In *Proceedings of the 2008 NSF Next Generation Software Program Workshop, April 14, 2008.*

Steps toward Activity-Oriented Computing

João Pedro Sousa[†], Vahe Poladian[*], David Garlan[*], Bradley Schmerl[*], Peter Steenkiste[*]

[*] Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213
{garlan,schmerl,steenkiste,poladian}@cs.cmu.edu

[†] George Mason University
Fairfax, VA 22030
jpsousa@gmu.edu

***Abstract*—Most pervasive computing technologies focus on helping users with computer-oriented tasks. In this NSF-funded project, we instead focus on using computers to support user-centered "activities" that normally do not involve the use of computers. Examples may include everyday tasks around such as answering the doorbell or doing laundry. A focus on activity-based computing brings to the foreground a number of unique challenges. These include activity definition and representation, system design, interfaces for managing activities, and ensuring robust operation. Our project focuses on the first two challenges.**

***Index Terms*—Pervasive computing, activities, home security, home automation.**

I. Introduction

Over the past few years there has been considerable progress in development of pervasive computing technologies. For example, research groups have developed toolkits for capturing and using context information [1,2,3,4,5] mobile platforms [6], and a variety of context-aware applications that target specific users and application domains, such as management of computer tasks [7,8], tour guides [9,10], lab experiments [11], and smart homes [12,13,14]. Most of these efforts share a common theme: they are computer-oriented in the sense that users use computers to solve their problems. Specifically, they center on computing platforms (handhelds, laptops, wearables, cell phones, etc.) that help users with specific tasks, or they help people use computers and manage computer tasks more effectively in diverse environments.

While a computer-centric focus is not surprising (after all, the research is done by computer scientists), it tends to limit the scope of pervasive computing technology to situations in which the computing platform is the primary object of user attention - be it accessing information, communicating with email or video conferencing, preparing a document, etc. This in turn limits the ability to create systems that *address the activities that most people do most of the time with little regard to the underlying computer support*.

In this project we take a radically different view of pervasive computing. Rather than *focusing on the computer itself*, we instead focus on support for user-centered "activities" where computing capabilities are used to enhance a user's ability to carry out every-day, non-computer tasks. While we use computing technology to support such activities, the computing infrastructure is largely transparent to the user, complementing what a user is already doing, and mapping users' broader goals to lower-level computing services that support those goals. For concreteness, we plan on investigating this new approach in the context of the home environment, where most activities are not intrinsically computer-oriented. Examples of activities range from the (seemingly) trivial (e.g., responding to a doorbell), to more complex and critical (e.g., maintaining the security or physical integrity of a house, or helping to monitor and manage the health of a resident sick elder.)

A focus on activity-based computing brings to the foreground a number of unique challenges. These include activity definition and representation, system design, interfaces for managing activities, and ensuring robust operation. This project focuses on the first two challenges. In this paper we give an update on two fronts. First, building on our experience in task definition and representation in the office environment, we describe an initial proposal for defining and representing activities. A key consideration is that, in contrast to computer-based tasks, users are active participants in the activity, so when marshalling services and resources to support an activity, the user should be viewed as a resource. A second consideration is that, in their daily lives, users move between different domains and activities must seamlessly move with them. This raises significant security challenges that must be addressed at the system design level.

The research described in this paper was mostly done by students at Carnegie Mellon University Part of the work described was done by a PhD student, Vahe Poladian, as part of his thesis work. The work described in Section III was in part executed by a team of MS students as part of their studio project the Masters of Software Engineering program. Finally, several undergraduate students participated in the research.

II. Activity Definition

We have been investigating an approach that allows end-users to assemble and evolve personalized activities. This

work has been reported in more detail in [15]. The approach must:

a) be simple enough for end-users to manipulate with little initial training;

b) have and effective cost/benefit ratio; and,

c) be precise enough to be used to automatically assemble a running system.

We are investigating whether an approach combining the component and connector architecture view with activity oriented computing is a suitable foundation to satisfy these requirements. We believe that an approach based on code structures and programming languages is too removed from the experience of end-users for achieving these goals.

We use a metaphor of boxes, pipes, and wires, which is similar to consumer electronics. In this domain, end-users may buy a number of devices and cables and try different assembly configurations. They need only to have a basic knowledge of what travels on each cable, without having to understand the corresponding electrical specifications.

We call our approach uDesign. uDesign represents run-time structure only, in contrast to code structures, and is at a higher level of abstraction than code structures. Furthermore, as is frequent practice in design disciplines, uDesign separates views of structure and behavior. (e.g. [16]).

