Design Principles for Inducing Reactivity in Ubiquitous Environments

Fahd Al-Bin-Ali
The University of Arizona
Computer Science Department
Tucson, AZ 85721, USA
albinali@cs.arizona.edu

Abstract

Ubiquitous environments are becoming increasingly aware of their inhabitants: they posses vision capabilities, auditory capabilities and more interestingly intelligent capabilities. At the heart of such environments is the ability of systems to initiate responsive reactions to contextual changes. In this paper, we describe the concept of ubiquitous systems' reactivity. We identify important design principles that should be embraced by systems designers to build reactive systems that can engage users in richer and more meaningful interactions. Finally, we describe a prototype that we deployed in a domestic space and we explain how our design principles can help bridge the gap between end-users and ubiquitous systems.

1. Introduction

Researchers in ubiquitous computing often envision people moving across spaces that are highly reactive. In this vision, ubiquitous environments are no longer unaware of their inhabitants: they posses vision capabilities, auditory capabilities and more interestingly intelligent capabilities. They characterize the human behavior by observing how humans act and interact together. Furthermore, ubiquitous environments collaborate with humans in meaningful ways and even assist them in making their surroundings more pleasant. Undoubtedly, at the heart of such environments is the capability of systems to initiate responsive reactions to contextual changes.

One important aspect of ubiquitous environments is the invisibility of the pervasive system. The ability of systems to fade into the background of everyday life requires striking a balance between systems reactivity and passivity: systems should not be disruptive, authoritarian or intrusive but inherently amiable and therefore invisible [28]. However, numerous challenges unfold in relation to systems' reactivity such as: devising accurate mechanisms for configuring reactivity in ubiquitous spaces, identifying the extent of users involvement in correcting reactive actions, defining appropriate models for representing knowledge about reactive actions, devising mechanisms that enable systems to evolve with their reactive behaviors etc. Confronted with the need to overcome these challenges and with the burden of dynamically controlling a myriad of devices and services, we advocate that inducing reactivity in ubiquitous environments is a non-trivial task that requires careful investigation.

In this paper, we present a design philosophy that we pursued in constructing a platform that induces reactivity in ubiquitous spaces. We note however that our research covers a broad set of disciplines that we felt could be of great value to our work including studies in human psychology aimed at understanding group and cooperative behaviors and work done by logicians in the area of reasoning. The rest of this paper is organized as follows: in section 2, we describe related work on reactivity of ubiquitous systems. In section 3, we give taxonomy of reactive behaviors in ubiquitous spaces. In section 4, we present four design principles for inducing reactivity in ubiquitous environments. In section 5, we describe the status of our current research and show how our design guidelines helped us build a prototype and deploy it in a domestic space. Finally, in section 6, we draw some final observations and conclude.

2. Related Work

There has been a good deal of work with context in pervasive environments [12, 13], with many definitions for context-awareness including both tangible aspects of context such as location awareness and intangible aspects such as activity awareness, user mood awareness, user intention awareness etc. Not surprisingly, intangible aspects of context have received less attention to date, in particular, due to the lack of proper methods for modeling salient factors such as intentions, moods and expectations. We note however that ubiquitous systems that react appropriately to users need to have a deep understanding of both tangible and in-

0-7803-8577-2/04/$20.00 ©2004 IEEE

tangible aspects of context.

Early research [11, 14] in systems' contextual reactivity focused on establishing simple relationships between tangible context and appropriate actions, for example, switching on and off devices based on user proximity. The Aware Home project employed this simple approach for augmenting the DUMMBO board to react to people (e.g. when people are gathered around the board it starts recording audio). These forms of reactive actions depend on simple mappings that have been programmed into the system.

Latterly, EasyLiving [7] from Microsoft aimed at developing an extensible reactive system that learns from users interactions and evolves with time. The project identified "automatic system behaviors" as one of the challenges that faces ubiquitous systems. In an effort to induce reactivity, the project developed a behavior engine that polls a world model database and issues commands to software agents (representing devices and applications). The engine examines a set of context variables in order to decide what actions to take. EasyLiving implemented two automatic behaviors based on users activities including: lighting changes occur automatically within the ubiquitous space and UI routing occurs as users move across the environment. Unfortunately, EasyLiving was terminated and reactive actions were limited to the above applications.

