Intrusiveness Management for Focused, Efficient, and Enjoyable Activities

Fredrik Espinoza[1], David De Roure[2], Ola Hamfors[1], Lucas Hinz[1], Jesper Holmberg[3], Carl-Gustaf Jansson[3], Nick Jennings[2], Mike Luck[2], Peter Lönnqvist[3], Gopal Ramchurn[2], Anna Sandin[1], Mark Thompson[2], and Markus Bylund[1]

[1] Swedish Institute of Computer Science (SICS), Kista, Sweden
[2] Department of Electronics & Computer Science, University of Southampton, United Kingdom
[3] Department of Computer & Systems Sciences, Stockholm University and the Royal Institute of Technology, Kista, Sweden

1 Introduction

When technologies for distributed activities develop, in particular the rapidly developing mobile technology, a larger part of our time will be spent connected to our various distributed contexts. When we meet physically we bring technology, both artifacts and services, which enable us to participate in these non-local contexts. Potentially this is a threat to focused and efficient activities due to the intrusiveness of the technology. Our aim is to contribute to the restoration of a number of the desirable properties of traditional local technology-free contexts. The intrusiveness itself is caused by at least four typical phenomena that have influenced current technology:

- Focus-demanding and clearly distinguishable artifacts like phones or PCs explicitly mediate interaction with the distributed context
- The functionality of services is traditionally based upon the assumption that communication is a deterministic flow of passive information, which for example, does not include information of the participants´ current context
- Services in general perform individually and without coordinated communication schemes
- The switches between contexts introduce a high cognitive load as each distributed context typically has its own system of characteristic objects and rules.

In the FEEL project, we have developed a system called "Focused, Efficient and Enjoyable Local Activities with Intrusiveness Management" (FEELIM) that constitutes an intermediate alternative between the technology-dense and technology-free environments, which addresses the problems cited above. This research is based on a collaborative and cooperative setting where problems of intrusiveness management are confounded by several users meeting and cooperating together as opposed to isolated users dealing with similar problems of interruption management (Chen 2004; Ho 2005).

N. Streitz, A. Kameas, and I. Mavrommati (Eds.): The Disappearing Computer, LNCS 4500, pp. 143 – 157, 2007.

2 Objectives

Usability in Physical Spaces

The design of DC environments in the form of physical spaces that support non-intrusive mechanisms for notification, supporting both local and distributed work is important. Usability studies in such physical spaces were conducted, with the following sub-objectives:

- Capturing concrete scenarios for use of integrated local and distributed services
- Making preliminary studies of user needs for the disappearing computer scenario
- Evaluating the usability aspects of concrete disappearing computer environments, in particular the users' experiences of the emergent functionality
- Evaluating the usability of the integrated local and distributed services in the disappearing computer environment (non-intrusive services).

These research issues were thus investigated in interactive environments with multimodal characteristics, which can enable mechanisms for handling intrusions. This means physical interactive spaces with a rich set of peripherals and computing devices integrated in the interior decoration) and where small wearable/personal devices can interact dynamically with the stationary/public devices to create a total computing and communication DC environment.

3 Principles of Design

We have identified general design principles that can prevent or diminish many of the negative effects of intrusiveness phenomena resulting from work in computer dense environments.

3.1 Creating Adequate Explicit Models of Work Environments

At the core of mechanisms for diminishing intrusiveness phenomena lay good coordination strategies. These in turn must rely on adequate models of the tasks being performed in parallel in each work environment (both in large and small scale), the characteristics of the participants, the physical properties of the work environment and the events that occur. We need also explicit and operable representations of these models, such that observations of a variety of contextual factors can be made and mapped onto those representations. One particular problem with the handling of parallel tasks is the cognitive load when switching between tasks. If the services supporting different tasks can minimize the effort in switching between tasks, the effects of intrusions will decrease. The implementation of such functionality in services in turn has to be based on better modelling of tasks and work situations. This also holds for the personalization of work environments, facilitating the configuration or restoration of an appropriate work environment suitable for a particular group.

3.2 Modelling Intrusiveness and Non-intrusiveness

This modelling should cover:

- the social rules valid for a certain situation,
- the normal behavioural and perception patterns within a group, and
- the creation of rough operational initial models.

