networks of people working together in offices have in some cases been replaced by large groups of people working on collaborative projects, where the work, group members and use of CSCW or IT tools changes frequently. Further issues arise as many people work in groups which are geographically spread.

The change in work practices has resulted in a situation where everyday cues are no-longer as relevant, examples include the number of documents in a persons out tray as an indication of how much work has been completed through to the sense that someone is at work but not at their desk via cues such as a jacket being left on the back of their chair. These two basic types of cues are important when we wish to ascertain if someone is ready to take on more work, or if they are going to be available for that all important chat later in the day. Furthermore it is comparatively easy to observe the working behaviours of a group of office workers based at the same location, however observing group work when the project is geographically spread is a more complex task, especially when there are a wide variety of electronic collaboration tools in use.

The work described in this paper discusses an early prototype of an ambient visualisation (known as the Ambient Workplace - AW) which informs people about the interactions of themselves and others, for example to provide information on whether someone has been busy. This is referred as the number and types of interactions that people undertake. It also explores the use of cues to make sure that people are aware of the presence status of other group members.

The main contributions of the work stem from the novel nature of what is being visualised, the system architecture and the visualisation. This is reflected in the structure of the paper which outlines the underlying theories used to develop the system, it then provides a description of the metaphor and

1 The work contained in this paper was developed during the tenure of an ERCIM Fellowship.

system architecture. As the system is currently in the early stages of development we have chosen not to provide code listings but instead to provide an overview of the architecture. The final section provides some suggestions for future work and a set of conclusions.

Background

Being kept aware of other people and their activities has been a key area of research within the CSCW community. For example Gutwin and Greenberg [1] explain that 'it is becoming more and more apparent that being able to stay aware of others plays an important role in the fluidity and naturalness of collaboration'. Dourish and Belloti [2] introduced the concept of awareness that they defined as 'an understanding of the activities of others, which provides a context for your own activity'. Ellis [3] argues that 'the philosophy of groupware is to encourage cooperation by making it known and instantly apparent to all who is sharing what with whom'. Humphries et al. [4] define activity awareness as 'knowing what has happened, what is happening and what will likely happen in the future over extended periods of time'. According to Greenberg [1] there are many forms of awareness:

- *Informal awareness of a work community is the general sense of who is around and what they are up to.*
- *Social awareness is the information that a person maintains about others in a social or conversational context. For example, whether another person is paying attention, their emotional state, or their level of interest.*
- *Group-structural awareness involves covers aspects such as the roles and responsibilities of the participants, their views on certain issues, their status and the processes within the group.*
- *Workspace awareness* includes information about *the identity of those in the workplace, their location, their activity, and the immediacy of changes with which others' activities are communicated.*

The work presented in this paper deals specifically with workplace awareness. To date a range solutions have been developed including: update notifications via email and displays indicating the number of people reading the same web page [5]. Other methods make use of graphics for example: the icons indicating who is on-line in MSN Messenger, avatars [6], representing users as abstract shapes [7], visualisations on mobile phones [8] and representing the other users actions in virtual environment [9]. Nearly all these solutions only provide awareness cues when people are using a specific application, for example MSN, moreover they frequently focus only on keeping people informed of the presence of others and not the nature or number of activities that are taking place. In order to overcome the first of these problems the AW provides the information within the Windows Active desktop, and as it develops within everyday applications. In doing so it draws on the ambient technology paradigm in the sense that the information is nearly always available, is unobtrusive and does not require the users specific attention. This approach builds on the idea introduced by Heath and Luff [10] of peripheral or 'out of the corner of the eye' awareness, i.e. awareness present in the periphery of people's attention. In order to overcome the second issue it is specifically concerned with displaying the nature and number of interactions either undertaken by individuals or the entire workgroup.

The Visualisation

What is Being Visualised

The AW visualises entities and their interactions. An entity is defined a resource, this can be a person, group of people or an artefact such as email or a shared document. An interaction is defined as the act of sharing information between these resources, for example sending an email to another person or adding something to an on-line forum. Within these categories we define interactions as being active or passive, mandatory or optional. An active interaction is one a person deliberately attends to, for example adding an item to a shared calendar, in contrast a passive one may be when a person scans the news headlines on the project website when their main objective is to click on a link to take them to another page. A mandatory interaction would be having to reply to an email from the team manager, where as an optional one would be a reply to a general request to join the office football team.

