

# **Proceedings of First International Workshop on Computer Support for Human Tasks and Activities**

**April 20, 2004  
Pervasive 2004  
Vienna, Austria**

## **Editors:**

**Jakob E. Bardram  
Henrik Bærbak Christensen,  
David Garlan  
João Sousa**

**University of Aarhus, Denmark  
University of Aarhus, Denmark  
Carnegie Mellon University, USA  
Carnegie Mellon University, USA**

## **Proceedings of**

# **Workshop for Computer Support for Human Tasks and Activities.**

**Vienna, Austria, April 20, 2004**

This report contains the proceedings of the 2004 workshop on Computer Support for Human Tasks and Activities. All papers submitted to the workshop have been reviewed by two members of the Program Committee as well as by the workshop organizers.

The workshop aims at exploring issues related to improving the computational support for human tasks and activities, with a special focus on infrastructures, software architectures, models of user tasks, and on the challenges associated with designing and implementing those.

The goals of this workshop are the following:

1. To build a network of researchers and practitioners working on aspects of task/activity-based computing
2. To create awareness about ongoing research and to identify commonalities
3. To foster collaboration among participants

## **Program Committee**

Jacob E. Bardram,  
Henrik B. Christensen,  
Scott Fahlman,  
David Garlan,  
Klaus Marius Hansen,  
Norman Sadeh,  
Ulrik Pagh Schultz,  
João P. Sousa,  
Manuela Veloso,

University of Aarhus  
University of Aarhus  
Carnegie Mellon University  
Carnegie Mellon University  
University of Aarhus  
Carnegie Mellon University  
University of Aarhus  
Carnegie Mellon University  
Carnegie Mellon University

## **Organizers**

Jacob E. Bardram,  
Henrik B. Christensen,  
David Garlan,  
João P. Sousa,

University of Aarhus  
University of Aarhus  
Carnegie Mellon University  
Carnegie Mellon University

## Table of Contents

### Infrastructures

A task-based approach to data and context management in a virtual personal server space. 1  
*Aaron Quigley, Bjorn Landfeldt, and David West.*

Hermes: A Software Framework for Mobile, Context-Aware Trails Applications. 6  
*Cormac Driver and Siobhán Clarke*

Supporting Activity-Based Computing using Dynamically (De)Composable Mobile Applications. 12  
*Kari S. F. Schougaard and Ulrik P. Schultz.*

### Modeling

Situated Profiles for Aware Environments. 17  
*Martijn Vastenburger.*

A Service Shaping Approach for Task-based Computing Middleware. 22  
*Jin Nakazawa, Jun'ichi Yura, and Hideyuki Tokuda.*

Getting things done: Towards a conceptual framework to support the management of multiple activities. 27  
*Victor M. González and Gloria Mark.*

### Interactions

User-centered support to localized activities in ubiquitous computing environments. 34  
*Helder Pinto and Rui José.*

Designing for Physical-Virtual Activities. 43  
*Thomas Pederson.*

### Open Issues and Lessons Learned

Three Desirable Properties of Activity-Oriented Intelligent Systems. 49  
*Fahd Al-Bin-Ali and Nigel Davies.*

Open Issues in Activity-Based and Task-Level Computing. 55  
*Jakob E. Bardram and Henrik Bærbak Christensen.*

# Workshop Program

## Session I – Infrastructures

A task-based approach to data and context management in a virtual personal server space.

*Aaron Quigley, Bjorn Landfeldt, and David West.*

Hermes: A Software Framework for Mobile, Context-Aware Trails Applications.

*Cormac Driver and Siobhán Clarke.*

Supporting Activity-Based Computing using Dynamically (De)Composable Mobile Applications.

*Kari S. F. Schougaard and Ulrik P. Schultz.*

## Session II – Modeling

Situated Profiles for Aware Environments.

*Martijn Vastenburger.*

A Service Shaping Approach for Task-based Computing Middleware.

*Jin Nakazawa, Jun'ichi Yura, and Hideyuki Tokuda.*

Getting things done: Towards a conceptual framework to support the management of multiple activities.

*Victor M. González and Gloria Mark.*

## Session III - Interactions

User-centered support to localized activities in ubiquitous computing environments.

*Helder Pinto and Rui José.*

Designing for Physical-Virtual Activities.

*Thomas Pederson.*

## Session IV – Open Issues and Lessons Learned

Three Desirable Properties of Activity-Oriented Intelligent Systems.

*Fahd Al-Bin-Ali and Nigel Davies.*

Open Issues in Activity-Based and Task-Level Computing.

*Jakob E. Bardram and Henrik Bærbak Christensen.*

# A task-based approach to data and context management in a virtual personal server space.

Aaron Quigley<sup>1</sup>, Bjorn Landfeldt and David West

Smart Internet Technology Research Group, School of Information Technologies,  
University of Sydney, Camperdown NSW 2007, Australia  
{aquigley, bjornl, dwest}@it.usyd.edu.au  
<sup>1</sup><http://www.it.usyd.edu.au/~aquigley>

**Abstract.** This paper presents our vision for an activity based pervasive computing system based on a virtual personal server space. Our target demographic is a group of elder computer users. Using collaborative interfaces technologies; we focus on reminiscence activities for their use in promoting social and intellectual fitness. Current computer applications while designed for single use are not suitable for multi-user, multi-device and multi-modal interaction. Our proposed architecture (middleware) consists of three planes of service (data, context and activity). The context plane mediates access to various services (depending on the current user activity) while optimizing data access around simple activity models.

## 1 Introduction

Our research aims to substitute the classical desktop interface with invisible and ambient interfaces that allow individuals or groups to engage in reminiscence oriented group or individual activities. Our own user studies have identified a broad range of currently used simple technologies and physical objects that stir and support these activities [7]. Rather than focus on specific “applications” our work is taking an activity based approach. We envisage our system being used by individuals or groups moving between the available modalities, across different devices to input, access and manage digital content in process of reminiscence activities [4] [5]. Our goal is to develop new technologies that will be used naturally by incorporating systems such as memory collection, meta-data specification, pen-paper control, voice control, large screen touch-based interfaces, haptic and ambient feedback.

To support such natural environments we have developed our virtual personal server space architecture. The server space incorporates two main planes, the context and data planes, in support of activities. The data plane manages synchronisation, caching, migration, and security as the individual moves through different network topologies.

---

<sup>1</sup>The authors would like to acknowledge the ongoing support of the Smart Internet Technology CRC Australia and the National ICT Australia.

The context plane, as is traditional in pervasive computing, manages device characteristics, profiles, service availability, modalities and learning. In practice, the current activity dictates that these services be offered from devices when decoupled from the network but may use services residing in the nearby computing environment as they become available.

## 2 Middleware (Vision for support in Pervasive Computing)

It is envisaged that in the future, people will be carrying devices which are capable of interacting with pervasive computing environments [6], namely environments instrumented with sensors, numerous I/O devices, compute engines, controllers and actuators [1][3][4]. The major features of this type of pervasive computing environments are:

- The personal devices users carry, which we refer to as virtual personal servers, consist of a collection of local and possibly remote devices
- The personal servers will range from a simple ID wireless drive, USB key, mobile phone, PDA and up to systems akin to the Intel personal server
- The personal servers will act alone or in unison to allow an individual use the information they carry
- They are likely to have no classical user interfaces, but some limited computational power, storage capabilities and network connectivity



Fig. 1. Task-aware context and data plane services running on personal server. This controls the available input and output modalities for set of task functions running in the Personal Area Network (PAN) or the Personal Local Area Network (PLAN), respectively.

A possible usage scenario of this type system is as follows: a young player is engaged in training for a sporting activity. The virtual personal server, while providing various

statistics, audio and video streams through various interfaces are also recording the current sporting event. At a later time, when the child visits her grandparents, she plays it back on a shared screen at their home and shares the experience with the grandparents.

The objective of this project is to develop some of the fundamental techniques for the collection, storage and use of the data in an activity based manner, and the interaction of the virtual personal servers and pervasive environments. Specifically:

- Seamless operation of virtual personal servers across heterogeneous pervasive environments
- Adaptive/ambient interfaces

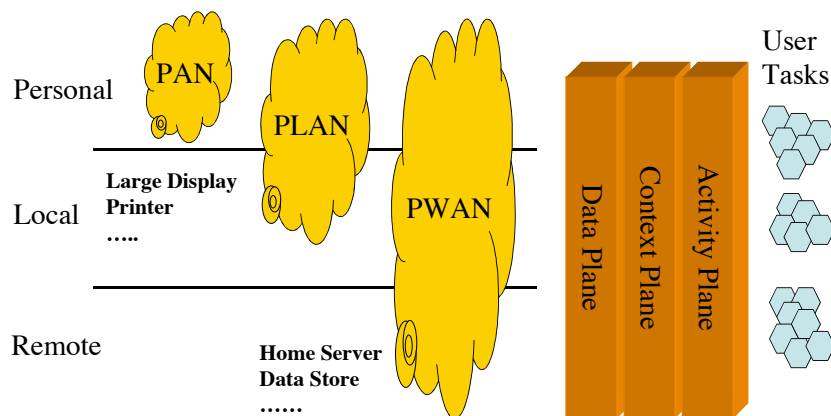
To achieve the seamless operation we are investigating various context-aware programming models. This will allow us to take the current context of the particular human activity or situation into account for data and application services. Context is any information that can be used to characterise the situation of an entity and includes information from the sensed environment (environmental state) and computational environment (computational state) [2][6].