McFarlane (1999) was the first to dissociate the notion of intrusiveness from the notion of interruption. He defines intrusiveness as the degree of interference with the realization of the main task of a group caused by a number of intrusions. In turn, an intrusion is defined as an occurrence of a process or event that is not intimately related to the current task of a group and that interferes with the realization of that task. It needs to be pointed out that interruptions and intrusions are clearly distinct concepts. According to McFarlane, intrusions are errors where people incorrectly perform actions from a pre-interruption task after task switching while interruptions are methods by which a person shifts his focus of consciousness from one processing stream to another. Whatever the form in which a message is received, according to Clark (1996), people have four possible responses then:

1. take-up with full compliance – handle the interruption immediately.
2. take up with alteration – acknowledge the interruption and agree to handle it later.
3. decline – explicitly refuse to handle the interruption.
4. withdraw – implicitly refuse to handle the interruption by ignoring it.

3.3 Support for Local Collaboration

To remedy the existing imbalance between technology support for shared local tasks and private and often distributed tasks the hardware and software support for local collaboration must be improved with respect to:

- Shared focus by co-use of public devices like interactive walls or tables.
- Simultaneous interaction by co-use of input devices such as mouse and keyboard.
- Transparent interaction in the sense of uniform mechanisms to interact with information elements across physical and virtual boundaries.
- Ad-hoc device integration. The different entities in the room should not be considered as separate artifacts but rather as components in a coherent dynamically configured system. This should also include the personal artifacts being used in the room.
- Personalization of the work space both during and between work sessions.

A lot of work has already been done in order to try and create environments like this, for example in the *Interactive Workspace* project at Stanford University (iwork.stanford.edu) (Johanson et al. 2002) or in the *i-LAND* project at GMD-IPSI (later Fraunhofer), Darmstadt, Germany (www.ipsi fraunhofer.de/ambiente/) (Streitz et al. 1999, 2001) but none of them has explicitly targeted the issue of intrusiveness.

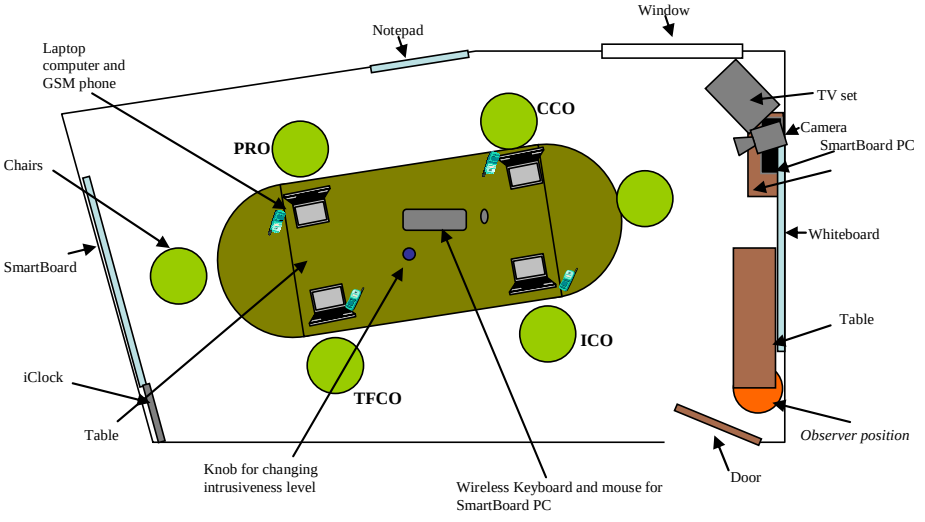

Fig. 1. The physical environment in the 15 square meter room 7517 "Pentagonen" at the IT University in Kista, furnished with chairs and a table catering up to six people. For the shared focus of the experimental groups the room is equipped with a front projected wall-mounted SmartBoard. TFCO, PRO, CCO and ICO represent team members using the room. Specialized hardware devices such as the iClock are described in following sections

4 Implementation

The FEELIM implementation has been build on top of the sView service architecture, see (Bylund 2001; Bylund and Espinoza 2000). The implementation is not a single monolithic component but rather a set of principles and protocols that service modules adhere to as part of their agreement to be FEELIM conformant. The next section describes the sView platform and this is followed by a description of the main FEELIM services.

The sView Service Platform

The sView service platform is an electronic service environment with heavy focus on the individual user. It enables a user to collect, store, and run electronic services, locally, or in a distributed fashion. The whole purpose of sView is to serve as a common area for the services to cooperate amongst themselves and to provide a unified access method to the services for the user. To a developer of services, the sView platform is characterized by openness, modularity, security, and a high degree of flexibility. To a user of the platform, it is accessible in many ways and available continuously.