The visualisation is based on two types of awareness, *Workspace Individual Awareness (WIA)* and *Workspace Global Awareness (WGA*).

- WIA is where information about a specific resource or user is provided. Examples include notification via email that a resource has been updated, or displaying an icon about the presence status of an entity (e.g. on-line, away, busy).
- WGA is aggregated or anonymous information that refers to the general level of activities in the workplace. Aggregated awareness results from processing a set of individual information elements, but information about each individual element is not displayed. For instance, the number of users connected to the workspace is displayed but not their names. Anonymous awareness gives the user some feedback on what happens in the collaborative environment but does not explicitly indicate which entities it relates to. For instance, an icon is displayed when the user receives a message but the icon does not indicate the name of the sender or the subject of this message.

With respect to global awareness (*WGA)*, we have chosen to focus on aggregated information that makes the user aware of the level of activities within the collaborative platform. At this level, basic statistics are used, such as the number of connected users or the number of new documents since their last visit. This kind of information helps to communicate the volume of activity in the collaborative environment but gives little feedback on the quality of the collaboration. However, it is proposed that *WGA* should be expanded to include aspects about the quantity and quality of interactions. The composite indexes proposed by Otjacques et al. [11] provide a starting point in the visualisation of WGA and are outlined below.

- *Coopadex* (*electronic Cooperation Activity Composite Index*) index is defined as the mean number of computer-mediated interactions by a member of the group for a given period. It represents the *quantity* of interactions.
- *N-Coopidex* (*Normalized electronic Cooperation Implication Composite Index*) is defined as the normalized weighted combination of the number of interactions according to their cooperative nature. It is intended to be an indicator of the mean *quality* of the interactions from a collaborative viewpoint.
- *Glocoopex* (*Global electronic Cooperation Composite Index*) is defined as the product of *Coopadex* and *N-Coopidex*. It provides a *global* metric of the cooperation from a quantitative and qualitative point of view.

The *Glocoopex* index provides one measure of *WGA*, however a visualisation is more useful if it also provides information about individual entities (i.e. *WIA*). For this the system focusses on monitoring the actions of individual users and not the modification of resources. In order to address this we have chosen to display the number of interactions that a given person or entity has undertaken during a recent time period (e.g. last week). This allows users to see if the entities they have chosen to monitor have initiated many interactions and whether the nature of those interactions. Another important aspect that supports WIA is the ability of people to communicate with others and to view the resultant interactions within a visualisation. .

Use Case

During the early stages of the research a decision was taken to identify a possible group of end-users, in order to address this need we enlisted the help of participants within a large European Union funded Network of Excellence (NoE). An NoE is a virtual research community which consists of a diverse range of partners such as universities and businesses. Within these organisations there exists a wide range of users from students through to project managers, each with varying knowledge of and interest in using CSCW tools.

The NoE chosen as the use case places substantial emphasis on the creation of a virtual research community. For example it has its own website with forums, news articles and places where people can upload files. Moreover it hosts regular Ph.D. schools. However despite the best intentions of the project managers the electronic features remain comparatively underused. The intention of the system presented in this paper is to encourage the use of such tools by making sure people are kept aware of the actions of others.

The NoE website, forums and other electronic collaboration features are driven by a database which stores information about the participants and their interactions. It is the information in this database which the AW uses in order to ascertain the nature and number (also referred to as the quality and quantity) of interactions that people are undertaking. As part of our work the database has been modified to include information as to whether the interactions are mandatory or optional and active or passive.

The Design

A maritime flags metaphor was chosen as the graphics are comparatively simple, do not require animation and the basic concept behind such flags is readily understood by many people. This is not to say that most people understand the content of maritime flags, but rather that people are aware that information is contained within them. In common with the work on informative art by The Viktoria Institute in Sweden[12] the designs are quite simple, and once understood most people should be able to interpret the data quickly and easily.