The adaptive interface aspects of this project rely on the pervasive environment to mediate access and control of the available modalities. Thus this work will focus on the seamless interaction with the virtual personal server across a number of devices in a multi-modal multi-device manner. To demonstrate our virtual personal server, we aim to develop a number of example pervasive computing environments using the surrounding computing infrastructure for accessing the virtual personal server. Our preliminary environment will consist of a pervasive environment that contains, wireless connectivity, local server, RFID reader, shared multi-user touch sensitive input device, gesture recognition and a few personal screens. This prototype pervasive environment will allow a number of individuals to manage multimedia content from their virtual personal servers in a collaborative reminiscence activity.

We have defined a single management plane, which has the responsibility of organising, synchronising and maintaining the information over the different available devices in the overall data storage system. Users interact with the system through a simple and well-defined API that allows them to put objects (during a particular activity) into storage and to retrieve objects from storage, according to a defined activity. When adding objects to the data storage, users have the option of indicating the importance of the object, which in turn affects where the object will be stored in the system. This way, increasing importance will translate to increased accessibility and possibly also increased replication. By including the notion of activity modeling into our data plane specification, we can decouple the various applications from the actual data accessed and instead associate it with a rich context model, encoded as an activity model.

Consider the following scenario. A user has got a file with information about bank accounts, passwords etc. that is marked as highest importance since it is used on a

daily basis. The user has a personal server, a digital camera and a USB memory key ring as personal devices with storage capacity. The camera is regarded as temporary storage space by the system and therefore the file is not replicated onto it. However, the USB key ring is categorised as a primary storage space for vital information since the user is always carrying it with him and therefore the file is synchronised and stored on this device every time the device becomes available to the system. The file is also stored on the personal server since it has a higher availability than the USB key ring even if the user might not always carry it with him.



**Fig. 2.** Data and context planes transcend the current network topology. Data stored while in a PAN is synchronised when a PWAN becomes available or to a data store in a PLAN for performance reasons. Context plane manages, computing context, user context, physical context and time context, for activity and multi-modal interface support.

### 3 Biographies

Dr Aaron Quigley is a Senior Research Fellow in the School of Information Technologies University of Sydney. Dr. Quigley was awarded his PhD in Sept 2002. Dr. Quigley commenced his studies at Trinity College Dublin in Ireland and finished his PhD at the University of Newcastle Australia. Since joining the University of Sydney he has developed a pervasive computing laboratory based on a large equipment grant. Along with over 20 publications, he is involved with the supervision of 4 PhD students. Dr. Quigley is now the project leader for two research projects within the Intelligent Environments program. Project Bluestar is a joint project with Telstra Research Laboratories researching a scaleable privacy-centric location aware system by in-door location sniffing. Project Nightingale is a joint project with the National ICT Australia for the development of natural pervasive computing interfaces in a virtual personal

server space. His interests include distributed systems, middleware, location-based computing, human computer interaction and pervasive computing.

Dr. Björn Landfeldt (MIEEE MIEICE) is a Senior Lecturer in the School of Information Technologies and Department of Electrical Engineering (University of Sydney). Dr. Landfeldt was awarded his PhD in May 2000. Dr Landfeldt started his studies at KTH in Sweden and finished his PhD at UNSW in Australia. During his studies he was running his own consultant company in parallel within the area of wireless networking. After his graduation, he worked for Ericsson Research Networks and Systems in Stockholm before returning to Australia where he currently holds a position at the University of Sydney. His interests include wireless networks, IP mobility management, QoS, Peer-to-Peer networks, Middleware Architectures and IPv6 migration methods.

David West is a Research Associate with the National ICT Australia and the Smart Internet Technology Research Group in the School of Information Technologies (University of Sydney). David holds a Masters degree in Networks and Distributed Systems from the University of Dublin, Trinity College. Along with graduating at the top of both his postgraduate and undergraduate classes, David is a University Medalist and scholar of the University. His interests include computer vision, ad-hoc networking, mobile computing, human computer interaction and web engineering.

## References

1. Brumitt, B., et. al "EasyLiving: Technologies for Intelligent Environments". In Proc. of 2nd International Symposium on Handheld and Ubiquitous Computing, HUC 2000, pages 12-29, Bristol, UK, September 2000. Springer Verlag.
2. Douglas B. Moran, Adam J. Cheyer, Luc E. Julia, David L. Martin, Sangkyu Park, "Multimodal User Interfaces in the Open Agent Architecture", Intelligent User Interfaces 1997, pp. 1-9.
3. Edwards K, Grinter R. "At Home with Ubiquitous Computing: Seven Challenges" in Proceedings of UbiComp 2001, LNCS 2201, pp. 256-272, 2001. Springer-Verlag Berlin Heidelberg 2001.
4. Fox, A., Johanson, B., Hanrahan, P. & Winograd, T., (2000), "Integrating information appliances into an interactive workspace", IEEE Computer Graphics & Applications, 20:3, May/June, 54-65.
5. Hindus D, Mainwaring S., Leduc, N, Hagström A.,E, Bayley O "Casablanca: Designing Social Communication Devices for the Home" in Proceedings of CHI'01, Seattle, March 31-April4, 2001, ACM Press
6. Kidd, Cory D., Robert J. Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth Mynatt, Thad E. Starner and Wendy Newstetter. "The Aware Home: A Living Laboratory for Ubiquitous Computing Research" In the Proceedings of the Second International Workshop on Cooperative Buildings.
7. Peter Risborg and Aaron Quigley, Nightingale: Reminiscence and Technology - From a user perspective, In the Proceedings of OZeWAI 2003, pp.12-19. December 2003

# Hermes: A Software Framework for Mobile, Context-Aware Trails Applications

Cormac Driver

Siobhán Clarke

Distributed Systems Group, Computer Science Department  
Trinity College Dublin, Ireland  
{Cormac.Driver, Siobhan.Clarke}@cs.tcd.ie

**Abstract.** The growth of ubiquitous computing has given rise to a range of possibilities for activity-based application development. Technologies for mobile computing and context-awareness can be combined to create innovative ubiquitous computing applications, notably those based on the concept of a *trail*. A trail is a collection of locations, together with associated information and activities, and a dynamically reconfigurable recommended visiting order. This paper describes work to date on Hermes, a software framework for mobile, context-aware trails-based applications which includes support for automatic context-driven dynamic reconfiguration of active trails.

## 1 Introduction

At the most general level, a trail can be thought of as a collection of locations, together with associated information and activities, and a dynamically reconfigurable recommended visiting order. The trail is a collection of connected locations rather than a strict sequence since it may contain alternative sub-routes to cater for such variables as different modes of transport or other user preferences. Trail activities can be either mandatory or optional, and each activity has a priority value which is used to rate its importance relative to other activities. Trails underpin a wide range of useful applications for a mobile user who has a set of activities that may or should be carried out throughout the day at different locations. Combining the trails concept with mobile, context-aware technology creates opportunities for innovative activity-based application development. Mobile, context-aware applications are those that run on wireless devices e.g., PDAs, and have an awareness of the physical and social situation in which they are deployed. In developing such applications it is important to shield the user from the underlying complexities of the paradigm, e.g., intermittent network connectivity and context-based decision making, allowing them to focus solely on their trail [1]. Examples of trails applications that are both mobile and context-aware include tour guides, courier support/management systems, basic route planners, treasure hunt games and student support systems.

Considering the example of a mobile, context-aware trails-based application for assisting students with performing campus-based activities, a user on her first day at college requires her PDA-based application to create a personalized route from the student registration point, to any library on campus, and on to a specific lecture theatre. Each point on the trail has an associated task (and subtasks) i.e., register,

submit library bond, attend introductory lecture with course director. The trail is presented on an augmented digital map of the campus with routes drawn between each activity point, and the user's current location denoted. At all times, the user's personal and environmental context can dictate that the trail being followed is dynamically reconfigured to reflect the relationship between their trail activities and the current state of their environment. Continuing the example above, the user is en route to the registration point with the purpose of registering. The application senses that there is a significant number of other system users currently attempting to register and that proceeding with the registration activity has the potential to make her late for her next activity, which has a higher priority than registration. The user's trail is dynamically reconfigured to reflect the prevailing context state. With the registration activity assigned to later in the day, the user now has time to go to a library that happens to be close to her current location. The trail is reconfigured to include the library activity, with the registration activity scheduled for after the introductory lecture. The user is informed and directed to the library.

As evidenced by the state of the art described later in this document, the mobile, context-aware trails-based applications that have been implemented to date are not diverse in terms of target audience or sophisticated in terms of ability to accurately reflect the user's environment. Tourist guide applications featuring trail generation and basic reconfiguration capabilities represent the current state of the art.

We are investigating the development of a software framework to support development of mobile, context-aware trails-based applications, which includes support for automatic dynamic trail reconfiguration. The framework will provide developers with components containing structure and behaviour common to all trails-based applications, facilitating the development of a diverse range of realistic trails applications that better fit the end-user model.

The major research challenges in this area concern a) capturing trails application components in a generic manner for reuse and extension in future trails applications b) providing trail dynamic reconfiguration capabilities to ensure the provision of a more consistently realistic representation of the user's environment and c) managing the trade-off between mobile device limitations and the significant application processing requirements of trail generation and dynamic reconfiguration behaviour.

The remainder of this document is as follows. Section two contains a brief state of the art in the area of mobile, context-aware trails-based applications. Section three contains a description of our work on the Hermes project. The concluding section contains author biographies.