Similarly, MIT's IRoom [6, 9, 19] identified that "inherent in the design of the intelligent room is the ability for the room to react to user behavior". Towards that end, the project developed its first generation reactive system as an expert system. In 2002, Kulkarni developed the second generation reactive system that uses java objects and macros to encode reactive behaviors. Basic features of the new system are: modular reactive components, layered architecture, customizability and extensibility. The IRoom provides several mechanisms for inducing reactivity including: hard coding the behavior logic in the platform itself, using GUI tools that aid users in dynamically inducing reactivity and identifying appropriate reactions based on observing users interaction in a ubiquitous space.

Other projects have also worked on various aspects of reactivity in ubiquitous environments including: Gaia [25], the neural house [21], the intelligent classroom [31], responsive offices [14] and musicFX [20]. However, due to space limitations, we cannot give a detailed account of the contributions of these projects. Next, we provide a classification of reactive behaviors in ubiquitous environments.

3. Taxonomy of Reactivity

As explained earlier, reactivity in ubiquitous environments is the capability of systems to initiate responsive actions to both tangible and intangible contextual changes. Previous research [7, 23] identifies (at least) 6 classes of reactive actions including:

1. Programmed: End-users explicitly program behaviors into systems. This method requires deploying appropriate mechanisms for Human-Computer Interaction (HCI) that are intuitive, simple and user-friendly.
2. Demonstrated: Ubiquitous systems capture appropriate behaviors by going through a training session.
3. Observed: Ubiquitous systems continuously observe users behaviors in order to identify appropriate reactive actions.
4. Fixed: Systems manufacturers could provide a default set of reactive behaviors that come with every ubiquitous system.
5. Installed: This is provided as an add-on component where software developers deliver installation updates that augment systems' reactive behaviors.
6. Shared: Spatially or geographically dispersed environments exchange knowledge about users habits including appropriate reactive behaviors.

It is important to emphasize that the above categories are neither exhaustive nor mutually exclusive, but we have deliberately chosen them because of their importance in the context of ubiquitous environments. In fact, reactive systems are likely to employ two or more of the above mechanisms. It should also be clear that many of the mechanisms discussed above are greatly exacerbated when multiple people are interacting with a ubiquitous system. The complexities associated with these situations will not be discussed further in this paper and are future work items for us.

4. Design Principles

In this section, we describe the design philosophy we pursued in building our reactive ubiquitous system. In formulating our design guidelines, we utilize experiences from previous research [19, 27], work done by psychologists and logicians in modeling human and group behavior [15, 22] and work on self-configuring autonomous systems [29, 30]. We list the following design guidelines for building a reactive system in ubiquitous environments:

4.1. Reactive systems should be activity-centric

Donald Norman in his book Learning and Memory points out how system designers misuse and ignore users: "they start with the machine, and the human is not thought of until the end, when its too late: witness the control panels in the nuclear power plants". We believe that modeling and

understanding human and group behavior is an essential ingredient for the development of reactive systems that can engage users in richer and more meaningful interactions. Indeed, earlier studies demonstrated that humans exhibit behaviors with computers similar to their behaviors with other humans [24]. Inspired by earlier work and a number of theoretical reflections [15, 22] and practical applications in information systems [4, 18], we incorporated concepts from activity theory in our design. It is important to emphasize though that our aim is not to show that activity theory concepts provide the right way for modeling reactive behaviors, but to show that its concepts provide a clean theoretical abstraction that suits users with sufficient detail and dynamism to adapt to their preferences.

Activity theory advocates that human behavior is always part of a particular activity that is taking place. Humans tune their behaviors and preferences on the basis of activities that they participate in. People tend to reconsider their actions and adapt them in order to avoid disrupting any ongoing activities, for instance, if there is a presentation in a room, then listening to loud music might be inappropriate, therefore, people would suppress their behaviors to avoid disrupting the activity. In our design, we think of a ubiquitous environment as a space that hosts a set of concurrent activities that occasionally conflict and therefore users require means for resolving these conflicts including: internal conflicts (i.e. conflicts between reactive actions for the same activity) and external conflicts (i.e. conflicts between reactive actions for separate activities that co-exist in the same ubiquitous space).