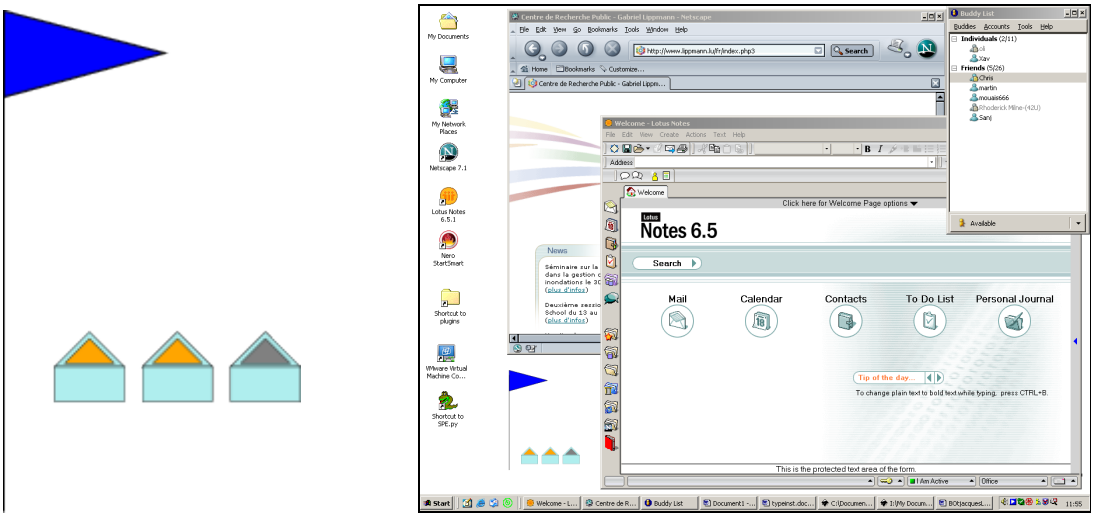

Figure 1: Left illustrates the flag and beacons. The flag represents global awareness where as the beacons represent individual awareness. The image on the right illustrates the same visualisation as part of the Windows Active Desktop.

Figure 1 provides a view of the visualisation with the large blue flag representing WGA and the beacons representing the WIA. The *Glocoopex* value is represented by the colour of the flag, the darker blue indicating a higher value. The length of the flag represents the number of electronic co-operations (Coopadex), the longer the flag the more interactions. The height of the flag represents the nature of the interactions (N-coopidex), for example whether they are mandatory/optional or active/passive.

The beacons provide an indication as to the number of interactions initiated by a specific entity. In figure 1, the two beacons on the left indicate that the entity has initiated some interactions, where as the one on the right indicates they have initiated none. As the system develops the beacons will also provide information about the on-line status of each entity.

Refining the Design

A four phase study is currently on going taking place with the intention of improving the design of the visualisation. The first part consisted of asking twenty one people to draw sketches which represented the various aspects of the visualisation, including individual components (i.e. the indexes) and to draw the entire visualisation. The second phase is currently ongoing and for this a smaller group of people

are asked to comment on and improve the designs from phase one, with a view to selecting a short list or developing a new design which they would like to see implemented.

Preliminary results from the first study were mixed and depended heavily upon whether the participants were asked to draw either the complete visualisation or the individual components. The participants thought that the early design was heading in the right direction, however there was clearly some room for improvement. Typical indications were that for the complete visualisation people preferred beacons to represent the on-line status of users and their interactions (WIA) and flags to represent the index values (WGA). A more detailed description of the this study can be found in [13].

Overview of System Architecture

The AW consists of a number of core components, including a client and server (BOT). The AW components are in turn linked with a number of parts built by third parties such as MySQL, Wildfire server and GAIM. A description of these and a diagram (figure 2) are provided in the remainder of this section.