## **2 State of the Art**

The GUIDE system [2], a mobile, context-aware trails-based tourist guide application, is the most advanced research effort in the field. The system contains a facility to request a structured tour based on a set of attractions that the user wishes to visit in the city of Lancaster, UK. Following tour generation the visitor is presented with a recommended sequence for visiting the chosen attractions. The ordering of the tour can dynamically change while the user is following it e.g., if the user stays longer

than anticipated at a certain attraction. In relation to the Hermes project the GUIDE system has two main drawbacks. Firstly, the dynamic trail reconfiguration behaviour is quite basic, only occurring “periodically”. Only positional and temporal context are sensed automatically, giving rise to the possibility of active trails being based on stale information, as the remaining context must be manually uploaded. Secondly, in terms of software, the system was designed from the ground up with only the GUIDE tourist application in mind. This means that the opportunity to reuse GUIDE components in future development efforts is limited.

As well as the GUIDE system there are a number of other research efforts contributing to the state of the art. The Cyberguide project [3] involved the development of two basic context-aware tour guide prototypes. The Stick-e Document [4], essentially an electronic post-it note, can be used to build basic mobile, context-aware tour guide applications. However, implementing a dynamically reconfigurable tour using Stick-e technology requires the specification of a note for every conceivable tour situation (user/context combination) before the application is deployed. A mobile aquarium tour guide system is described in [5]. This system features trail creation functionality but does not automatically sense context. HyCon [6] is a framework for mobile, context-aware hypermedia. HyCon facilitates the creation of mobile applications featuring context-aware searching, browsing, linking and annotation. A prototype application, the HyConExplorer, exhibits these behaviours. Using the framework it is possible to specify trails through collections of objects. In practice this means trails through collections of locations, with each location being a point on the trail with associated categorized material. This approach is suitable for building basic trails such as the field trip for school children described in [6], which has various layers of information at each point and the option to annotate each point with further information. More complex trails, such as those involving intelligent context-based trail generation, activity dependencies, mandatory and prioritized tasks, collaborative context and automatic dynamic reconfiguration, do not appear to be supported.

### 3 Hermes

“Frameworks are reusable designs of all or part of a software system described by a set of abstract classes and the way instances of those classes collaborate” [7]. Framework development is an iterative process driven by multiple application development efforts with framework refinements following the development of each application. The result of this process is a generic framework that can cater for the requirements of many different applications within a specific domain. The approach taken on the Hermes project adheres to this method, implementing a series of mobile, context-aware trails-based applications and refining the framework as appropriate. Below we describe our work to date.

#### *Analysis*

We began the framework development process by specifying requirements for four dissimilar mobile, context-aware trails-based applications. These were a courier

support system, a city route planner, a team-based treasure hunt game and a student support system. Using these requirements we constructed design diagrams (UML) for each application. An examination of the requirements documents and design diagrams exposed the commonality that exists between the applications in terms of trails-related functionality. Table 1 presents the results of the application analysis. The first row contains a list of application features which we believe to be representative of what a generic mobile, context-aware trails-based application should support. In terms of context-awareness we are referring to the use of both personal and environmental context e.g., location, time, weather, user preferences, trail history and awareness of other application users. The first column lists the applications examined. Our analysis illustrates that although the set of trails-based applications have significantly different high-level requirements, the high level of commonality that exists between the requirements for trails-related functionality renders framework support valuable.

	Navigation	Context-Awareness	Trail Adaptability	Compulsory Activities	Optional Activities	Group Communication
Route Planner	✓	✓	✓	-	✓	-
Courier Support	✓	✓	✓	✓	-	✓
Treasure Hunt	✓	✓	✓	✓	✓	✓
Student Support	✓	✓	✓	✓	✓	-

**Table 1. Analysis of mobile, context-aware trails-based applications**

#### *Initial Framework Design*

Figure 1 shows a UML component diagram containing the components found to be common across the four applications designed in the first phase of our work. The UML class diagrams for each of the applications were analysed for commonality and cohesive elements were grouped into components.

The Context Engine component is responsible for providing context data. This involves sourcing relevant context data from numerous distributed sources and rendering and distributing it in a manner appropriate to the application under consideration.

The Trails Engine component is responsible for managing the lifecycle of a trail. Trails are created based on input data from both the user and context engine. Following this, the component is responsible for managing the trail as the user embarks on it, ensuring the trail's relevance at all times.

The Trail component provides a software representation of a trail that is created and followed by the user. This component is manipulated by the Trails Engine component.

The Dynamic Reconfiguration component is responsible for monitoring environmental context throughout the active lifetime of a trail and taking the appropriate actions to ensure that the trail consistently reflects the user's environment. The component uses context data to provide the Context-based Inference component

with information and then uses the behaviour contained in this component to arrive at decisions regarding active trail reconfiguration. The Dynamic Reconfiguration component then advises the Trails Engine component regarding appropriate trail alterations.

The Context-based Inference component, which contains an Inference Rules subcomponent, is responsible for the aggregation and interpretation of context data. Context data, received from the Dynamic Reconfiguration component, is analysed with a view to discovering which actions need to be taken in order to keep the active trail in a state that accurately reflects the environment in which the user exists. The Inference Rules subcomponent contains rules to assess real-time context data.

The GUI component is responsible for presenting the trail application to the user and providing a way for the user to communicate with the underlying application and vice versa.

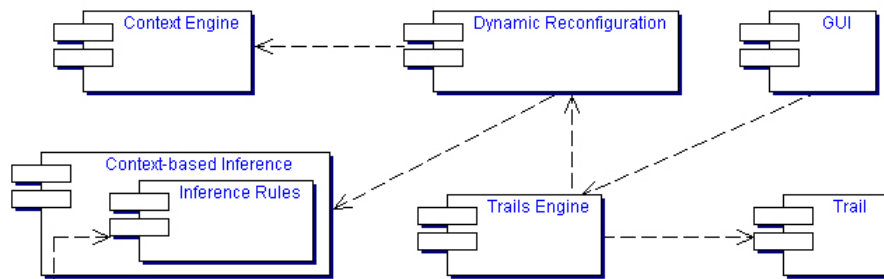


Fig. 1. Component diagram of initial framework design

### 3.1 Initial Framework Implementation

In order to build the first version of the Hermes framework we are implementing a student support application based on mobile, context-aware trails-based technology. The application will be used by new students at Trinity College Dublin during their initial period at the college and caters for the generation of campus-wide trails including both compulsory tasks such as registration and optional activities such as visiting college buildings and joining societies. The trails are both based on and affected by environmental and personal context, and are dynamically reconfigurable.

The first version of the framework will be further refined and developed into a generic framework through the implementation of the remaining three applications.

### Author Biographies

Cormac Driver is currently in his 2<sup>nd</sup> year of PhD research with the Distributed Computing Group (DSG) at Trinity College Dublin’s Computer Science Department

following the completion of a M.Sc. in Networks and Distributed Systems in 2002. His focus area is dynamic reconfiguration of mobile, context-aware trails-based applications.

Siobhán Clarke is an academic with the Distributed Systems Group in the Department of Computer Science, Trinity College Dublin. Her research interests are programming models and middleware for mobile, sentient, context-aware systems. She lectures at postgraduate level on software engineering for distributed systems, and middleware for ubiquitous computing.

## References

- [1] Z. Wang, D. Garlan. "Task-Driven Computing". School of Computer Science, Carnegie Mellon University, Pittsburgh, USA. CMU-CS-00-154, May 2000.
- [2] K. Cheverst, N. Davies, K. Mitchell and A. Friday. "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project". In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking, p. 20-31, Boston, Massachusetts, 2000. ACM Press.
- [3] G. Abowd et al. "Cyberguide: A Mobile Context-Aware Tour Guide". ACM Wireless Networks - Special issue: Mobile Computing and Networking: selected papers from MobiCom '96, p. 421-433. New York, 1997.
- [4] P.J. Brown. "The Stick-e Document: a Framework for Creating Context-aware Applications". In Proceedings of International Conference on Electronic Documents, Document Manipulation, and Document Dissemination (EP 96), Palo Alto, California., 1996, pp. 259-272. Proceedings published in Origination, Dissemination, and Design (EP-ODD), vol. 8, no. 1, John Wiley & Sons, New York, 1996, pp. 1-13.
- [5] F. Bellotti, R. Berta, A. De Gloria and M. Margarone. "User Testing a Hypermedia Tour Guide". IEEE Pervasive Computing, Vol. 1, No. 2, April-June 2002.
- [6] N.O. Bouvin, B.G. Christensen, K. Grønþæk, F.A. Hansen. "HyCon: A Framework for Context-aware Mobile Hypermedia". Department of Computer Science, Aarhus University. To appear in New Review of Hypermedia and Multimedia (NRHM). Vol 9. 2003.
- [7] D. Roberts, R. Johnson. "Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks". ACM Pattern Languages of Program Design 3, 1997, pp. 471-486.

# Supporting Activity-Based Computing using Dynamically (De)Composable Mobile Applications

## Position Paper for Support for Human Tasks and Activities'04 (Pervasive'04 Workshop)

Kari S. F. Schougaard and Ulrik P. Schultz

Centre for Pervasive Computing/ISIS/DAIMI  
University of Aarhus  
Denmark  
{kari,ups}@daimi.au.dk

### 1 Introduction

In the era of pervasive computing computers are ubiquitous. Activities carried out by the use of computers can now encompass various devices and be carried out while moving as well as in front of the personal computer. The activity-centered computing paradigm has been suggested as better than the document-centered paradigm for pervasive computing: it crosscuts work with different kinds of entities and sets the focus on the user and the user's tasks [1].