The system assumes a client/server model, but instead of having a uniform client without service specific functionality for access to all servers (as in the case with the World Wide Web), access to the servers is channelled through a virtual service briefcase. The briefcase in turn supports access from many different types of devices and user interfaces (Nylander and Bylund 2001, 2002). It is also private to an individual

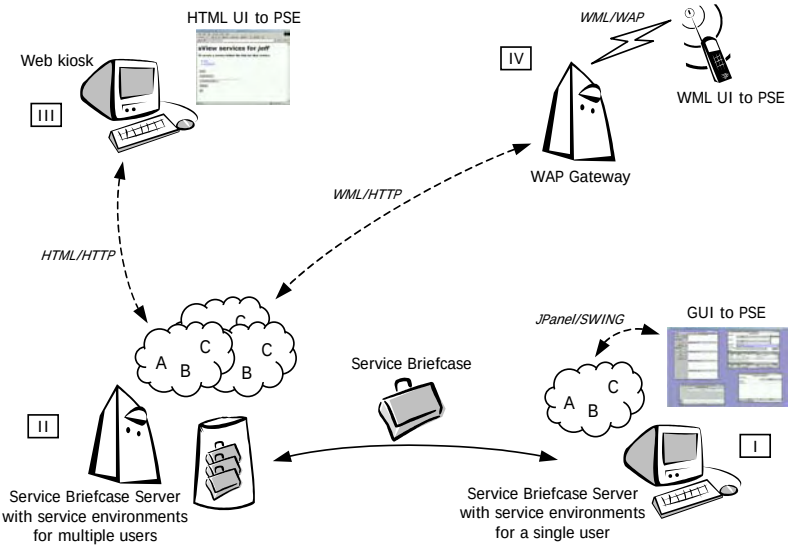

Fig. 2. The main parts of the core sView specification and their relations

user, and it can store service components containing both service logic and data from service providers. This allows the service provider to split the services in two parts. One part provides commonly used functionality and user-specific data that executes and is stored within the virtual briefcase. The other part provides network-based functionality and data that is common between all users. Finally, the service briefcase is mobile and it can follow its user from host to host. This allows local and partially network independent access to the service components in the briefcase.

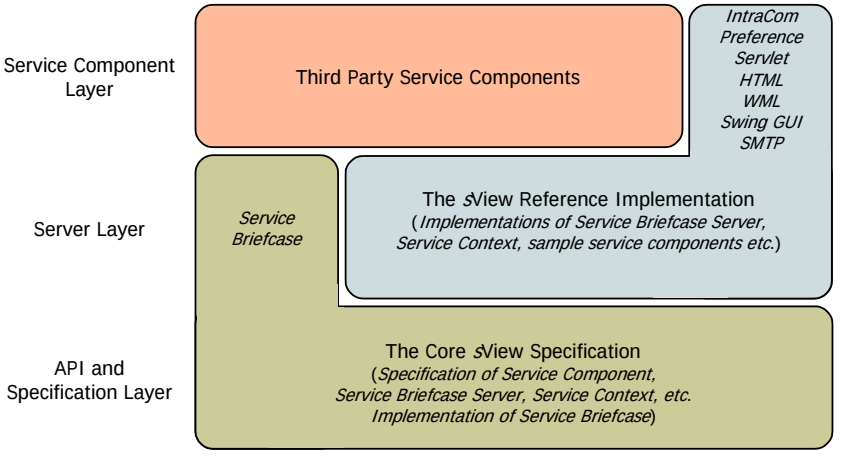

Fig. 3. A schematic overview of the sView system in its three layers

The *core sView specification* provides Application Programming Interfaces (APIs) to developers of service components and service infrastructure that builds on sView technology. Implementing these APIs and adhering to the design guidelines that accompany the APIs, assures compatibility between sView services and service infrastructure of different origin (cf. Figure 1). The *sView reference implementation* provides developers with a development and runtime environment for service components as well as a sample implementation of an sView server. An illustration of how the core specification and the reference implementation are organized in three layers can be seen in Figure 2.

5 The Network Briefcase Lookup Server

The sView platform allows users' service briefcases to move around the network, from server to server (these servers are called sView Enterprise servers and they can house many user's briefcases at once) depending on where in the world and the network the user may be. Since services for a user should always run somewhere, to stand ready in case the use should call, these servers must be situated in the network in computers that are constantly available. This moving around of a user's briefcase of services becomes a problem, however, when someone or something tries to contact the user through one of the services, since it is impossible to know where the briefcase is at the moment. To remedy this problem we have built the Network Briefcase Lookup Server (NBLS).