The boxes in uDesign correspond to running entities (or services) that can incorporated in a system, rather than to classes or instance factories. Choosing the latter option would mean that end-users would have to create programs or scripts to control the creation, assembly, and destruction of instances in the system. Instead, uDesign uses a discovery mechanism to find available service instances, and offers interactive primitives for end-users to integrate and interconnect those services into a system.

We have been inspired by a recent trend in service-oriented computing to separate the roles of service supplier and service consumer. This helps to manage the detail that is required to manipulate the activities, and the usablility for a broad user base. In uDesign we take this trend one step further by advocating two groups of service consumers: domain specialists, such as doctors; and end users with a general education.

Services are required to work out of the box, with default behavior, or possibly with a set of typical behavior templates. A general user should be able to make use of such services using the default behaviors or possibly recognizing abstract parameters or modes of operation, such as normal operation and emergency operation. Domain specialists or technical users can tailor those generic templates; for example, a doctor defining that the emergency mode corresponds to the heart rate exceeding 140 beats per minute (bpm) for a given patient, but only 120bpm for another patient.

A. An Overview of uDesign

This section introduces the concepts in uDesign, illustrating the understanding that end-users need to create and tailor activities such as the ones presented in Sections II.B and II.C.

As mentioned above, the three main constructs in uDesign are boxes, pipes, and wires. Boxes are the locus of computation, while pipes stream data among boxes. Wires control starting and stopping activities on boxes, as well as the flow of data on pipes, based on observed conditions.

To help manage visual clutter and to separate concerns, uDesign uses three overlays:

1) *Structural:* identifies the boxes, their properties, and internal structure, and the piping of data among boxes;
2) *Box Behavior:* identify the start and stop conditions of the activities in boxes; and
3) *Pipe Behavior:* specifies the conditions that enable or disable the flow of data on pipes.

These views may be merged together to form one complete view of the activity.

Boxes correspond to entities of interest or their activities, and may be hierarchically decomposed to allow scaling or information hiding. Boxes may be associated with the TV set in a user's living room, with the living room as a whole, or with the user's activity of watching a TV show. Boxes may also be associated with software components, which like devices are viewed in the perspective of a concrete operating component that contributes to the system's function.

When a box is associated with a physical space or an object, such as a couch, the box is realized as a combination of software and hardware that monitors and maybe controls the corresponding physical entity. We envisage that software and hardware to integrate with the environment will be sold along with the commodity. For example, building companies will construct smart homes; furniture stores will sell smart couches (or the means to make old couches smart). End-users, via uDesign, will assemble and configure the smart objects to suit their needs.

In addition to smart objects and spaces, users and their activities may have associated boxes. Such boxes identify the properties of interest and clarify the user's role in achieving the system's intended function. In this way, users can be represented in an activity definition as another resource that can be used in the system to achieve the activity's goals. We anticipate that smart spaces will be equipped with generic software components for modeling activities, and which may be associated with humans and their activities.

Boxes have inputs, which are entry points for data, and properties. Properties are any observable aspect of a box, such as the video output of a DVD player, whether it is powered up, or its location.

Data may be piped between any property of a box, a producer of data, and an input in a box, a consumer of data. Whenever a piece of data is available on the producer side, the pipe will transmit it towards the consumer side. uDesign tools check for type compatibility and disallow invalid piping, such as trying to pipe a video output to a textual input.

The box behavior and pipe behavior overlays identify the conditions that give rise to starting and stopping activities in boxes, and that enable or disable the flow of data on pipes, respectively.

Conditions are expressions over the inputs and properties of

the box they are associated with, or over the properties of the boxes contained in the latter. In addition to operators such as equals (=), and (&), and or (|), conditions may include temporal operators such as count(c, t) that counts how many times condition c became true in the latest time interval t; or sust(c, t) which is true if condition c sustained a true value during the latest time interval t.

Wires transmit the result of evaluating a condition and may trigger one of three operations on boxes: start, pause, and stop, denoted by ▶, ⏸, and ■, respectively. Start operations may indicate the values of one or more inputs, which then should not be connected to pipes. The pause operation preserves the values of the properties and inputs to the box until a start is triggered again, possibly overriding some of those input values. A stop operation resets all the values in a box, being used, for instance, for privacy purposes.

Valves can be placed on pipes, preventing the flow of data unless the enabling conditions are met. For example, the video output of a medical camera will not be released unless a potential emergency is declared.