In a domain such as healthcare, activities are fundamentally mobile: doctors and nurses move throughout the hospital as their work progresses. Hence, the computer must support mobility of activities. For this reason, we consider it important to investigate whether a computing platform that embraces mobility at a fundamental level supports implementing software for this new paradigm.

This paper describes our initial design ideas for supporting activity-centered computing in the Pervasive Object Model Project (POMP), a project which concerns the development of a platform for pervasive computing applications. We first describe POMP, then present the notion of activity-based computing, and last investigate how activity-based computing can be implemented in POMP.

### 2 Background: POMP

POMP is an ongoing effort to implement a language and a virtual machine for applications that execute in a pervasive computing environment consisting of heterogeneous devices [2]. We see application mobility as a central concern in such an environment; the activity the user is currently performing can be supported by applications that migrate to devices in close physical proximity to the user or in close virtual proximity<sup>1</sup> to another application.

The Pervasive Object Model virtual machine (POM) supports applications split into coarse-grained, strongly mobile units that communicate using method invocations through proxies. POM is written in Java and supports execution of applications written in a minimal dialect of Smalltalk. We are currently investigating efficient execution of mobile applications and construction and de-construction of applications composed from strongly mobile units.

In POM, a mobile entity is referred to as a *system* (in the sense of a self-contained entity). Systems are normally employed in POM to modularize applications. A

---

<sup>1</sup> Our language and virtual machine supports the notion of *virtual distance*: the distance between two computing systems measured in terms of bandwidth and latency.

single virtual machine can host multiple systems that communicate using proxies; communication through proxies can be structured using “plugs” which are akin to interfaces in the POMP module system. A system can at any time migrate to some other virtual machine where it continues its execution (e.g., strong migration of all threads); it is thus a set of objects that migrate as a single unit.

POM applications are normally structured into several systems that communicate using proxy calls. Services that require access to native resources (e.g., access to GUI or to files) are typically isolated in a single system, which is then tied to the current location because of its use of native resources. The remaining application is then free to migrate to other virtual machines while maintaining an indirect reference to the services of the original virtual machine (in the form of proxies). Alternatively, the application can decide to obtain access to the services of the virtual machine that it has migrated to, for example by replacing a remote GUI with a local one. We intend for such reconfiguration of an application to be handled using the POMP module system.

The POMP module system is not yet completely implemented, but is designed to allow dynamic construction and de-construction of applications. A system is equipped with a number of *plugs*, each of which exports a set of proxy objects and imports a set of proxy objects. Each proxy object can be named according to the role that it plays in the plug. For two plugs to be compatible (e.g., two systems can communicate through this plug), the sets of exported and imported proxy objects must match, including any named roles. An imported proxy object can only be used when connected to another system (when not connected, all calls block). The programmer can define code to be executed when two systems connect using a plug and when two systems *disconnect*. Disconnection causes all imported proxies to become disconnected; pending calls are terminated with an exception on the caller side but allowed to continue on the callee side (although any value from the call is lost — the thread of control can be said to have been cut in two).

### 3 Activities

Our notion of activities is inspired by Christensen and Bardram [1]. An activity or a task is a process (either simple or composed of several subtasks) carried out by a person. Some activities are linked to certain locations and some are linked to certain physical artifacts, while others can be carried out with access only to public computers or mobile devices. Interaction with non-computerized objects can also be part of an activity, but activities consisting only of interaction with non-computerized objects will not be represented in a computer system. Orthogonally to the device demands, some activities have to be carried out at a certain time. *Activity* covers a range of action processes which can be connected to time, location and artifacts, and are carried out by a person.

Activities are human-centered. This means that even when removed from the context, a person is still carrying out an activity. The activity can nonetheless be taken over by or transferred to another person. Activities can also be shared when several persons work together to complete a task, but the important point is that at a given point in time each activity is owned by a specific person or group of persons. This understanding of *activity* gives rise to a set of demands on an activity-based computing environment.

The first computerized representation of an activity that comes to mind is a collection of applications, or rather, their functionality and current state. Activities follow a person and therefore, as pointed out by Sousa and Garland [3], the infrastructure should support the transferring of activities by finding and configuring locally available applications and devices; this is necessary to let the activity follow the user while adapting to surroundings.

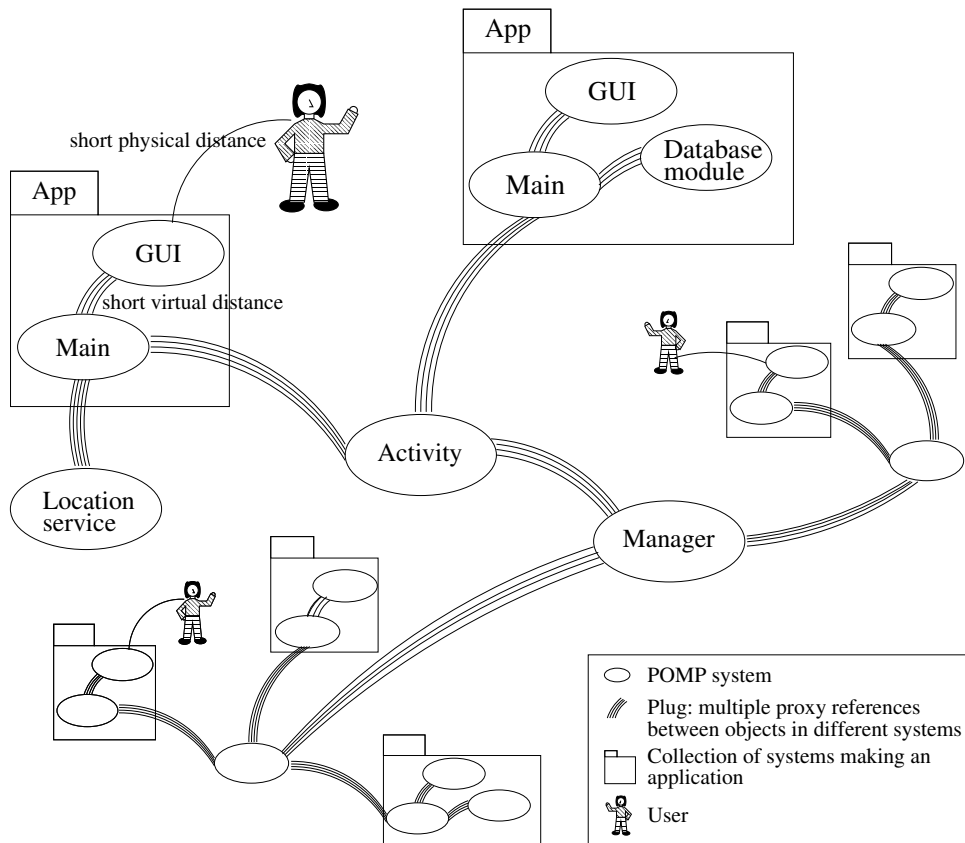


Fig. 1. Activity architecture in POMP.

Users should have a way of managing the activities they own or take part in; but to lighten the burden of defining new activities, location- or artifact-dependent activities can suggest themselves when the user is in the right context. The management of activities should include support for cooperation with others and for changing the ownership over time.

#### 4 POMP-based Activity Architecture

We now explain our ideas for a POMP-based architecture of an activity-centered computing system (ACCoS). We envision that an ACCoS encompasses both the underlying infrastructure with activity-, file- and context services as well as the user-level applications. The main elements of an ACCoS are the activities, activity managers (one for each user), the applications that support the activities, and the services in the underlying infrastructure.

Figure 1 shows an overview of our POMP-based ACCoS. Each activity is represented using a POMP system which acts like a mediator between the components of the activity and also handles communication with the activity manager (described later). The mediation allows the components of the activity to coordinate whether they are currently in use (they should only be visible if they are used for the activity currently pursued by the user), decide when to migrate to a different device, and reconfigure themselves in a consistent manner e.g. for a different kind of GUI. Moreover, we imagine that certain application-specific events, such as updates to

the data that an application is using, can have an interest to the other applications of the activity and therefore should be communicated through the mediator system which would implement a local publish-subscribe functionality.

The activity manager is the users portal to manage activities: it support interaction and coordination. The manager supports inviting other users to or expelling them from an activity. It also supports making new activities and closing down old ones. Context-dependent activities that are suggested by the system to a user being in the right context are suggested through the activity manager as well.

The connectors between the systems are plugs, which for example allows an application part to disconnect one system implementing a workstation-sized GUI and plug in another one implementing a handheld-sized GUI; the look of the GUIs can thus be completely different, although the objects that handle the roles specified by the plug have to be present.

Plugs give the platform a uniform way of interacting with systems that represent activities. For example, each activity should have objects that take care of inviting a user to and expelling a user from the activity. A system which represents an activity can delegate this functionality to other systems in a specific application or manage it locally.

The applications are not explicit in the system, but are represented as collections of systems that collaborate to form the functionality of an application. The applications can adapt to the environments by using local systems or by moving some of their systems.