The NBLS is a server which runs in a well know place in the network. All sView and FEEL sub-systems are aware of its network address and use it to ask for directions when trying to locate a specific user. Users' briefcases all register their own network addresses with the NBLS as soon as they change. Consequently, the NBLS has a constant record of the whereabouts of each user's briefcase, and can forward requests for service or action to the correct one from any incoming request, much like a dynamic DNS provider.

This server can route requests it receives to the appropriate user's service briefcase, and within this, to the pertinent service, regardless of where the user and his or her briefcase are located in the real world and in the network. If the NBLS is unable to contact the user's briefcase it should return an appropriate response and the querying service should also react accordingly. The address tracking function is the main function of the NBLS, but as we will see below, it also performs several other important functions.

6 Implementing Mechanisms as Services

Conceptually, the implemented mechanisms integrated in what we call FEEL software technology consist of four distinct functional tasks, each described in more detail below:

- Filtering
- Routing

- Notification
- File sharing

6.1 Filtering and Scheduling Services

The Sentinel Intrusiveness Controller[1] (Sentinel) is used to establish a common level of agreed upon intrusiveness for an ongoing meeting. The intrusiveness-controlling mechanism is described by the behaviour of the Sentinel in conjunction with the end-user services it controls, as these cooperate to achieve intrusiveness management (see Figure 4). The filtering of notifications relies on the intrusiveness level set in the Sentinel. A high intrusiveness allows for messages to be displayed immediately and in an attention-grabbing manner, whereas a low intrusiveness hinders the message from being immediately displayed, or makes it display less conspicuously.

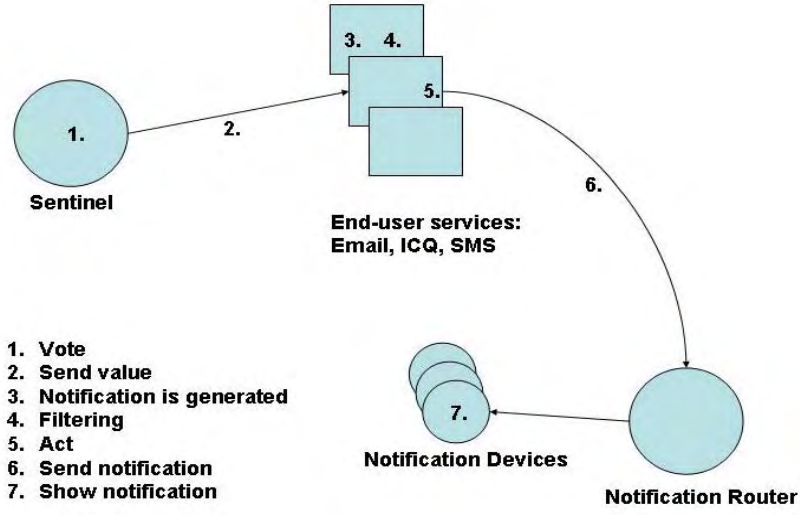

Fig. 4. Sentinel intrusiveness control schema using sView, the Sentinel (in voting mode), and the end-user services, which enable the choice of notification and filtering

6.2 Routing Services

We believe that receiving notifications about messages is less intrusive than receiving real messages. The purpose of the Communication Agent is to act as personal central hub for incoming communications. The Communication Agent does not show the actual content of the incoming messages or phone calls. Instead it decides, based upon the Sentinel value, if the communication is allowed to pass through, or not; if the message is not allowed to pass through the user gets a notification instead (cf. Figure 5). The characteristics of the notification are varying, depending on the current intrusive-

[1] One that keeps guard; a sentry. Dictionary.com. The American Heritage Dictionary of the English Language, Fourth Edition. Houghton Mifflin Company, 2004. http://dictionary.reference.com/browse/sentinel (accessed: January 24, 2007).

ness level. If the user is in state A (60-100%, high intrusiveness), a notification is presented in the graphical user interface, and the real communication is let through. The notification contains information about the sender, the message type, and the time it arrived. If the user is in state B (30-60%, medium intrusiveness) the notifications (but no content) are visually presented in the graphical user interface, and on any available public display. The real communication is blocked and queued. Furthermore, the sender receives an auto-reply saying that the message has been queued and will be delivered to the user at later stage. Finally, if the user is in state C (0-30%, low intrusiveness), the communication is blocked and queued, as in the medium state, and auto-replies are sent out. A personal notification is made in the Communication Agent but no public displays are used. The partitioning of three discrete intrusiveness levels was chosen as a reasonable trade-off between an understandable model which would work with users in user tests and a sufficiently interesting span of intrusiveness degrees for demonstrating our findings.