According to activity theory, three natural mechanisms exist for resolving conflicts in human behavior including: coordination, cooperation and co-construction. These mechanisms can be directly mapped onto a ubiquitous environment to provide basic models for conflict resolution in the following ways:

- Coordination: This means that the actors (i.e. inhabitants of a ubiquitous environment) are not involved in conflict resolution: conflict resolution is done automatically by the reactive system. Possible ways of providing such a capability is through an expert system that resolves conflicts using premeditated rules, a neural network that observes conflicting situations and learns how to deal with them or prioritizing activities or actors and therefore identifying what actions have precedence. We elaborate on conflict resolution later in the paper.

- Cooperation: This means that the system was unable to resolve the conflict on its own and therefore requires the assistance of actors (i.e. inhabitants of the ubiquitous space). This capability can be supported by developing interfaces and applications that allow users to cooperate and to reach decisions on how to resolve conflicts, or simply by allowing users to explicitly suppress some of the systems' reactive actions to restore it to a non-conflicting stable state. Cooperative solutions can be learned by the system to resolve future conflicts transparently.

- Co-construction: This means that the system is consistently reacting incorrectly or inaccurately thus requiring restructuring of the coordination-level mechanism. This provides dynamism in the reactivity model and provides a way for the system to evolve. For example, a reactive system could have an over-trained neural network that is controlling the lights and is consistently functioning incorrectly, or a set of rules in an expert system that are triggering incorrect behaviors. In such cases, users or system administrators must be given means to isolate and deactivate malfunctioning components, to cancel their actions and to be able to take correcting measures, like replacing the rule set or resetting the neural network to train itself.

Activity theory is a good way of thinking about what humans expect in reactive ubiquitous environments. It provides them with a suitable model for representing events of interest, namely activities. Users do not want to think about fine grained contextual information such as GPS coordinates, temperature or light intensity, instead they would prefer thinking in terms of activities such as breakfast, presentation, meeting etc. Moreover, it appears that identifying appropriate reactive behaviors based on fine grained contextual information is likely to be a harder problem than identifying behaviors based on coarser contexts such as activities. We think that activity theory gives us some important guidelines on what is needed when designing reactive environments and therefore we used an activity-centric approach in building our reactive ubiquitous system.

4.2. Reactive systems should synergize multiple intelligent components

Towards improved reactivity, intelligent environments will utilize a suite of intelligent components that may include expert systems, neural networks, regression models, genetic algorithms, logic programming modules etc. This observation stems from the nature of the problems that face reactive environments and the inability of a single intelligent component to accurately model all problems. For example, consider classification problems that have linearly separable datasets. Such problems would require discriminant functions having decision boundaries that are linear or more generally hyperplanar in higher dimensions thus suggesting the use of simple techniques such as linear regression or single layer neural networks. However, such

techniques will fail to model many other practical applications that exhibit more complex relationships (e.g. problems with non-linear decision boundaries). Instead, multilayer neural solutions allow for more general mappings that can in fact approximate any continuous functional relationship thus making them more suitable for a general class of applications. It is important to emphasize though that the behavior of a neural network is always unpredictable in itself: neural networks are black boxes and there is an implicit assumption that users completely trust their behavior. This could jeopardize the behavior of the system as a whole, especially when incorrect behaviors get triggered.

Alternatively, rule-based systems allow users to specify behaviors in an expressive way based on a causal condition-action model which is more understandable to humans. Nevertheless, rule-based systems suffer from their own drawbacks including: they are too rigid and they lack sufficient dynamism needed in highly dynamic ubiquitous environments, accuracy of the decisions made by rule-based systems hinges on the granularity of the knowledge stored in their rules which cannot account for all possible situations and finally, having complex rule sets typically hampers the overall responsiveness of systems and increases the possibility of having cycles and conflicts.

Clearly, each approach has its pros and cons and choosing one approach or the other will depend on the problem in hand and the extent of users involvement in the decision process. Reactive systems however should be able to accommodate multiple intelligent components that work harmoniously together to get an enhanced combined effect. We therefore integrated into our system several intelligent components that can model different decision problems. We give additional details in section 5.