In POMP the concepts of physical and virtual distances help the programmer implement composite devices. The notion of physical distance is currently being implemented in the POM library, to allow the programmer to refer to devices in physical proximity of the user. The notion of virtual distance is supported in POM to facilitate optimization of applications that distribute across multiple devices in a meaningful way. The optimizations are based on improved use of resources that are virtually close (meaning they have a good connection, bandwidth-, latency- and interruption-wise). An example use is keeping a logging system on the virtually closest reliable server. Activity-centered computing is obviously supported by the notion of physical distance enabling the infrastructure to choose devices in proximity of the user. But the virtual distances also play an important role by keeping the migrating applications in good shape, and facilitating the adaptation to local resources.

In the work of Bardram and Christensen, activity discovery and sharing of desktop application are important issues. Activity discovery would probably not be fundamentally different in a system that supports mobility, although a peer-to-peer based architecture might be preferable. We intend for sharing of desktop applications to be supported by default in the POM GUI (which is currently under development).

## 5 Evaluation

Compared to a centralized activity-based computing infrastructure such as the one presented by Bardram and Christensen, the run-time observable qualities gained by using POMP is that a central server is no longer needed (migration is by nature peer-to-peer) and that applications can be dynamically reconfigured to meet specific demands. The primary development-time observable quality is that application mobility comes for free. Specifically, the individual threads of the application do not need to be coordinated when migrating an application, and it is trivial to design applications that migrate between devices with equivalent resources (e.g., workstations).

**Acknowledgments** Simon Nejmann is currently working on a C++ version of the POM virtual machine. Michael Dalgaard Jepsen and Jan Nielsen are currently implementing a mobile GUI for POMP.

**About the authors** Kari S. F. Schougaard is a PhD student at the Centre of Pervasive Computing, University of Aarhus. Her research interests include pervasive and mobile computing—especially the middleware-, language-, and conceptual framework aspects of these. She participates in the work on supporting pervasive and palpable applications in POMP. Besides she takes part in the home networks theme of the B&O+OO Project at ISIS, where distance-based restrictions on methods are used to control access according to the physical location of the user.

Ulrik P. Schultz is a research associate professor at DAIMI, University of Aarhus, funded by ISIS. His research interests include programming languages, software architecture, implementation and optimization of pervasive computing systems, object-oriented software systems, program analysis, compilers, partial evaluation, and embedded systems. He is manager of the POMP project and the “B&O+OO” industrial collaboration project which aims at developing a complete infrastructure for network-enabled home AV equipment.

## References

1. Henrik Bærbak Christensen and Jakob E. Bardram. Supporting human activities — exploring activity-centered computing. In *Proceedings of the Fourth International Conference on Ubiquitous Computing UBICOMP 2002*. Springer Verlag, September 2002.
2. Kari Schougaard and Ulrik P. Schultz. POMP – Pervasive Object Model Project. 9th Workshop on Mobile Object Systems (Resource Aware Computing), ECOOP 2003, <http://www.daimi.au.dk/~kari/publications/mos03.pdf>, 2003.
3. Joao Pedro Sousa and Davis Garlan. The aura software architecture: an infrastructure for ubiquitous computing. Technical Report CMU-CS-03-183, School of Computer Science, Carnegie Mellon University, 2003.

# Situated Profiles for Aware Environments

Martijn Vastenburg

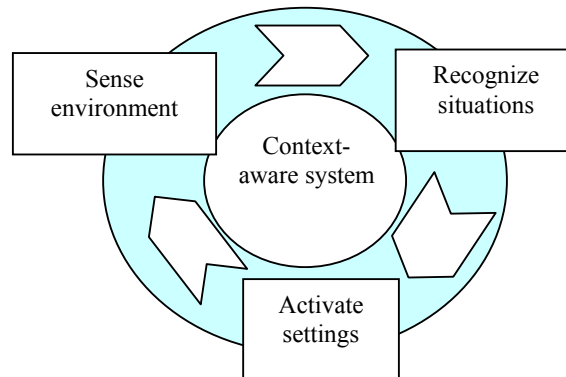
Intelligence in Products Group, Faculty of Industrial Design  
Delft University of Technology  
m.h.vastenburg@io.tudelft.nl

**Abstract.** This position paper propagates the use of *situated user profiles*, in which system settings and user preferences are linked to situations. Aware systems will change the way people experience their environment. Future homes will be able to sense situations and activities, and react appropriately. User profiles can be used to personalize the system behavior. Situated profiles could help making aware systems more accessible and understandable to the average user. First, an ontology for situations and actions must be developed to serve as a backbone for the aware system and the user-system interaction.

**Keywords:** User profiles, situated awareness, ambient intelligence, mediation.

## 1 Introduction

Aware environments can adapt to for example their inhabitants, activities, tasks, and events (Figure 1). Context-aware systems interpret the environment by clustering low-level stimuli from sensors into abstract situations. These situations trigger appropriate system actions based on stored settings. System actions and user actions change the environment and the cycle is started again.



**Fig. 1.** Context-aware system overview

The behavior of an aware system can be personalized. These personal settings are stored in a user profile. A user profile is a container with attribute-value couples for a specific user. A profiler is a component that creates, maintains and manages user profiles.

### 1.1. Situated Profiles

A situation refers to the current state of the environment. In her book [5], Lucy Suchman describes the situated action theory. She states a dynamic environment is unpredictable in nature; actions cannot be planned in advance. Also, people are not solely working on logical tasks and goals, they are unpredictable emotional beings. Hypothetically, a system that reacts to situations can better handle the dynamic world compared to a system based on logical steps.

Non-situated Profile		Situated Profile		
attribute	value	attribute	situation	value
music-taste	Bach	music-taste	at night, when alone	Bach
	News radio		at breakfast	News radio
...	...	...	...	...

**Fig. 3.** Example of situated and non-situated profiles

Situations can also be used as a framework for a user profile. A situated profile is a user profile where the values are relative to situations, thus the values are only valid in specific situations. Consider for example a situated profile describing the music preferences of a user. The user might like to listen to Bach, but only at night. The preference (Bach) is situated. In preliminary user studies, test subjects were well capable of specifying system behavior using situated preferences.

Next to personal settings, a profiler can also store intelligence on how to update the profile. It would be interesting to store application-specific intelligence as objects inside the profiler. An application can then create “intelligence”-objects, and feed them to the profiler. These intelligence-objects tell the profiler what information to sense and store, and how to reason about the information in the model. The intelligence-objects can also be situated, i.e., the intelligence would only be active in specific situations. A situated profile could help a developer structure the intelligence.

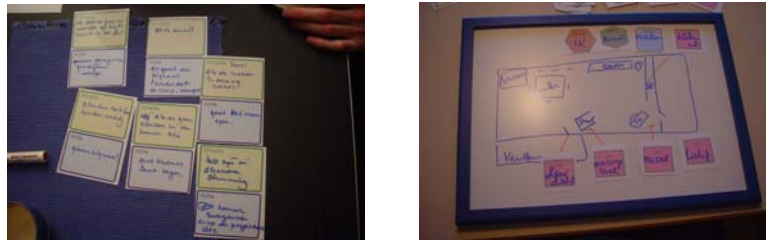
### 1.2. Formal Model for Home Situations

Situations can be used as a framework for a user profile. A formal model for situations is needed. Average users most likely do not want to define entities and relations themselves; they will be using predefined entities and relations most of the

time. We perform user studies to get a better insight in how people process situations. The studies will also tell whether the ‘feeling in control’ increases when users are able to inspect the model and change settings. A formal model will be based on the results of the studies, and the model will then be used in the situated profiles.

Humans and computers do not necessarily communicate at the same abstraction level. A person could say “when I’m working at home”. This informal description of the situation leaves many aspects ambiguous, so it will be hard for an automated system to recognize the situation. Different approaches help make the abstraction-problem manageable: 1) help the user to accurately specify the situation, and 2) make a system that knows how to cope dynamically with ambiguous situations as they occur. An aware system needs to be able to translate abstract descriptions of situations into concrete situations that can easily be detected. The formal model must allow for changing the abstraction level of situations. For many users, an ideal system would allow them to specify situations in a natural, abstract way.

The current focus is on how people build mental models of situations in the home. A qualitative approach is followed, based on the grounded theory [4]. Using the grounded theory approach, data can be systematically gathered and analyzed through the research process, resulting in a theory grounded in data. Data is collected using interviews and puzzle-like assignments (Figure 4), in quick iterations. The research begins with an area of study with no preconceived theory, and allows the theory to emerge from the data.



**Fig. 4.** In the user study, users are asked to write down situated actions (1), and link these actions to a map of their living room (2)

The preliminary results of the study indicate people tend to think in activities (“I am reading a book”), events (“I enter the room”) and needs (“The flowers need water”). If people consider certain views of situations more natural than others, it seems logical to use this insight when designing a formal model for situations. Consequently the interaction concept is also influenced.

#### **1.4. Mediation**

Situations may lead to ambiguity, because of subjective/ambiguous interpretation of situations [3], imperfect sensors [1], and conflicting settings in profiles. These conflicts can not always be solved in advance. For example, a profile might say “Bill likes Bach in the evening”, and also “Bill does not like music when he is reading a book”. There is no problem when Bill does not read books in the evening. On the other hand, when Bill does read a book in the evening, the system does not know what to do. The conflict in the example has no serious implications, but one could think of more serious conflicts.

A profiler might contain intelligence on how to deal with conflicts. It could detect part of the conflicts before they occur. A profiler could be able to compare alternative approaches to solve the specific conflict, and decide which one is best. And it could decide if it is necessary to bother the user with the conflict, or if the conflict can just as well be ignored. Dey [2] proposes to leverage off any useful AI techniques for reducing the ambiguity and involve end users in removing any remaining ambiguity, through a process called mediation.