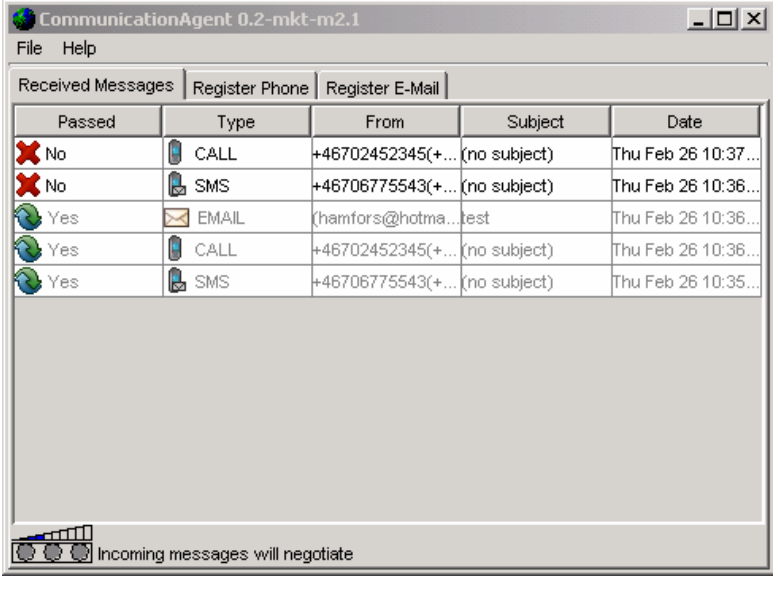

Fig. 5. The Communication Agent, running inside sView in the user's briefcase

6.3 Notification Services and Public Displays

Not all notifications take place within the sView briefcase, which runs on the individual computers of the meeting participants. In some states, notifications are sent to "real world" terminals on public displays. The iClock is implemented as a Java application where the user interface resembles the face of an old-fashioned analogue clock. The clock is publicly displayed on a wall using a small flat screen panel. The iClock is designed to display notifications for incoming messages for all users in its vicinity. When a message is received, the clock display fades out and changes notification mode. In this mode the message is displayed as text. The actual content of the message is not displayed, only, the sender, and the intended recipient and the time stamp, and a

notification sound is played. This enables users to notice incoming messages with a minimum of disturbance since the notification sound is very short and the content of the message is withheld. If several notifications arrive at the same time, they are queued by the clock and displayed one after the other in the proper order. In addition to displaying messages, the iClock is aware of the Sentinel states, and the display is coloured in accordance with the state (green for state A, yellow for state B, and red for state C, respectively). The iClock can also be configured to know when messages should be displayed or not, for example only allowing received messages to be displayed in state A.

6.4 File Sharing Services

Zenda is a file-sharing service. It is compatible with sView, but can also run as a stand-alone application. It permits users to start files on each other's computers, using the underlying functionality of the operating system to launch the appropriate application when a file is received. For example, receiving a PDF document will launch Acrobat Reader on the receiving computer. Technically, each running Zenda service instance (running on individual user computers) works as a fileserver, dealing out files, and as a client, receiving files from other Services. Communication between involved users computers for this purpose is enabled by the Jini middleware.

7 The FEEL Intrusiveness Management System

The *FEEL Intrusiveness Management system (FEELIM)* has progressed through iterative development from a set of individual and basic components (the underlying service platform, display devices, negotiation algorithms, etc.) to a fully functioning and demonstrable prototype. Note that FEELIM to a large extent is not visible to the user at all: most of the technology is at work behind the scenes in servers in the network.

The basic intrusiveness management is based on location: where a user is presently located, where a room is located, and so on. An administrator may specify intrusiveness rules for a location, say a meeting room, or the intrusiveness rules may be specified by users themselves when they are in a certain location.