B. *Example 1: Susan's heart condition*

In this section we present an example where a uDesign activity helps to manage the health of an elderly lady, Susan, who has developed a heart condition. Susan's doctor wants her condition to be constantly monitored. Being a domain specialist, the doctor ceates a box in uDesign for monitoring Susan's health, which wraps three services (see Figure 1(a), left hand side): heart rate monitoring, stream logging (for offline reference), and video capture (meant for checking on Susan remotely should a problem arise). More sophisticated biometric devices could be included into the service at a later date, but for now the doctor decides that monitoring Susan's heart is sufficient.

The doctor uses pipes to connect the monitored heart rate to the log input, and also to make the video output visible at the top level, so that it can be used by other services. After discussing Susan's lifestyle and physiological characteristics, the doctor identified two conditions to be monitored: when Susan's heart rate sustains a level above 90bpm for 20 minutes, and when it either exceeds 120bpm or is short of 50bmp (Figure 1(b), left hand side).

To make it easy for Susan's family to recognize the prescribed conditions, the doctor names them emergency and concern. uDesign can show either these names or the expressions (expressions are shown in the figure). The doctor also discusses the possibility of involving Susan's family as first-line responders to the conditions above, notwithstanding alerting emergency services.

Later at home, Susan discusses the doctor's prescription with her son John and they agree on alerting John if either condition is observed, and on alerting the emergency services in the event of an emergency, or if a concern condition arises but John is not available.

To coordinate the activities on his side, John defines the John'sWatch box where he includes services to follow his location and determine if he is available, and alert him over the cell phone network. A location service also helps determine the best device to map the PlayVideo service. John leaves the video pipe unhooked, to preserve Susan's privacy, planning to establish the connection only if the need arises. Alternatively, John could have used valves to control the flow of video on the pipe (see the next example).

C. *Example 2: Surveillance in John's home*

John has recently made arrangements for a dog-sitter to come in during the day and walk his dog. However, John would like to be sure that the sitter does not venture into the