## **2. Example of a Future Scenario**

The following example illustrates the use of a situated profile in an aware home.

### **Part I: Situated Settings**

It is 2030, Bill lives in an aware house in Seattle, and he arrives home after a day of hard work. A camera next to the front door recognizes his face and his mood, and activates the appropriate personalized home atmosphere settings: dimmed lights and relaxing background music. Bill likes to watch the Jay Leno late night show when he comes home. He instructs the system: “When I enter the living room after work, I want to see the latest Jay Leno show on the TV.” The next day, when Bill enters the room after work, the show is started automatically.

### **Part II: Mediation**

After watching the show, Bill carries the rubbish bag to the street corner, and he locks the house for the night. When he returns to the living room, the situated preference is triggered again; the system recognizes Bill entering the living room after work. The profiler notifies Bill already saw the show, and decides to ask Bill what to do. The television screen shows the message “press PLAY to see Jay Leno again”. Bill does not press PLAY, and the profiler decides to update the situation in the profile.

### **3. Challenges**

In the view of situated profiles, several challenges exist:

1. A formal model for situations and actions needs to be developed, that allows for formalization, abstraction and concretization of situations.
2. A natural user-product interaction mechanism for situations must be developed. Users should not be restricted to communicating directly with a computer system, since the whole environment can be used for interaction.
3. Conflicts are not avoidable in aware systems. A mediation mechanism must be developed that chooses the best strategy when conflicts are detected.

### **Acknowledgements**

The author thanks the ID-StudioLab for their support and David Keyson for supervising the project.

### **Author's Biography**

The author is a Ph.D. student at the Faculty of Industrial Design in Delft. He has a Master's Degree in Computing Science, and has worked for 3 years as a designer and developer of both traditional and intelligent user interfaces in the Aerospace domain. In 2002 he started on a Ph.D. project in order to gain a better theoretical understanding of user profiles and intelligent systems.

### **References**

1. Dey, A., Abowd, G.: Towards a Better Understanding of Context and Context-Awareness (2001)
2. Dey, A., Mankoff, J., Abowd, G., Carter, S.: Distributed Mediation of Ambiguous Context in Aware Environments (2002)
3. Lueg, C.: On context-aware artifacts and socially responsible design (2001)
4. Strauss, A., Corbin, J.: Basics of Qualitative Research - Techniques and Procedures for Developing Grounded Theory, 2nd edition (1998)
5. Suchman, L.A.: Plans and Situated Actions, Cambridge (1987)

# A Service Shaping Approach for Task-based Computing Middleware

Jin Nakazawa    Jun'ichi Yura    Hideyuki Tokuda  
Graduate School of Media and Governance    Faculty of Environmental Information  
Keio University  
5322 Endo, Fujisawa, Kanagawa, JAPAN  
{jin,yurayura,hxt}@ht.sfc.keio.ac.jp

**Abstract**—This paper proposes a new service modeling and lookup method based on service shaping. In existing component-based service frameworks, assembling pervasive services requires application programmers to translate their requirements to the type space defined by service programmers. This translation, however, causes a problem which we call the *hidden service problem*, where an application cannot find a service required for a user task, even though such services exist. To cope with this problem, we propose a shape-based service framework, called *Galaxy*. In *Galaxy*, a service programmer describes the capability of a service, called *shape*, as its meta-level information. Applications query services by specifying partial shape without depending on programmer-defined types. Through the shape-based service management, *Galaxy* achieves an effective utilization of pervasive devices.

## I. INTRODUCTION

In a pervasive computing environment where services are distributed over various devices, user applications need to find services to facilitate users' tasks. Such services are software components running in the devices, and implemented by device vendors, or in some cases by network administrators. Typically, the service finding is conducted by specifying its type and attributes. In addition, a type represents the common structure of a set of service instances, and is defined by a service programmer. An application programmer, then, specifies the types of required services for the application. Therefore, assembling services requires application programmers to translate their requirements manually to the type space defined by service programmers.

This translation, however, disables some of user tasks due to two types of gaps between the user tasks and the service types: the time gap, and the abstraction gap. These gaps cause a problem called *hidden service problem*, where applications cannot find services which satisfies a user's requirement, even if such services exist.

To cope with this problem, we propose a shape-based service framework, called *Galaxy*. In *Galaxy*, service programmers describe capability of services for each service as its external meta-level information in an XML document. We call the capability of services *shape* in a metaphorical sense that each physical object has particular shape to enable its own purpose, such that scissors have two sharp edges to cut papers. To find services, applications specify partial shapes that represent the user requirements. Since the application-defined

shapes can be a part of service-defined ones, applications can find services both developed in the future and of different abstraction levels.

This paper focuses on our vision: the shape-based service management. First, section 2 introduces a scenario and overviews the service model in *Galaxy*. Section 3 organizes issues in the service definition and its finding especially in the type-based service framework. Section 4 proposes the service shaping approach, followed by a prototype implementation. Finally, section 6 summarizes this paper.

## II. SERVICE MODEL IN GALAXY

*Galaxy* is a service framework based on the stream programming aiming at enhancing pervasive computing at home. This section describes the service model in *Galaxy*, and focuses on the shape-based service management.

### A. Scenario

Before getting into the detail, let us introduce a typical application scenario. Our framework is designed with a strong focus on human-to-device interaction, or human-to-human communication through device assists. It also supports a context-aware device selection based on, for example, users/devices location information, and their preferences.

*As well as being a part time researcher, Jin is a househusband and father of a two-year-old daughter. Today, since he must finish writing a research paper, he cannot play with her. Instead, he gives her some toys, and lets her play by herself in the living room. Then, he hits on an idea that he exchanges messages once an hour. In this idea, he creates messages by utilizing his keyboard and mouse. She receives the messages through ambient devices, such as TV, printer, or a speaker set. When she moves to another room, the device to show the message automatically switched according to the device availability.*

### B. Service Model

*Galaxy* defines two notions of services: *primitive services* which control the devices, and *composite services* which coordinate them. A primitive service represents an independent device function. It contains one or more communication endpoints to interact with humans and other services. In the

scenario, a display device to show the daughter Jin's messages should contain at least one end-point for receiving the message text. The end-points are called *Ports* in our framework.

A composite service is a user-side application that integrates multiple primitive services required for the user's task through mappings. The requirement is defined mainly with existence of ports and data types that they can exchange. It also defines their cooperation policy as message paths among them. The application in the above scenario specifies two services: one with a physical text input and a network text output (*requirement A*), and the other with a network text input and a physical output (*requirement B*).

### C. Mapping

When a user invokes a composite service, a mapping occurs. Galaxy conducts the mapping through the following two steps. First, it searches for a service by confronting its port with a requirement in the composite service. When there are multiple services that conform to the requirement, they are sorted by the user's preferences to select the most suitable one. The preferences, which are defined in the requirement or in an external database, include location, quality of service, and so on. If the sorted result still contains multiple candidates, Galaxy prompts the user for manual selection. In the scenario, a text input service with a GUI running on his PC, and a device service (e.g. one in a TV, a printer, etc.) would be mapped to the requirement A and B, respectively. Second, it establishes communication paths based on the cooperation policy. When the source and destination ports declare different data types, mediator services relay the data translating its type. A mapping is context-aware. Particularly, it is updated according to the location of users and devices to support their migration. In the scenario, the daughter's movement triggers the mapping for the message actuation device to be updated.

## III. ISSUES

In most cases, services and applications are developed by different programmers at different moments. This causes the following two problems, due that services in a pervasive computing environment are accessed by unlimited applications.

### A. Time Gap

In the existing component-based service frameworks for pervasive computing[2][1][3], programmers define a type for each service. A type represents the role of a service based on a preexisting agreement between the services' and an application's programmers on their meaning. The type-based service management does not allow type names to change or mean other roles to keep the type space stable.

We see, recently, many kinds of new devices have been and will be produced. These new devices will be wrapped by new services whose types are unknown currently. In the scenario case, the requirements would be fulfilled even with a type-based framework on either of the following two assumptions. Now, let us take *requirement A* for instance. First, programmers firmly comply with a common type, e.g.

*TextInput*. For example, a newly developed service with voice input can substitute the normal GUI input version if its type is *TextInput*. However, it disables another application to utilize the voice input service explicitly. Second, the service programmer determines a new type, e.g. *VoiceInput*, and application programmers modify the requirement. This, in turn, enables the explicit utilization, however, forces users to update the application frequently.

### B. Abstraction Gap

Users' requirements are specified in various levels of abstraction. Sometimes they are highly abstract. In the scenario case, Jin's daughter does not concern how to acquire the message from her father. There might be stricter requirements as follows: print the message, print it with black and white, print it with 600dpi full-color, etc.. On the other hand, the abstraction of a service is fixed in the type-based framework. The service programmer defines its structure, and attaches it the name of type.

This gap causes hidden services. The application in the scenario can accomplish its message-delivery task if there is a display in the environment. If there is none, a printer, a TV, or a facsimile etc. can be a substitute for the display. However, the type-based framework requires the application programmer to specify all types of the possible substitutes to utilize devices effectively in the user's environment. This is nearly impossible due to the time gap.