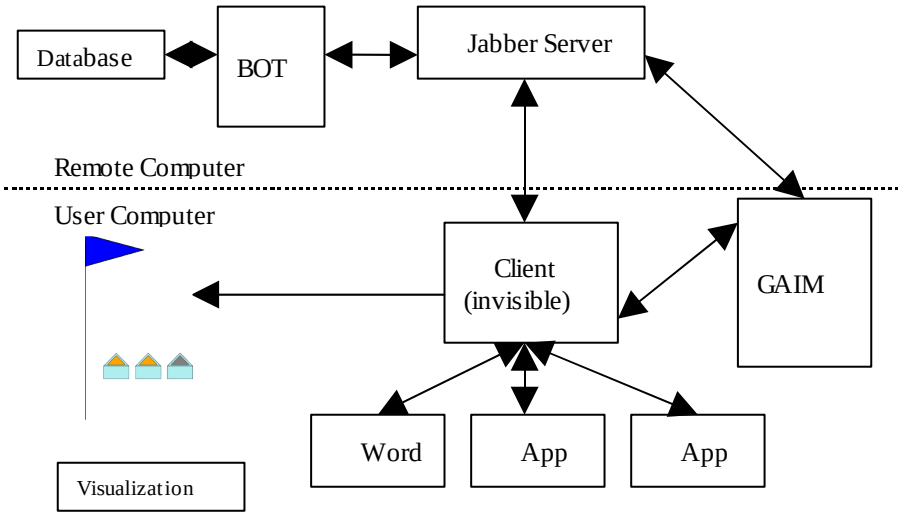

Figure 2 The system architecture used in the Ambient Workplace.

A summary of each component is provided below, however a more in depth discussion of the client and server are presented later.

- **Jabber Server:**[14] this uses the XMPP instant messaging protocol, it provides user login/out services and presence support (i.e. indicating if the user is on-line, away or busy). Privacy is an issue and as the AW develops we plan to use the presence aspects of the Jabber server to let users control which information and with whom they share it. Messages sent to the clients (users) utilise a combination of the body and subject fields found in standard XMPP messages, this makes it possible to convey a wide range of information. At present the system uses the free (GPL) version of Wildfire server from Jive Software. A full description of Jabber/XMPP is beyond the scope of this paper, but more information can be found on-line [15].
- **BOT (server):** rather than customizing the Jabber server an automated BOT responds to requests for information (or message stanzas) from other users or other parts of the system. The BOT server behaves like an automated user and can retrieve information from the database and handle message requests from users.
- **GAIM Client:**[16] this is an open-source instant messaging client that is compatible with most popular IM platforms. This means that users only need one client to access every major IM platform and the Active Workplace.
- **Database:** the database contains the information about the members of the project and their interactions. This is implemented using PHP and MySQL.

- **Client:** each user has a client application which runs on their computer. This is responsible for sending and receiving messages to/from the Jabber server and GAIM client. It is also responsible for generating the visualisation and refreshing the Windows Active Desktop. The client application has no GUI and subject to user approval will start up automatically when they login to their computer (this has yet to be implemented). This is consistent with the idea of the system operating in the background without the need for user intervention.
- **Visualisation:** the visualisation is created by the client application, hence it requires no intervention by the user. The actual visualisation consists of a mix of HTML and SVG data, which should mean it can be displayed on most platforms.
- **Third party applications**: although not implemented at present the system will interface with applications such as Word, Internet Explorer and Outlook.

The database sends and receives MySQL queries and their results. The BOT server communicates with the database and Jabber servers using XML messages based on the Jabber/XMPP protocol. The Jabber server communicates with the client and GAIM using XML messages. Communication between GAIM, all third party applications and the client will either use Windows 32 API calls or sockets.

The client and BOT server are written in the Python programming language and use a range of standard libraries to enhance performance and provide relevant features. These include mySQLdb [17], a cross-platform GUI development kit (wxPython) [18], an XMPP library for Python (XMPPPY) [19], a network protocol package (DNSPython) [20] and PyWin32 [21] which provides access to the Windows 32 API. GAIM can be extended via C++ (it's native language) or via Python using the PyGAIM [22] extension. Additional extensions are used to enhance performance and create executable files. The visualisation is displayed using the Adobe SVG plug-in for Internet Explorer.