4.3. Reactive systems should provide means for resolving conflicts

Previous work in reactive ubiquitous systems has often ignored conflict resolution. A conflict is a state of disharmony between incompatible reactive behaviors in a ubiquitous space. We divide conflicts into two classes:

4.3.1 Explicit Conflicts.

These are conflicts that can be identified by directly examining the behaviors that get applied in a particular environment within a particular time frame. For example, one user might set in his/her preferences that after lunch, she/he likes to listen to a particular genre of music, while another prefers not to listen to any music. Trying to apply both preferences (in the same physical space) results in an explicit conflict that requires resolving.

4.3.2 Implicit Conflicts.

These are conflicts that cannot be detected using formal methods, they are not accounted for within the system, and therefore they are not considered as conflicts. Many of these conflicts stem from hidden factors such as human intentions and mood conditions. The lack of appropriate mechanisms to infer moods and intentions makes it hard (if not impossible) to identify such conflicts.

Even though explicit conflicts are easier to resolve than implicit ones, systems should have a capacity to resolve both. We highlight several strategies that need to co-exist in activity-centric reactive systems to provide robust means for handling conflicts:

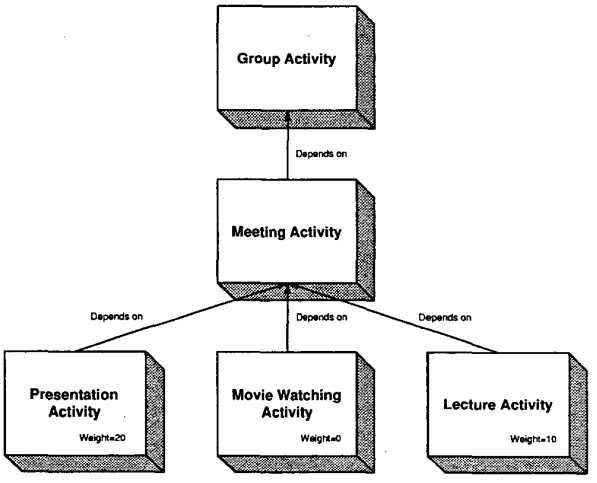

Figure 1. Example Layered Activities.

1. Layering Activities: Activities in a reactive environment could be organized as a Directed Acyclic Graph (DAG). Figure 1 shows an example of such graph, lower activities are more specialized while higher ones are less specialized. Behaviors of less specialized activities get overridden by behaviors of more specialized activities. For example, if the lighting behavior in a meeting activity is to switch the lights on, it gets overridden by the lighting behavior of a presentation activity that requires all lights to be off. In addition, non-conflicting behaviors of less specialized activities are inherited by more specialized children. For example, if a meeting activity requires the video recording system to record the meeting, this gets automatically inherited by the presentation activity.

   It is important to note that the above organization makes use of elementary concepts in set theory,

namely, containment. This also facilitates recognizing activities (e.g. by requiring that the chain of activities that parent a particular activity to hold).

2. Weighted Activities: On an event of having multiple activities with conflicting behaviors that cannot be organized as a DAG or that belong to different branches of a DAG, weights (as shown in figure 1) can be utilized to identify which activity is of greater importance and therefore arbitrate what reactive behavior should be applied if a conflict indeed exists. Weighting activities can be delegated to system administrators, negotiated among inhabitants of a ubiquitous space or dynamically assigned based on a room-activity schedule.

3. Humans in the Loop: Systems should always provide mechanisms that allow users to manually override, to undo or to deactivate reactive behaviors in any circumstance. In addition, users must be given means to negotiate a conflict resolution mechanism (as explained in cooperative interaction in activity theory). For example, a tool could provide a basic election booth that gets sent to the inhabitants of the space to elect a possible solution from amongst a range of possible solutions or to elect a person to resolve the conflict. Furthermore, new behaviors can be used (when possible) as a means to evolve with systems: new behaviors can be transparently applied when conflicts reoccur.

In conclusion, reactive systems should be able to resolve conflicts automatically as well as through users intervention. Relying on systems completely in conflict discovery and resolution might not be possible in all cases: systems might fail to discover some conflicts or might fail to react appropriately due to hidden factors. In such cases, inhabitants of a ubiquitous space should be able to take correcting measures.