### C. Requirement

These problems originate with encapsulation in component-based systems. Basically, encapsulation helps application programmers access services without concerning details of their functions. However, the abstraction gap indicates that, in some cases, application programmers concern only a part of the encapsulated details, and in the other cases, they might have more specific requirements than the encapsulation represents. In addition, the time gap indicates the encapsulation disables applications access future-defined services. Therefore, for enabling an application flexible access to pervasive services, component-based service frameworks should define their capabilities explicitly, in addition to the implicit definition through the encapsulation. This originates the following requirements.

#### 1) Representing physical interaction capabilities

Services should be able to advertise their capabilities on interaction with humans through the physical world, which includes data type they actuate, actuation formats, detailed quality of the actuation, etc.. This helps application programmers to specify users' requirements directly in various granularities.

#### 2) Representing network interaction capabilities

They also should be able to advertise their capabilities on interaction with services through a network. The capabilities include, for example, the type, protocol, and quality of data they exchange. This also helps the requirement specification in various granularities.

```

DataType ::= Perception "/" Media
Perception ::= "aural" | "visual" | "tactile" |
               "smell" | "taste" | "*"
Media ::= "voice" | "paper" | "display" |
          "braille" | ... | "*"

```

Fig. 1. Definition of Real-World Data Type

#### IV. APPROACH: SERVICE SHAPING

To cope with the aforementioned issues, we propose the notion of service shape. A shape of service is defined by its external interface, internal operation, and their dependencies. The external interface represents how the service looks like from the users' and other services' viewpoints, while the internal operation represents how the service works. The dependency expresses the operation conducted when an external interface is invoked. Through the set of these three entities, we are aiming at representing services' capabilities.

##### A. External Interface

The external interface of a service defines how the service interacts towards both the real and the virtual worlds. We define the end-point of the interaction as *Port*, and there are possibly { real world , virtual world } { input , output } ports.

A real world input port receives data from the physical environment, such as temperature, humidity, audio, and video, while a real world output port exports data there such that a printer exports documents by printing them on papers. A virtual world port abstracts data flow between a service and the network. For example, a thermometer service sends the temperature information to the network through a virtual world output port, and a video viewer service acquires a video stream from the network through a virtual world input port.

Each port specifies its own details by the acceptable data type, and the protocol for data transmission. In case of virtual world ports, acceptable data types are defined with the MIME-types[4], [5] and protocols by their names. Protocol stacks are instantiated based on the specification in a pluggable manner[6], though we do not look into its details in this paper.

There is, however, no de-facto standard that defines types for real-world data. One of the early investigations for categorizing such data is Media Queries[7]. It aims at enabling web browsing adaptive to devices that users utilize, and categorizes such devices into the following eight: aural, braille, handheld, print, projection, screen, tty, and tv. This categorization, however, mixes up perceptions and media types, such that "tv" is a device type while "aural" is a perception. Instead, we define the real-world data type by layering the perception and the media as seen in MIME-types. Figure 1 shows our experimental categorization. A data type is composed by the *perception* which means how a human perceives the data, and the *media* which represents a physical object which carries the data. For example, a printer service outputs a document through "visual/paper" and a microphone service inputs a voice through "aural/voice."

##### B. Internal Operation

The internal operations are defined with the following two: dependency between ports and a control, and data flows between input and output ports. Suppose a power controller service that turns on/off a TV when it receives an event through a virtual world input port. If a user's task includes turning on a TV, the corresponding application should find the power controller service. This means that a service shape should define the operation originated by a data input. Some of the operations are data forwarding between input and output ports. A simple example is a printer service that receives a postscript document from a virtual world input port and forwards it to a real world output port. Therefore, the shape should also define such a static data flow between ports.

The dependency between a service operation and a data input is defined as a state graph and mappings between each state and ports. A state represents a particular condition of the device, which the service operates. In case of the power controller service, it contains two states: *on* and *off*. Another example is a VCR controller service which contains *playing*, *recording*, *forwarding*, *rewinding*, *stopped*, and *paused*. The state graph defines a possible transition among states as a finite-state automaton  $M$  as follows:

$$M = (Q, \sum, T, s, F) \quad (1)$$

$Q$  is a set of states that includes programmer-defined states.  $s \in Q$  is the programmer-defined initial state, while  $F \in Q$  is the final state which the framework defines.  $\sum$  and  $T$  are, respectively, sets of input and transition functions. The transition function is defined by attaching a virtual world input port to its destination state. Suppose creating a transition from *on* to *off* in the power controller service above. In this case, an input port is attached to the state *off*. When the port receives some data, it inputs the destination state name to the automaton. Therefore,  $\sum$  is equivalent to the set of state names. The specific characteristic of the automaton is that a virtual world output port is attached to each state. When a state transition has occurred, an event object is dispatched through the output port attached to the destination state. This enables applications control devices sequentially by connecting the virtual world input and output ports.

Another part of internal operation is static flows between input and output ports in a service. In case of the printer service, there is a static flow between the virtual-world input port (ps input) and the real-world output port (paper output). Defining flows enables composite services to know what a service conducts, since each end of flows defines its acceptable data type as described in the former section. In the example above, the input and output ports declare to deal with "text/ps" and "visual/paper," respectively. Also, a service with ports which declare "text/plain" and "aural/voice" will be a voice synthesizer, and another with "aural/voice" and "text/plain" will be a voice dictator. Applications can know capabilities of services from these flows without assuming any programmer-defined service types.

## V. IMPLEMENTATION

### A. Definition of Shape

We enabled symmetric description for definition and lookup of service shape. There is no abstraction gap between the definition and these queries, since the queries can include the requirements for external interface and internal operation directly. There is also no time gap, since queries can be configured without depending on programmer-defined types. Consequently, adopting the same syntax for both definition and lookup of service solves the hidden service problem.

In the initial (and current) implementation, we adopted XML as the description language of shape, and Java as an implementation language of service. Galaxy is benefited much from its transformability. One of the transformation tools is XSLT[8] with which we provided service programmers with *Shape Compiler*. It transforms the XML document containing shape into a skeleton of service implementation into which the programmer appends a few code to control the actual device. The transformability also allows Galaxy to be operated on different directory services by translating the shape description into the original languages of concrete directory services, such as the name-specifier in Intentional Naming System[9] and XPath[10] in Apache Xindice[11] XML database.

Figure 2 is an XML document, which defines the shape of plasma display service. In the document, `<body>` tag represents the shape of this service. Among its child tags, `<graph>` and `<map>` in lines 5 – 12 and 25 – 35 defines the internal operation. The former defines the automaton that contains two states and one bidirectional state transition in this case. The latter configures the static flows (lines 26 – 28), and mappings between the states and ports (lines 29–34). `<port>` tag defines external interface, which includes two real-world ports and four virtual-world ports.

Composite service is defined with an extended set of the tags for defining primitive services. Its definition is different from the primitive service's in that it does not require implementation, since Galaxy runtime automatically lookup required services and bind them based on the definition. The required services are specified in the additional tag `<composite>` as shown in Figure 3. Among its child tags, `<services>` includes queries for required service: `sender` and `receiver` in this example. `<connection>` tag defines cooperation policy among these services.

### B. Service Finding by Shape

Queries inherit the grammar of service shape definition, meaning that developers of composite services build queries by XML. The exception is that queries are allowed to be a subset of the complete service shape definition. For example, the query for `sender` in Figure 3 denotes a service which outputs an MPEG video stream to the virtual world. The other queries for `receiver`, then, denotes a service which inputs an MPEG video stream from the virtual world, and displays it.

Directory service in Galaxy is implemented by a native XML database called Xindice Database[11] developed by the

Apache XML project[12]. It stores shape of services as XML documents, and accepts partial shape as queries that are also described in XML as described above. Adopting the common format for defining both shape and query enabled this simple implementation.

## VI. SUMMARY

This paper proposed a shape-based service framework, called Galaxy. In Galaxy service framework, a service programmer describes the capability of a service, as a *shape* in an XML document. Applications can query services with various granularities by specifying partial shape. This provides pervasive applications with flexible service lookup. Since Galaxy adopted XML for both service shaping and lookup, its directory service can be realized by an off-the-shelf XML database. This helps service programmers and users deploy our framework with limited costs. We currently believe that it can complement type-based service management as a common naming infrastructure among different middleware systems. As future work, we are extending this work as follows. First, since the current prototype deals with only the data type that services exchange, it should be extended to incorporate quality of service including those of data and an actuation device. We are also aiming at improving lookup performance. In the current configuration where the directory service is centralized, it scales as many as one hundred services on the assumption that a query should return results in ten seconds. We are extending the directory service to have self-organization capability to distribute the queries and to increase fault-tolerance.