Description of BOT Server

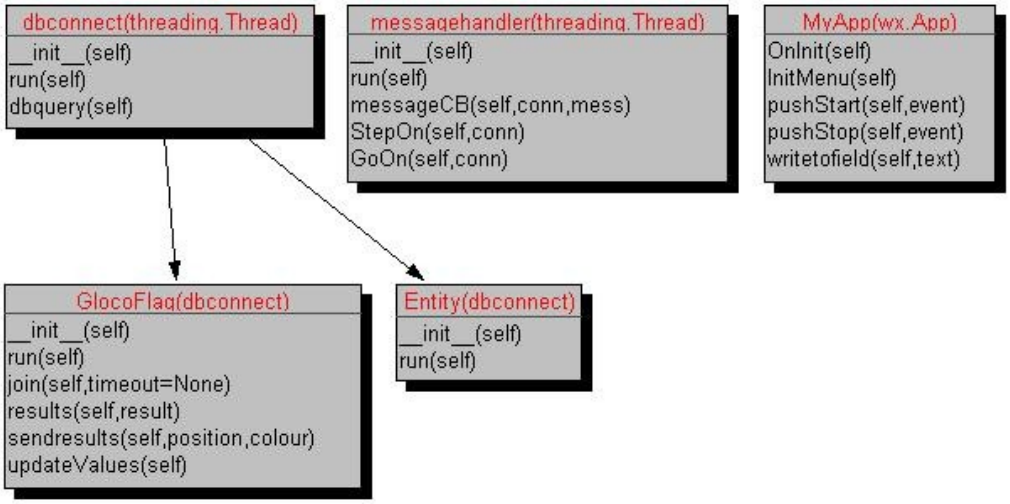

Figure 3:A UML diagram indicating the structure of the BOT server.

The server (BOT) behaves like the automated agents found in many on-line chat systems. Its basic role is to send and receive messages as well as obtain information from the database. The UML diagram (figure 3) indicates the class structure used within the BOT, in addition there are also a number of message handler functions which are activated when a particular message is received. The MyApp class provides a basic user interface which is used to start and stop the server.

The BOT uses a threaded architecture so that it is able to process messages while other tasks, such as retrieving information the database are taking place. The main classes are dbconnect and messagehandler. The dbconnect class provides the basic structure for connecting to the database, this in turn is instantiated through two subclasses GlocoFlag and Entity. Glocoflag is responsible for

retrieving information related to the flag (see figure 1) where as Entity retrieves information related to the on-line status of other users or groups of users (beacons). Both the flags and beacons are updated on a regular basis.

The Glocoflag object sends a string to the client containing the graphics required to render the flag, this consists of SVG XML data. In order to generate this information it queries the database for three sets of information, the value of Glocoopex, Coopadex and N-coopidex each for a given period of time, for example seven days. Three lists are created which store the retrieved values. A mean value is then taken from each list and the size and colour of the flag computed.

Obtaining information about the various entities is a more intensive task, for example at present there are around fifty entities (or members) of the Network of Excellence. This could potentially result in each entity sending a message to every other entity, each one containing an SVG graphic representing its on-line status and interactions. In order to overcome this problem two strings are sent each containing integer values which represent the on-line status and interactions for *all* the entities. This reduces the number of messages being sent to convey this information by around 95%.

The sending and receiving of messages is handled by the messageHandler class and a series of functions. There is a login function which is responsible for logging the BOT into the Jabber server. The messagehandler class then parses the message. Based on the contents of the subject field the relevant handler is called, for example a message with subject “Gloco” (short for Glocoopex flag) will activate the routine for drawing the flag. New handlers can be added by creating a function which corresponds to the message subject.

Description of Client Application

The client application shares a number of architectural features with the server, for example it makes use of similar code to handle messages and threads to ensure the updates, screen redraws, receiving of data and other aspects can take place simultaneously. The main additions are the inclusion of classes to support graphics.

At present the system creates a scene graph using raw SVG (XML) code. The visualisation is represented by the scene object which in turn contains the individual elements. The core objects (or elements) are beacon, svgShape and itemgroup. The beacon object represents the individual entities, svgShape is a generic function, this receives objects from the server then adds them to the scene, in theory this function can be used to add any type of graphic to the scene. At present this is only used for receiving the flag data. The itemgroup object is used to store and place groups of related items, for example the three beacons (see figure 1). At present it does not make use of XML specific functions but simply creates and modifies a series of strings. The updating of the graphics is handled by the thread based displayFlag class which periodically checks if an update is required. Although this is currently pre-set future versions will allow the user to specify the update period.