4.4. Reactive systems should support self-configuration

In earlier projects, systems provided no means for automatic self-configuration. Later, some projects provided simple mechanisms for configuration like the application bridge from Gaia [25] where users utilize a GUI interface to configure different system behaviors by dragging and dropping application bridges on different system components. Copperstock et al. [10] criticize such techniques saying "such interfaces tend to exaggerate rather than solve the problems of current technology": users often require configurations that are not provided by the GUI. Consequently, we believe that inherent in the design of reactive systems is an ability to configure themselves when necessary. We list some possible techniques for supporting self-configuration in reactive systems:

4.4.1 Expert Systems and Neural Networks

Self configuration can be achieved using expert systems or neural networks. For example, it is possible to provide a set of static rules within an expert system that re-wire the applications or the devices in a ubiquitous space based on the activities taking place. Another approach could use neural networks, for example, Hopefield networks are often used in solving optimization problems, configuring ubiquitous environments can be reduced to an optimization problem that utilizes a Hopefield network to identify the best possible configuration. We note however that such techniques inherit the limitations (described earlier) associated with rule-based and neural techniques. Further details about applying expert and neural techniques for self-configuration can be found in [19, 21].

4.4.2 Temporal Propositional Logic

A new promising approach to self-configuration in autonomous systems uses temporal propositional logic [29, 30]. We decided to investigate the use of this approach in the context of ubiquitous environments. We note however that there are non-trivial differences between the domain of autonomous systems and ubiquitous reactive systems including: components in autonomous systems do not change, while in ubiquitous spaces devices and applications come and leave at will and autonomous systems are always aware of all system components and their compatibility (i.e. the possibility of configuring them together), while in ubiquitous environments it is unclear how components or applications will advertise their compatibility for potential configuration. To better understand the implications of these issues, we closely examine temporal logic and explain how it can configure components in a ubiquitous space.

Temporal logic can model a ubiquitous environment as a transition system with: state variables, a feasible subset of a state space and a finite set of transitions between space states. Figure 2 shows a feasible 'teleconferencing' space state with a number of state variables.

activity=teleconferencing $\Rightarrow$ ((plasma screen 1=on) $\vee$ (plasma screen 2=on)) $\wedge$ ((volume $\geqslant$ 3 decibels) $\vee$ (volume $<$ 2 decibels))

Figure 2. Example Space State.

Sequencing in reasoning can be supported in temporal propositional logic using the $\Rightarrow \bigcirc$ operator. For example, in an activity-centric system, activities can be identified not simply on the basis of contextual information at a specific point in time but based on how contextual information changes across time. Figure 3 describes an allowable state transition that illustrates the use of the $\Rightarrow \bigcirc$ operator.

Reactive systems should continuously monitor the physical space and modify transitions between space states reflecting configuration changes in the physical space (e.g. transitions can be added when new sensors or devices are introduced in the space or transitions can become obsolete when sensors or devices are removed from the space). Furthermore, when systems detect that a space state is not satisfied, they should generate configuration trajectories of all possible configuration chains and select an appropriate trajectory to restore the system to a feasible state. Figure 4 shows a subset of possible trajectories for a ubiquitous space where there is a need to increase the volume of the speakers.

(activity=presentation) $\vee$ (activity=meeting) $\vee$ (activity=nothing) $\Rightarrow$ $\bigcirc$(activity=teleconferencing)

Figure 3. Example State Transition.

Possible policies for trajectory selection include: choosing a path with unused devices, choosing the first path, choosing a path with minimal cost etc. In the above example, selecting a path could result in switching on an amplifier and therefore improving sound and restoring the activity transition system back to a feasible state.

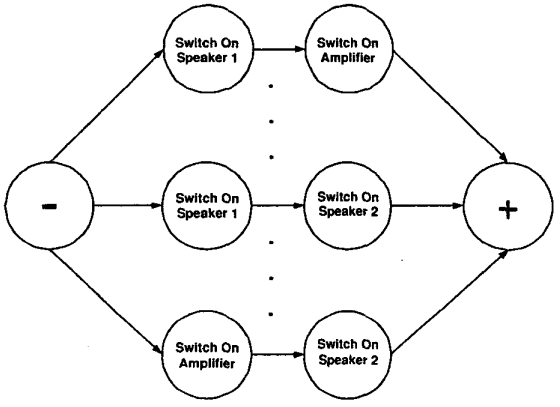

Figure 4. Example Configuration.