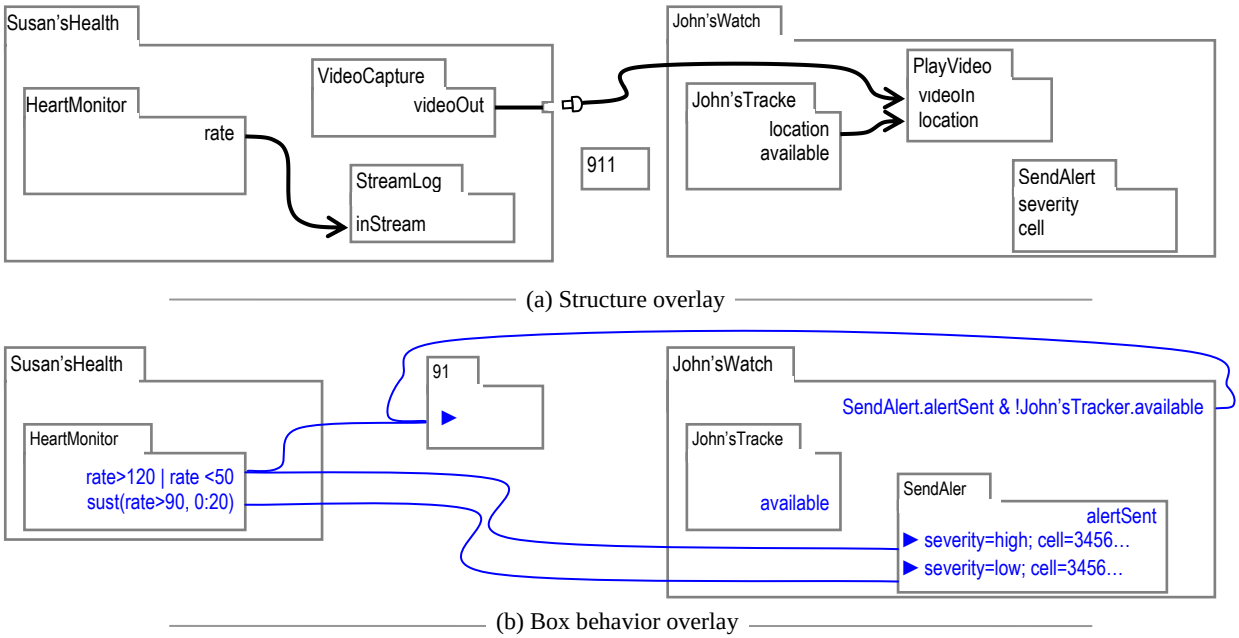

Figure 1. Monitoring Susan's heart

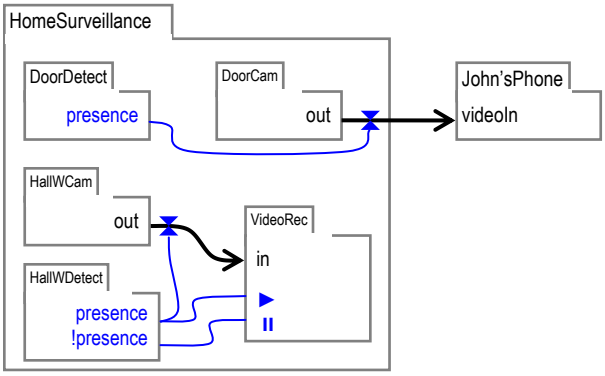

Figure 2. Surveillance in John's home

private areas of the house. After work, John buys a couple of uDesign-enabled cameras and motion detectors. Upon powering up these devices at the home, uDesign's wireless discovery mechanisms find them, and John is able to assign them unique names within his house.

John deploys one camera and motion detector by the kitchen door, where the sitter will be coming in, and another camera and detector in the hallway leading up to the main part of the house (**Error! Reference source not found.**). Instead of installing a uDesign-enabled electric opener for the kitchen door, John simply provided the sitter with a key.

To be aware of the sitter's movements, John uses uDesign to pipe the output of the door camera to his cell phone, places a valve on that pipe so that video only flows when someone is detected in the door area. John's cell phone will alert him of incoming video.

For the hallway camera, John chooses to record its output when a presence is detected, which John may review upon returning home. This is accomplished by placing a valve on the output of the hallway's camera, saving the home's wireless network from continuously piping video when no-one is in the hall.

We have given two examples of how uDesign could be used to define everyday, non-computer related activities. In [15] we describe in more detail the implementation of uDesign on our current Aura task infrastructure. Future work will involve verifying that this design is simple enough to be used by a broad range of users.

III. System Design

In their daily lives users move between different domains and activities must seamlessly move with them. This raises significant security challenges that must be addressed at the system design level, and which differ from traditional views of security in the following ways:

Spectrum of trust. Rather than designating a particular environment as either trusted or untrusted, users in a ubiquitous environment may be willing to partially trust some environments. For example, at a local coffee house, a user might be willing to conduct activities such as lecture presentation or entertainment, but not be willing to pay the household bills or manage their bank accounts.

Ease of use. Users do not want to manage multiple passwords or accounts for different environments. Having to do this goes against the ethos of the overall project, the aim of which is not to distract users with continual management of their computing access.

Flexibility. The means of authenticating people and services in environments needs to be flexible, so that different authentications services (password, face recognition, fingerprint readers, etc.) can easily be added and used as services in environments.

We have developed an approach that adds security to the existing protocols within the Aura activity management layer to deal with these environments. Our approach to securing activities is multi-pronged, and involves:

1. Having the user define *personae*, which are groups of activities that can be used in environments, and assign trust levels to personae that specify the types of environments activities in those personae can be accessed.
2. Having protocols in place that allow suppliers of services to find components in the environment to register with and set up a secure session with.
3. Establishing the trust level that those components have in the environment.

A. Personae

A persona is a group of activities that can be considered to make up a particular role for a user. For example, a teaching persona might include activities to prepare lectures, assign grades, access online class notes; a meal preparation persona might include activities to manage the pantry, plan meals, etc. We hypothesize that the concept of persona strikes a balance in dealing with the management of the multitude of activities that they have, one of which is trust. We plan to verify this through user studies.

B. Security design

As mentioned above, the consequence of having activities in a ubiquitous environment raises some unique security aspects that need to be addressed to allow users to interact with activities in multiple environments. The security objectives that we have identified are:

- Services that are registered with an environment need to be identified and authenticated.
- Each environment must have components that can be located and that manage the security in that environment.
- Users must have some guarantees about the privacy of the data associated with activities.
- It must be easy to add services and users to environments.

To deal with these objectives, we identify two components that must be in every environment: *Speakeasy*, which has the role of authentication server in an environment, and *Environment Manager (EM)* which has the role of application server in the environment. These two components are part of any environment, and listen to well known ports in that environ-

ment. To authenticate with an environment, suppliers of services engage in the Environment Management Binding Protocol (EMBP), that (a) identifies the EM for the supplier's location, (b) establishes a secure session with that EM, and (c) establishes the trust level that the environment has on the supplier. For a supplier to be used in an environment, it must register its services with an EM in the environment. To locate an appropriate EM, when a supplier starts up it requests a session with an EM through a trusted Speakeasy (which is known to it through configuration files). This Speakeasy might be the local speakeasy, or one in a trusted environment. It encrypts this request using the public key of that Speakeasy, among other things proposing a symmetric key that should be used for the resolution protocol. The Speakeasy replies with the location of an EM, the key to use in communication with the EM, and a ticket to the EM. This reply is encrypted using the proposed symmetric key that the supplier passed in the original request. Once the location, key, and ticket for the EM are available to the supplier, it has all that it needs to establish a secure communication channel with the environment manager to engage in the other protocols that allow it to register and be recruited by the environment. Engaging in these protocols also results in key generation to secure these communications, based on the original EMBP interaction.