Using Python

The decision to use Python was not an easy one, not least because the authors had never used Python to any extent. Moreover, CRP- Gabriel Lippmann prefers to use Java for development work. The factors which led to the decision to use Python included the desire to experience a new language, the 'promises' made in many newsgroups by Python enthusiasts, availability of libraries and the ability to undertake rapid prototyping.

Python and the associated libraries allowed us to develop the prototype very quickly, the language syntax meant that many of the more unpleasant aspects of Java and other tools were avoided. More over the range of libraries available, in particular those supporting the Jabber and MySQL protocols proved both easy to use and for the most part reliable. The community of users and the range of help sites available proved invaluable as did the use of open source code.

Although our experience was a positive one, the main areas of weakness were the problems involved in shipping free standing applications. Many of these problems were a direct result of the status of Py2Exe which is currently around version 0.6.x. In many cases these problems will be solved with time and overall our experience of using Py2Exe in combination with Inno Setup [23] was positive. There

were also problems with programs running correctly within one development environment and not another, or them refusing to run as stand alone applications. Other problem areas when compared to other tools include the lack of high quality and GUI design tools for wxPython, while some are available again they are in a state of continual improvement or in some cases are very expensive.

Future Work

The prototype is currently in the very early stages of development and a number of improvements need to be made for example both the client and server need to be modified to run as Windows services. New features such as support for third party applications need to be added as well as a control panel which will let users select which data they wish to display. New flags or beacons will be added to represent other information such as the number of posts or emails left to read.

From a technical viewpoint much of this work was carried out while the main developer was learning Python for the first time. Hence it is acknowledged that many improvements are required. These include improving the efficiency of the various algorithms within the client and BOT. In addition there is a need to improve the thread synchronisation techniques and SVG/XML data handling to make them more compliant with standard practice, although as yet there no known problems with either issue.

Conclusion

This paper has presented an overview of an ambient display built using Python to inform people about the work patterns of a group of co-workers. Python allowed the successful development of an early prototype due to the open source nature of the vast array third party libraries. Moreover the language made it comparatively easy to develop the prototype and make changes quickly. The object oriented threaded nature of the prototype, combined with the fact that the system is built upon the Jabber messaging protocol means that it is easy to add new features. The authors or this paper are in many ways new to Python, hence this paper has charted a comparatively successful attempt at both learning a new language as well as innovating a new technology.

From an technical viewpoint the architecture used within the system should be of use to other who are seeking to develop ambient displays. For example, the emphasis is on re-using software and systems which are already in existence, for example the use of Jabber as the messaging platform. Also the extensible nature of using a messaging platform such as Jabber means that adding new features should be much easier than having to write a whole new server application capable of delivering such messages. Moreover, the emphasis on using open standards such as SVG means the system does not rely on any proprietary formats, even the Wildfire service could replaced with a totally free alternative if required.

From a scientific perspective the approach in this paper is novel in the sense that there are few other systems which visualise similar information, indeed it is our view that the metrics presented and visualised are unique to the project in which they were developed. Moreover the early design studies while suggesting there is room for improvement pointed to people agreeing with the basic design ideas.

In summary this paper has presented an overview of the Active Workplace with the aim of informing people about its academic rationale, design and technical architecture. These areas are of use to other developers, for example the academic rationale (indexes) will be useful when developing other environments which seek to visualise co-operation. The design of the visualisation will be useful to those developers who are looking for ways to represent information using a comparatively clean, simple and ambient metaphor. Whereas the architecture represents an interesting way to leverage existing technologies in way which permits developers to focus on the rationale and design to a greater extent than the underlying technical considerations.

Acknowledgements

The authors gratefully acknowledge the assistance of the members of the Network of Excellence highlighted within the paper as well as other staff at CRP – Gabriel Lippmann. The authors would also wish to thank the members of the Python community, in particular those who have written the libraries which we have used to create the visualisation. We are also indebted to those who have provided assistance or examples upon which we have based some of the software.