As noted earlier, self-configuration dictates a need for identifying compatibility between devices and applications and possibly their impact on state variables in order to be able to rewire components appropriately. Many devices and applications in a ubiquitous environment might choose not to advertise their compatibility. However, we explain three possible ways for specifying compatibility between various system components to enhance a particular objective (e.g. enhancing the volume of speakers in a ubiquitous space) provided that system components choose to advertise their compatibility:

1. Content-based Compatibility: Devices or applications advertise themselves as consumers or producers of content. For example, an amplifier advertises that it consumes sound and produces sound. Having an infeasible teleconferencing space state due to a sound state variable would make systems identify the amplifier as a potential candidate for reconfiguration. One drawback in this approach is that it assumes agreement on content types across applications and devices. Finally, we note that several technologies provide frameworks that support content-based compatibility such as UPnP [1] and Jini [2].

2. User-identified Compatibility: Systems provide tools (similar to Gaia's application-bridge) that allow inhabitants of ubiquitous environments to identify groups of compatible devices or applications for potential reconfiguration with respect to specific state variables. For example, users can specify that an amplifier and a microphone and some speakers (connected in a specific way) can increase the volume in the space.

3. Negotiation-based Compatibility: In this approach, a negotiation phase precedes configuration to identify possible candidates for reaching a particular objective. Devices and applications negotiate among themselves the possibility of enhancing a particular state variable to restore the system to a feasible state. This again might require users' assistance.

So far, we have listed four design guidelines for building reactive ubiquitous systems. Next, we describe our system prototype and we explain how the above guidelines helped us develop a better understanding of reactivity in ubiquitous environments.

5. Research Status

One important goal of our work was to deploy and evaluate a system in a real-world setting where users participate in their daily activities while the system collects information, analyzes various behavioral patterns and reacts appropriately. This approach helps us demonstrate the feasibility of our work and provides better understanding of the design principles that we described earlier. We therefore used the above principles to build a reactive ubiquitous system. Key aspects of our implementation are:

1. System reactions are based on users' activities.

2. The system utilizes two intelligent components, namely, feed forward neural networks and logistic regression components for classifying users' activities.

3. The system resolves conflicts by invoking end-users to disambiguate situations of high uncertainty.

We note however that self configuration is not yet supported by our system.

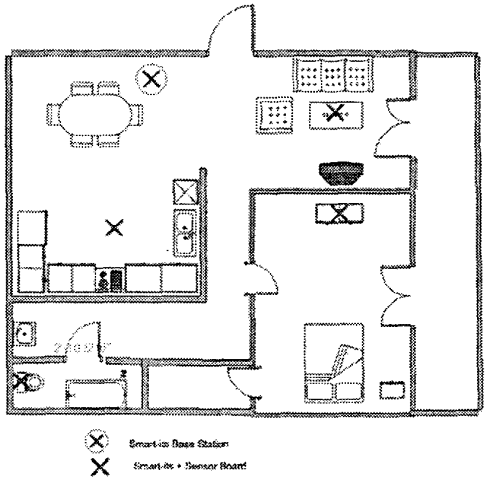

Figure 5. Smart Space Floor plan

Our architecture consists of two main components: a sensing subsystem and an intelligent subsystem. Figure 5 shows a sketch of the floor plan of the space with various sensor boards. We give a brief overview of these components.

5.1. The Sensing Subsystem

The sensing subsystem has 5 smart-its boards. A smart-its board [16] is built around a microchip PIC microprocessor (PIC18F252) with 14 inputs for binary sensors and analog-to-digital conversion units that allow five analog sensors to be attached. Attached to each board are the following sensors:

- Two dual-axis accelerometers (ADXL311) that measure dynamic acceleration (e.g. vibration) and static acceleration (e.g. gravity).
- A capacitive proximity sensor.
- A temperature sensor.
- A light intensity sensor.
- An infrared sensor.

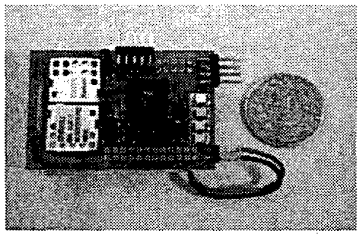

Figure 6. Smart-Its Board

Each board is also equipped with a radiometrix BIM3 chip that transmits and receives data wirelessly over a short range. Figure 6 shows a smart-its board.