The position of a user is determined by a positioning system based on wireless LAN base station lookup from the SICS *Geonotes* project (Espinoza et al. 2001). This system achieves an accuracy of 10-100 meters since it is based on identifying the position of the closest connected base station, which is registered in a database. The system is further enhanced by allowing users of the system to define arbitrary *place labels* (Espinoza et al. 2001; Fagerberg et al. 2003) tied to any position. These serve to further qualify the position and enables the positioning system to achieve an arbitrary level of accuracy. The position of the user is made available to services in sView through the Position service (see Figure 6). Any other service may subscribe to the Position service for notifications when the user’s position changes.

The management of intrusiveness level is based on location, since the location is used by the Sentinel to determine the intrusiveness level. The position or location of a user is of course only one in a probably endlessly large set of possible properties which could be used to influence the intrusiveness management. Location, however,

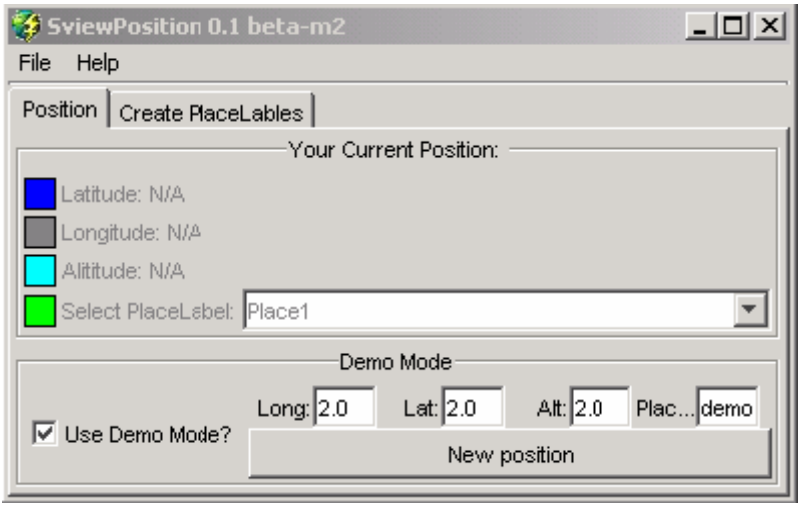

Fig. 6. The sView position service

is easily understood by users and it is often used for conceptual partitioning in real life (stay on this side of the line, park within the parking space, no cell phone use inside the movie theater, borders between countries, etc), and was therefore useful for our user testing purposes.

The intrusiveness level of a location is determined in one of three ways. In the first, there is no pre-set intrusiveness value for the position. This is a very common situation since only a limited set of locations will have been configured by an administrator. In this case any users that happen to meet in this position will have to use local negotiation to determine the appropriate intrusiveness level (local negotiation is explained below). In the second state, the intrusiveness value for the position has been set by an administrator. This situation is straight forward and the rules of the administrator will regulate the intrusiveness levels for any users who meet in that location. In the third state, the intrusiveness value for the position is governed by a hardware knob located in this position. The knob, which is an actual physical device which may be turned left or right to decrease or increase the intrusiveness level, is put in a location to control the intrusiveness rules for that location. Users who meet in the location can turn the knob to regulate the overall intrusiveness management for that spot and the physical properties of the knob make this a tangible experience. Also, since users are in the same location they can discuss, face to face, and agree upon the appropriate turning of the knob. The co-location of users also means that the process of changing the intrusiveness level is apparent to everyone there.

Technically, the knob continuously sends its value to the position database. Any users who are co-located with the knob can query the position database for this value. If the users who are in this position, and by their location are implicitly involved in a common meeting, decide to change the intrusiveness level they can do so using the knob. All users' Sentinel services will find out about this change next time they query the position database for the intrusiveness level. This mode is also characterized by social interaction since it is left up to the meeting participants to discuss how the knob should be set.

Fig. 7. The knob

The hardware knob is a *GriffinTechnologies PowerMate*, a brushed aluminum knob which communicates with a computer through a USB interface. The knob sends events whenever its state is changed; it can be turned in either direction, and additionally it works as a button that can be pressed. In our prototype only the turning of the knob is registered. Through low-level drivers the knob communicates with the Linux operating system running on a laptop attached to it, and its events are then forwarded to FEELIM by software running on the computer.