References

1. Greenberg, S., Gutwin, C., and Cockburn, A. Awareness Through Fisheye Views in Relaxed-WYSIWIS Groupware. In Proceedings of the Graphics Interface 1996 Conference (Toronto, Canada, May 22-24, 1996).
2. Dourish, P. and Belloti, V. Awareness and coordination in shared workspaces. In Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW'92) (Toronto, Canada, October 31 - November 4, 1992). 107-114.
3. Ellis, C. Keepers, Synchronizers, Communicators and Agents. In ACM SIGOIS Bulletin, April 1995, Vol. 15, N° 3, 10-14. Special issue: workshop write-ups and positions papers from CSCW'94.
4. Humphries, W. D., Neale, D., and McCrickard D. S. Laboratory Simulation Methods for Studying Complex Collaborative Tasks. In Proceedings of the 48th Annual Meeting of the Human Factors and Ergonomics Society (HFES'05) (New Orleans, Louisiana, September 20-24, 2004). 2451-2455.
5. Robinson, M., Pekkola, S., Korhonen, J., Hujula, S., Toivonen, T. and Saarinen M.-J. O. Extending the Limits of Collaborative Virtual Environments, in Churchill, E.F., Snowdon D. N. and Munro A. J. (eds): Collaborative Virtual Environments, London, UK, Springer, 2001.
6. The Palace User Software Guide for Windows, version 3.5, http://www.palacechat.us/documentation.php, accessed 24 March 2005.
7. Donath, J., Karahalios, K., and Viegas, F., Visualizing Conversation. In Proceedings of 32th Hawaï International Conference on System Sciences (HICSS-32), (Maui, Hawaii, January 5-8, 1999).
8. Bardram, J.E. and Hansen T.R. The AWARE Architecture: Supporting Context-Mediated Social Awareness in Mobile Cooperation. In Proceeding of the ACM Conference on Computer-Supported Cooperative Work (CSCW'04) (Chicago, Illinois, November 6-10, 2004). 192-201.
9. Pinho, M.S., Bowman, D.A., and Freitas C.M.D.S. Cooperative Object Manipulation in Immersive Virtual Environments: Framework and Techniques. In Proceedings of the ACM symposium on Virtual Reality Software and Technology (VRST'02). (Hong Kong, China, November 11-13, 2002). 171-178.
10. Heath, C. and Luff, P. Collaborative activity and technological design: task coordination in London Underground control rooms. In L. Bannon, M. Robinson and K. Schmidt (eds.) Proceedings of the 2nd European Conference on Computer-Supported Cooperative Work (ECSCW'91) (Amsterdam, The Netherlands, September 24-27, 1991). 65-80.
11. Otjacques, B., Noirhomme M., Figueiredo, J., and Feltz, F. (2006), Composite Indexes as Metrics of Cooperative Activity. In 11ème Conférence de l'Association Information & Management (AIM 2006), (11th French-speaking Conference on Management of Information Systems) (Luxembourg, June 7-9, 2006).
12. Informative Art Project, Viktoria Institute Sweden, http://www.viktoria.se/fal/projects/infoart/, accessed 16th June 2006.
13. Otjacques, B., McCall, R. and Feltz, F. An Ambient Workplace for Raising Awareness of Internet-based Cooperation. In Proceedings of The Third International Conference on Cooperative Design, Visualization and Engineering (CDVE'2006). (Mallorca, Spain, September 17-20th 2006) (in press).
14. Wildfire Jabber Server http://www.jivesoftware.com, accessed 16th June 2006.
15. The XMPP Messaging Protocol http://www.xmpp.org, accessed 16th June 2006.
16. Gaim Instant Messaging Software, http://gaim.sourceforge.net/, accessed 24 March 2006.
17. mySQLdb http://sourceforge.net/projects/mysql-python, accessed 16th June 2006.
18. wxPython GUI Library, http://www.wxpython.org, accessed 16th June 2006.
19. XMPPPY library for Python http://xmpppy.sourceforge.net/, accessed 16th June 2006.
20. DNS Python Toolkit http://www.dnspython.org/, accessed 16th June 2006.
21. PyWin32 Extensions http://starship.python.net/crew/mhammond/win32/, accessed 16th June 2006.
22. pyGaim extension for GAIM http://pygaim.sourceforge.net/, accessed 16th June 2006.
23. Inno Setup installer for Windows http://www.jrsoftware.org/isinfo.php, accessed 21st June 2006.