In addition, the sensing subsystem also has three software components:

1. Smart-its Client Software: samples data from the sensors approximately every 30 seconds and transmits the information wirelessly to a base station.
2. Smart-its Base Station Software: receives data from client smart-its and copies the information onto a serial port.
3. Data Collection Software: reads data from the serial port, formats it and stores it appropriately for further processing by the intelligent system.

5.2. The Intelligent Subsystem

The intelligent subsystem consists of two modeling tools: Feed-forward Neural Networks (FNNs) with logistic activations and a binomial and a multinomial Logistic Regression (LR) component.

FNNs are the most popular and most widely used neural models in many practical applications. They provide a general framework for representing non-linear functional mappings. They are composed of successive layers of adaptive neurons. FNNs are trained by changing the neuron weights until a desired input-output relationship is realized. They are well suited to classification problems and therefore we decided to employ them in our system. Further details of FNNs can be found in [5].

Similarly, the use of LR modeling has become extremely popular during the past decade. LR classifies inputs based on responses with the largest expected probability for a factor/covariate pattern. It has less stringent requirements than conventional regression techniques including:

1. It does not assume linearity of relationship between the independent variables and the dependent.

2. It does not require normally distributed variables.

3. It does not assume homoscedasticity (i.e. the variance around the regression fit is the same).

Furthermore, several studies [8, 26] have shown that LR is a powerful analytical tool for classification problems thus making it a suitable choice for our task. Details of LR can be found in [17].

Finally, we note that training the intelligent subsystem requires users to specify to the system their activities (e.g. cooking, exercising, dancing) as they participate in them using wireless handheld devices.

5.3. Summary of Results

While the aim of this paper is not to provide detailed experimental results but to discuss the guidelines that were employed in our design, we give a brief description of our experiments. Our experiments analyzed sensor data recorded over 4 weeks in the domestic space shown earlier. We performed experiments to determine the classification performance of FNNs and LR for classifying activities. Our results showed that for small numbers of activities and sensors, both feed forward neural networks and logistic regression are effective in identifying activities with minimal degradation in performance as the number of activities increases. For further details, we refer the reader to a more detailed discussion in [3].

6. Conclusion and Future Work

In this paper, we presented four important guidelines for designing reactive ubiquitous systems that can engage users in richer and more meaningful interactions. First, reactive systems should have a deep understanding of users' context and in particular users' activities to react appropriately and invisibly. Second, reactive systems should synergize multiple intelligent components to improve the overall human-computer experience and to be able to model a wide variety of problems. Third, mechanisms for resolving conflicts between reactive actions should be deployed in ubiquitous spaces. This includes mechanisms for resolving explicit conflicts (that can be identified by systems) and implicit conflicts (that require users intervention). Finally, reactive systems should support self configuration to be able to rewire different system components for satisfying users' objectives.

It is important to note that ubiquitous systems face many other challenging problems in relation to reactivity such as devising mechanisms that prevent systems from violating the privacy of individuals, identifying ways for specifying device ownership and how bound is an individual to a particular device [23], devising mechanisms for exchanging information about users behaviors and habits across different spaces with possibly different devices and applications etc. We note that these issues are not of less importance than the technical issues we discussed earlier and will require careful investigation.

While our prototype currently recognizes several activities and utilizes several intelligent components, our system still does not fully support conflict resolution or self-configuration. Our plan for future work is to augment our system with components that provide these functionalities. Moreover, we are currently developing a more dynamic logistic model that captures the temporal and the spatial structure in our data. We also intend to deploy our system in different ubiquitous environments including private and public spaces and to make our system accessible to a user community that can report on the impact of our system on user perceptions of activity analysis, conflict resolution and self-configuration. Finally, we hope that our future work will provide a more informed insight on how our design guidelines impact the relationship between end-users and ubiquitous systems.

References

[1] `http://www.upnp.org`, 2004.

[2] `http://www.sun.com/jini`, 2004.

[3] F. Al-Bin-Ali, P. Boddupalli, N. Davies, and A. Friday. Correlating sensors and activities in an intelligent environment: A logistic regression approach. In *Proc. the First European Symposium on Ambient Intelligence*, Eindhoven, The Netherlands, 2003.

[4] P. Barthelmess and K. Anderson. View of software development environments based on activity theory. *Computer Supported Cooperative Work*, 2002.

[5] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 2003.