The use of this protocol means that users do not need to manage multiple passwords for accessing services in the environment, and that any data that is passed as part of the protocol is private to the level of trust that the user associates with that environment.

C. *Authenticating users in the environment*

One missing piece to the security design that is an area of future work is being able to authenticate users into an environment and to gain access to personae. While we have designs for this that are similar to the means of authenticating services, these have not yet been verified or implemented.

IV. Future Work

In this paper, we described the status of our project activity-oriented computing. We are in the process of refining the designs that are described in the two previous sections. The next step will be to develop a prototype system, using our infrastructure for task-oriented computing which was developed as part of the Aura project [17] as a starting point. That will allow us to evaluate and refine our architecture and activity definition and representation.

References

[1] Glenn Judd and Peter Steenkiste, Providing Contextual Information to Pervasive Computing Applications, IEEE International Conference on Pervasive Computing (PERCOM), ACM, March 2003.

[2] Christian Becker and Frank Durr, On Location Models for Ubiquitous Computing, Personal and Ubiquitous Computing, Volume 9, Issue 1, pp 20-31, Springer.

[3] M. Sheshagiri, N. Sadeh and F. Gandon, Using Semantic Web Services for Context-Aware Mobile Applications, MobiSys 2004 Workshop on Context Awareness,ACM, June 2004.

[4] D. Salber, A. Dey and G. Abowd, The Context Toolkit: Aiding the Development of Context-Enabled Applications, Proceedings of the 1999 *Conference on Human Factors in Computing Systems (*CHI'99), ACM, 1999.

[5] B. Schilit, N. Adams, R. and Want, Context-Aware Computing Applications. Proceedings of the Workshop on Mobile Computing Systems and Applications, IEEE, 1994.

[6] Bradley Rhodes, Nelson Minar and Josh Weaver, Wearable Computing Meets Ubiquitous Computing, The Proceedings of The Third International Symposium on Wearable Computers (ISWC '99), IEEE, October 1999.

[7] Joao Pedro Sousa, Scaling Task Management in Space and Time: Reducing User Overhead in Ubiquitous Computing Environments, Ph.D. Thesis, Department of Computer Science, Carnegie Mellon University, 2005.

[8] Joao Pedro Sousa and David Garlan, Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments, Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture (WICSA 2002), May 2002, pp 29-43.

[9] N. Davies, K. Mitchell, K. Cheverest and G. Blair, Developing a Context Sensitive Tourist Guide, First Workshop on Human Computer Interaction with Mobile Devices, 1998.

[10] Jie Yang, Weiyi Yang, M. Denecke and A. Waibel, Smart sight: a tourist assistant system, 3rd International Symposium on Wearable Computers, October 1999.

[11] L. Arnstein and S. Sigurdsson and R. Franza, Ubiquitous Computing in the Biology Laboratory, Journal of Lab Automation (JALA), Volume 6, Number 1, March 2001.

[12] G. Abowd and A. Bobick and I. Essa and E. Mynatt and W. Rogers, The Aware Home: Developing Technologies for Successful Aging, Proceedings of AIII Workshop on Automation as a Care Giver, July 2002.

[13] S.S. Intille, Designing a home of the future, IEEE Pervasive Computing, April-June 2002, pp 76-82.

[14] Center of Future Health, The Smart Medical Home at the University of Rochester, http://www.futurehealth.rochester.edu/smart_home

[15] J.P Sousa, B, Schmerl, V. Poladian and A. Brodsky. uDesign: End-User Design Applied to Monitoring and Control Applications for Smart Spaces. In *Proceedings of the 2008 Working IFIP/IEEE Conference on Software Architecture*, Vancouver, BC, Canada, 18-22 February 2008.

[16] J.S. Gero. Categorizing Technological Knowledge From a Design Methodological Perspective. Conference 'Technological Knowledge: Philosophical Reflections', Boxmeer, The Netherlands, 2002.

[17] David Garlan, Daniel Siewiorek, Asim Smailagic and Peter Steenkiste, Project Aura: Towards Distraction-Free Pervasive Computing, IEEE Pervasive Computing, April-June 2002, Volume 1, Number 2, pp 22-31.