The feedback channels to the users consist of the applications in the sView briefcases, and the iClock. The sView briefcases of the different users get their intrusion setting events through the NBLS. This means that when a user turns the knob, the knob handling software updates its entry in the NBLS. It then sends a second event to the NBLS, which in turn is forwarded to all sView briefcases, telling them to update their intrusiveness setting. Similarly, when using our Jini architecture, an event is sent through to the NotifyRouter, informing it that a change in the intrusiveness setting has occurred. The NotifyRouter then queries the NBLS, and updates the state of the iClock when a change has occurred. Feedback is then provided to the user through the applications running in the sView briefcases, as well as through the iClock. The actions whenever a user turns the knob are as follows:

1. The user turns the knob
2. Events about the change are sent through USB to the computer to which the knob is attached
3. The events are filtered by low-level drivers, and forwarded to higher-level software
4. The higher-level software updates the intrusiveness entry connected to its location in the NBLS
5. The higher-level software sends an event to the NBLS that a change has occurred

6. The higher-level software sends an event through the Jini middleware to the NotifyRouter, telling it that a change has a occurred
7. The sView briefcases get notified by the NBLS that a change has occurred
8. The sView briefcases query the NBLS to get the new intrusiveness setting for this location
9. The NotifyRouter queries the NBLS to get the new intrusiveness setting
10. Applications, including the IClock, update their interfaces to reflect the change

8 Agent Negotiation

Whenever a message is sent to a user in the meeting room, the user's agent will intercept the message and negotiate for the utility maximizing notification. The utility an agent obtains from a given notification is denoted by its preferences that constitute its utility function. In this work, agents have their preferences specified and fixed a priori. The utility function returns a value between 0 and 1 for a given message received by the agent. The agents can also obtain utility from the "points" that other agents may share with them.

Whenever a message is displayed on a given device, the agent handling that message must pay an appropriate fee to the system agent managing the meeting room. There is no cost to sending notifications to private/non-disturbing devices, but the cost increases the more intrusive the device is. However, the more intrusive the device, the more it will catch other users' attention. Thus other users have an interest in getting certain messages displayed on these devices and will pay a part of the cost, if and only if the contents/recipient/sender of the message brings them some utility. Negotiation (even when there is no need to display on a public device) can then lead to an important message (for other users) being transmitted to a public device (in case it also brings some utility to the direct recipient of that message). The decision making of an agent receiving a message is generally as follows:

- Receive message and parse subject and sender fields to see if matching with preferences of owner (user).
- If message is of no value (i.e. no sender or subject found in the preference list), do not use any arguments in ensuing negotiation.
- Otherwise, negotiate using promises and appeals (to past promises) with other agents.
- If funds received from other agents then choose the public display if sufficient funds are available.
- Else choose the instant messenger if sufficient funds are available.
- else choose the email client (or queue)
- Send confirmation to meeting-room server and update all agents' budgets and commitments accordingly.

The algorithm for the negotiation is as follows.

- Start
- Agent receives the message - parses its contents and determines utility.
- Agent sends out offers to all agents – offer (msg, DeviceX) - for every device available to it. (We assume here that messages do not need to be private but can easily retract that assumption if we simply code the agent to "silently" notify its user).
- Agents respond with their investment given the argument supplied and the message. The offer is not made strategically but in good faith.
- Proponent then pools the investments and determines the device all other agents are willing to invest in according the utility it derives from each.
- Proponent then notifies all agents of its final decision (allowing other agents to store commitments and deduct investments).
- Proponent sends appropriate investment to system agent (keeps the extra for itself) and gets the message displayed.
- End

The choice of which argument to send is dependent on whether the opponent in the negotiation has made promises in the past and whether the proponent is in a position (given its budget) to make a promise of a future reward. In trying to maximize utility, the proponent will go for an appeal if this is enough to cover its needs for funds. Otherwise, the proponent may make a promise of future points.

The above process happens sequentially with all opponents in the environment. In this way, the proponent cumulates points from past negotiations as it is negotiating with each agent. In doing so, the proponent also keeps tracks of promises it is making and appeals it is making and therefore adjusting its cost and budget for that set of negotiation.

9 Potential Impact of Results

We have demonstrated that it is possible to develop a robust prototype, i.e. our FEELIM system, which can handle intrusiveness from standard communication services in a collaborative work setting. The prototype is realistic in the sense that it handles real communication services like phone, SMS, instant messaging and email in a general fashion. The prototype together with demonstration scripts that we developed in the project is an excellent basis for discussions with telecom operators and other service providers on the development of commercial services built upon the same principles as FEELIM.

The commercial development in the area of embedded systems, mobile and handheld computing is fast and short sighted; little effort is put into usability, considering the usage of the products in a situated usage context. This results in products that might fit an isolated usage scenario, but create problems when introduced en-masse in actual working contexts. This is a potential threat to health and quality of life for such systems' users.