[6] R. Brooks., M. Coen, D. Dang, J. DeBonet, J. Kramer, T. Lozano-Perez, J. Mellor, P. Pook, C. Stauffer, L. Stein, M. Torrance, and M. Wessler. The intelligent room project. In *Proc. of the Second International Cognitive Technology Conference (CT'97)*, Aizu, Japan, 1997.

[7] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. Easyliving: Technologies for intelligent environments. In *Proc. Handheld and Ubiquitous Computing*, Bristol, UK, 2000.

[8] J. Brzezinski and J. Knafl. Logistic regression modeling for context-based classification. In *Proc. of DEXA Workshop*, Florence, Italy, 1999.

[9] M. Coen. Design principles for intelligent environments. In *Proc. of AAAI*, Madison, Wisconsin, 1998.

[10] J. Cooperstock, S. Fels, W. Buxton, and K. Smith. Reactive environments: Throwing away your keyboard and mouse. *Communications of the ACM*, 1997.

[11] K. Cory, R. Orr, G. Abowd, C. Atkeson, I. Essa, B. MacIntyre, E. Mynatt, T. Starner, and W. Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Proc. the Second International Workshop on Cooperative Buildings (CoBuild'99)*, Pittsburgh, USA, 1999.

[12] A. Dey, J. Mankoff, G. Abowd, and S. Carter. Distributed mediation of ambiguous context in aware environments. In *Proc. 15th Annual Symposium on User Interface Software and Technology*, Paris, France, 2002.

[13] A. Dey, D. Salber, and G. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. In *HCI Journal 16 (2-4)*.

[14] S. Elrod, G. Hall, R. Constanza, M. Dixon, and J. D. Riveres. Responsive office environments. *Communications of the ACM*, 1993.

[15] Y. Engestrom. *Learning by expanding: An activity-theoretical approach to developmental research*. Orienta-Konsultit, 1987.

[16] L. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artifacts. In *Proc. Ubicomp 2001*, Atlanta, GA, 2001.

[17] D. Hosmer and S. Lemeshow. *Applied Logistic Regression*. John Wiley Sons, 2000.

[18] M. Korpela, A. Mursu, and H. Soriyan. Information systems development as an activity. *CSW Journal: Special issue on Activity Theory and Design*, 2001.

[19] A. Kulkarni. A reactive behavioral system for the intelligent room. Master's thesis, The Massachusetts Institute of Technology, Cambridge, MA, USA, May 2002.

[20] J. McCarthy and T. Anagnost. Musicfx: An arbiter of group preferences for computer supported collaborative workouts. In *Proc. of the 1998 ACM Conference on Computer Supported Cooperative Work*, 1998.

[21] M. Mozer. The neural network house: An environment that adapts to its inhabitants. In *Proc. the American Association for Artificial Intelligence, Spring Symposium on Intelligent Environments*, Menlo, CA, USA, 1999.

[22] B. Nardi. Context and consciousness: Activity theory and human-computer interaction. *MIT Press*, 1996.

[23] P. Phillips, A. Friday, and K. Cheverst. Understanding existing smart environments: A brief classification. In *Proc. of Workshop on Ubiquitous Computing in Domestic Environments*, 2001.

[24] B. Reeves and C. Nass. The media equation: How people treat computers, television, and new media like real people and places. *Cambridge University Press/CSLI*, 1996.

[25] M. Roman, B. Ziebart, and R. Campbell. Dynamic application composition: Customizing the behavior of an active space. In *Proc. of PerCom 2003*, Dallas-Fort Worth, Texas, 2003.

[26] M. Schumacher, R. Robner, and W. Vach. Neural networks and logistic regression: Part i. *Computational Statistics and Data Analysis*, 1996.

[27] S. Shafer, B. Brumitt, and J. Cadiz. Interaction issues in context-aware intelligent environments. *Human-Computer Interaction*, 2001.

[28] M. Weiser. The computer for the twenty-first century. *Scientific American*, 1991.

[29] B. Williams and P. Nayak. A model-based approach to reactive self-configuring systems. In *Proc. the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, Portland, Oregon, 1996.

[30] B. Williams and P. Nayak. A reactive planner for a model-based executive. In *Proc. of the Fifteenth International Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[31] L. Winer and J. Cooperstock. The 'intelligent classroom': Changing teaching and learning with an evolving technological environment. *Journal of Computers and Education*, 2002.