The explicit purpose of the FEEL technology is to increase quality of life in the sense of improving the work environment in collaborative work situations. If intrusions can be diminished in such situations, the level of stress can be lowered and the efficiency and shared focus can be promoted. Even if the FEEL project has studied collaborative work situations, the FEEL technology is applicable also in many other situations of everyday life, where a focused co-located setting has to be protected against potentially intrusive communication requests.

Acknowledgements

The authors acknowledge the funding of the FEEL project by the European Community under the "Information Society Technologies" Programme (project IST-2000-26135).

The authors wish to thank research administrators Gloria Dixon-Svärd (DSV) and Eva Gudmunsson (SICS). Important technical contributions were also made by DSV master students Pelle Carlsson, Jennie Carlstedt, and Oskar Laurin.

References

Bylund, M.: Personal Service Environments - Openness and User Control in User-Service Interaction. Licentiate Thesis, Uppsala University (2001)

Bylund, M., Espinoza, F.: sView – Personalized Service Interaction. In: Bradshaw, J., Arnold, G. (eds.) Proceedings of 5th International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 2000), pp. 215–218. The Practical Application Company Ltd., Manchester, UK (2000)

Chen, D., Vertegaal, R.: Using mental load for managing interruptions in physiologically attentive user interfaces. In: CHI '04 extended abstracts on Human factors in computing systems, Vienna, Austria, pp. 1513–1516 (2004)

Clark, H.H.: Using Language. Cambridge University Press, UK (1996)

Espinoza, F. et al.: GeoNotes: Social and Navigational Aspects of Location-Based Information Systems. In: Abowd, G.D., Brumitt, B., Shafer, S. (eds.) Ubicomp 2001: Ubiquitous Computing. LNCS, vol. 2201, pp. 2–18. Springer, Heidelberg (2001)

Fagerberg, P., Espinoza, F., Persson, P.: What is a place? Allowing users to name and define places. In: Proceedings of ACM Conference on Human Factors in Computing Systems (CHI 2003), pp. 828–829. ACM Press, New York (2003)

Ho, J., Intille, S.S.: Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In: Proceedings of the SIGCHI conference on Human factors in computing systems, Portland, Oregon, USA, pp. 909–918 (2005)

Johanson, B., Fox, A., Winograd, T.: The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. In: IEEE Pervasive Computing, pp. 67–74. IEEE Computer Society Press, Los Alamitos (2002)

McFarlane, D.: Coordinating the interruption of people in human-computer interaction. In: Human Computer Interaction - INTERACT'99, Riccarton, Edinburgh, Scotland (1999)

Nylander, S., Bylund, M.: Providing device independence to mobile services. In: Carbonell, N., Stephanidis, C. (eds.) Universal Access. Theoretical Perspectives, Practice, and Experience. LNCS, vol. 2615, pp. 465–473. Springer, Heidelberg (2003)

Nylander, S., Bylund, M.: Device Independent Services. SICS Technical Report T2002:02 (2002)

Ramchurn, S.D.: Multi-Agent Negotiation using Trust and Persuasion. PhD, Electronics and Computer Science, University of Southampton (2004)

Ramchurn, S.D., Deitch, B., Thompson, M.K., de Roure, D.C., Jennings, N.R., Luck, M.: Minimising intrusiveness in pervasive computing environments using multi-agent negotiation. In: Proceedings of 1st Int. Conf. on Mobile and Ubiquitous Systems, Boston, USA, pp. 364–372 (2004)

Ramchurn, S.D., Jennings, N.R., Sierra, C.: Persuasive negotiation for autonomous agents: A rhetorical approach. In: Kurumatani, K., Chen, S.-H., Ohuchi, A. (eds.) IJCAI-WS 2003 and MAMUS 2003. LNCS (LNAI), vol. 3012, pp. 9–17. Springer, Heidelberg (2004)

Streitz, N., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P., Steinmetz, R.: i-LAND: An interactive Landscape for Creativity and Innovation. In: ACM Conference on Human Factors in Computing Systems (CHI'99), Pittsburgh, Pennsylvania, USA, pp. 120–127. ACM Press, New York (1999)

Streitz, N., Tandler, P., Müller-Tomfelde, C., Konomi, S.: Roomware: Towards the Next Generation of Human-Computer Interaction based on an Integrated Design of Real and Virtual Worlds. In: Carroll, J. (ed.) Human-Computer Interaction in the New Millennium, pp. 553–578. Addison-Wesley, Reading (2001)