## REFERENCES

- [1] Michael H. Coen, Benton Phillips, Nimrod Warshawsky, Luke Weisman, Stephen Peters, and Peter Finin, "Meeting the computational needs of intelligent environments; the metaglug system," in *International Workshop on Managing Interactions in Smart Environments (MANSE'99)*, Dec. 1999, pp. 201–212.
- [2] Manuel Roman, Christopher Hess, Renato Cerqueira, Anand Renganat, Roy H. Campbell, and Klara Nahrstedt, "Gaia: A middleware infrastructure to enable active spaces," in *IEEE Pervasive Computing*, Dec. 2002, pp. 74–83.
- [3] Sun Microsystems, Inc., "Jini Architecture Specification," Nov. 1998, <http://www.javasoft.com/products/jini/specs/jini-spec.pdf>.
- [4] N. Borenstein and N. Freed, "Mime (multipurpose internet mail extensions) part one: Mechanisms for specifying and describing the format of internet message bodies," Sept. 1993, RFC1521.
- [5] K. Moore, "Mime (multipurpose internet mail extensions) part two: Message header extensions for non-ascii text," Sept. 1993, RFC1522.
- [6] Jin Nakazawa and Hideyuki Tokuda, "A pluggable service-to-service communication mechanism for home multimedia networks," in *ACM Multimedia 2002*, Dec. 2002, pp. 621–630.
- [7] Hakon Wium Lie, Tantek Celik, and Daniel Glazman, "Media queries," July 2002, W3C Candidate Recommendation.
- [8] James Clark, "Xsl transformations (xslt) version 1.0," in <http://www.w3.org/TR/xslt/>, Nov. 1999.
- [9] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley, "The design and implementation of an intentional naming system," in *Symposium on Operating Systems Principles*, 1999, pp. 186–201.
- [10] James Clark and Steve DeRose, "Xml path language (xpath) version 1.0," in <http://www.w3.org/TR/xpath/>, Nov. 1999.
- [11] "Apache xindice," <http://xml.apache.org/xindice/>.
- [12] "The apache xml project," <http://xml.apache.org/>.

```

1 <?xml version="1.0" ?>
2 <!DOCTYPE service>
3 <service>
4 <head>(attributes such as name, author, url are defined here.)</head>
5 <body>
6 <graph> <!-- automaton -->
7 <state ref="s1"><name>OFF</name><comment>turned off</comment></state>
8 <state ref="s2"><name>ON</name><comment>turned on</comment></state>
9 <transition>
10 <direction>bi</direction><source ref="s1"/><destination ref="s2"/>
11 </transition>
12 </graph>
13 <port> <!-- external interfaces -->
14 <!-- real world -->
15 <input ref="p1">
16 <name>in</name><direction>real</direction><type>video/mpeg</type>
17 </input>
18 <output ref="p2">
19 <name>out</name><direction>real</direction><type>visual/display</type>
20 </output>
21 <!-- virtual world -->
22 <input ref="p3">
23 <name>turn_off</name><direction>virtual</direction><type>event/*</type>
24 </input>
25 <input ref="p4">
26 <name>turn_on</name><direction>virtual</direction><type>event/*</type>
27 </input>
28 <output ref="p5">
29 <name>turned_off</name><direction>virtual</direction><type>event/tv_off</type>
30 </output>
31 <output ref="p6">
32 <name>turned_on</name><direction>virtual</direction><type>event/tv_on</type>
33 </output>
34 </port>
35 <map> <!-- other internal operations -->
36 <entry> <!-- static flow of video stream -->
37 <input ref="p1"/><output ref="p2"/>
38 </entry>
39 <entry> <!-- attach ports to OFF state -->
40 <input name="p3"/><state ref="s1"/><output name="turned_off"/>
41 </entry>
42 <entry> <!-- attach ports to ON state -->
43 <input ref="p4"/><state ref="s1"/><output name="turned_on"/>
44 </entry>
45 </map>
46 </body>
47 </service>

```

Fig. 2. Shape Definition for Plasma Display Service

```

1 <composition>
2 <services> <!-- queries by partial shape -->
3 <service name="sender"><body>
4 <port>
5 <output><type>video/mpeg</type></output>
6 </port>
7 </body></service>
8 <service name="receiver"><body>
9 <port>
10 <input><type>video/mpeg</type></input>
11 <output><direction>real</direction><type>visible/display</type></output>
12 </port>
13 </body></service>
14 </services>
15
16 <connection> <!-- communication-paths among services above --> </connection>
17 </composition>

```

Fig. 3. Definition of Composition

# Getting things done: Towards a conceptual framework to support the management of multiple activities

Victor M. González and Gloria Mark  
Department of Informatics  
School of Information and Computer Science  
University of California at Irvine  
 [{vmgyg, gmark}@ics.uci.edu](mailto:{vmgyg, gmark}@ics.uci.edu)

## Abstract

This paper presents and discusses a set of initial elements that conform a conceptual framework explaining how information workers manage multiple activities. In spite that previous studies have found that involvement in multiple and varied activities is a central part of the practices of information workers, there is not a clear understanding of how individuals manage them and how information technology can support it in a comprehensive way. Taking an activity-centered perspective, the elements of the conceptual framework emphasize the ways that individuals delineate their activities, the evolution of those activities, the effects of urgency, and the interplay between the intentions of individuals and the situations they face to enact activities. Derived from workplace observations of information workers, we provide illustrations of how activity management is currently supported by ensembles of automated and non-automated artifacts and highlight the relevance of the elements of the conceptual framework to understand those strategies.

## 1. Introduction

The management of multiple activities is an often mentioned but a barely understood aspect of the daily lives of many information workers. Studies from many different disciplines, ranging from psychology, CSCW, and organizational research have described information work as requiring the involvement in multiple projects, teams or initiatives, which results in individuals having constant interactions with co-workers, and demands the integration of multiple information resources to get each activity done [2, 6, 10, 13, 17, 19, 22]. However, what it is still very unclear is the understanding about what it really takes for individuals to manage those multiple activities and how it can be supported by information technology in a comprehensive way. Most current designs of information technology are oriented towards supporting distinct *tasks* such as editing of graphics, word processing, quantitative analysis, message interchange or information retrieval. Technology is not organized in terms of larger themes that are associated with separate projects or activities. It is left up to people to integrate their information into cohesive structures related with each activity and to introduce ad-hoc strategies to manage all of them.

From previous studies of information work we can understand that the management of multiple activities is a very complex phenomena, which requires individuals to handle time, documents, contacts, and communication threads [1, 3, 4, 7, 11, 12, 14, 23]. Because most studies have been directed towards particular areas,

we still not have a clear picture that let us to understand all the elements together and the way they interrelate with each other.

We argue that in order to create technology that really meets the needs of information workers with respect to the management of their activities, we have to take a step back and reconsider the way activity management is enacted in practice. This effort requires a holistic view to understand all the different elements involved in it (time, contacts, documents, communications, etc.), the contributions of both the automated and non-automated artifacts used by people, the relationship between subjective and collective perspectives with respect to the activities and, even more important, the way that activity management strategies are actually enacted under changing circumstances. In this paper, we propose a set of initial elements, which are part of a conceptual framework to consolidate these understandings about activity management. These elements include (1) the delineation of activities, (2) their evolution processes, (3) the relationship between individual and collective perspectives, (4) handling of urgent vs. normal activities, (5) the evaluation processes to reassume work after interruptions. As a way to illustrate the implications of some of the elements of the framework, we present a scenario of activity management taken from one of our workplace studies, which describes the current way people support this kind of work.

## **2. From Intentions to Situations: situated activity management**

Having multiple activities require people to constantly interleave their attention among them. Studies of information workers have identified different kinds of strategies used to interleave activities [5, 9, 18]. They include (1) time-shared execution, where people spend certain time on an activity and then switch to the next activity in line; (2) prioritizing, where the order of execution of activities is established beforehand; or (3) linearization, which occurs when resource constraints make people organize their activities into a single linear stream, so as to use resources efficiently.

It has been found, however, that these strategies are executed as long as the person is not forced to stop their use [9]. We might say therefore that those strategies reflect the *ideal* scenarios of activity management rather than the *actual* ways in which activities are performed in practice. When circumstances around people change, the execution of the strategy is suspended and the individual has to adjust, perhaps opting for another strategy. This potentially creates a situation where people end up mixing their day with periods of prioritization, periods of linearization and periods of time-sharing [9]. All this indicates that activity management is situated in nature, and as a situated phenomena, we face the impossibility of describing it as a set of pre-planned actions blindly executed by individuals [21]. However, to understand activity management, we also have to consider that individuals have intentions and motives to get things done at work. They have projects to attend, people to talk to, deadlines to meet, etc. Assuming that people are just guided by circumstances is to negate the intentionality of their actions. Therefore, we believe that it is by understanding the *interplay between intentions and situations* that we can fully capture the mechanisms that people use to handle multiple activities. How this interplay occurs and how the different aspects related with the enactment of an

activity affect this interplay, is described with the elements of the conceptual framework presented in the next section.

### **3. Elements of a conceptual framework: working sphere approach**

The workplace studies that we have conducted allows us to construct elements that are fundamental to understanding how information workers manage multiple activities. In these studies, we observed the practices of information workers including software developers, financial analysts, and managers<sup>1</sup>. Guided with the notions of activity theory we placed *activity* as the unit of analysis from where we derived an understanding of the behavior of our informants [15, 16]. With this perspective in hand, the analysis of our data indicates that the following elements will be central components of a conceptual framework describing the management of multiple activities:

#### **Definition and delineation of activities**

While we recognize that an activity is in essence a motivated-collective effort, where the actions of the individuals contribute partially to get the activity done [16], our data indicates that when our informants talk about their work, they describe it in terms of the activities rather than the actions they are involved with. This finding led us towards trying to understand how these activities were defined from their perspective. We derived from our data the grounded concept of *working sphere*, which serves to describe an activity from the informant's perspective. A working sphere is defined as a set of interrelated tasks, which share a common motive, involves the communication or interaction with a particular constellation of people, uses ensembles of resources and has its own individual time framework. It is the whole web of motives, people, resources, and time frameworks that distinguishes a working sphere from others. This element points out that information technology should encompass support for the way that individuals define their activities and the resources associated with them.

#### **Evolution of activities**

A fundamental factor to be considered is how an activity (a working sphere), gets created, how it evolves and how their resources are defined, and how it changes from representing a current central concern to an irrelevant one. Our observations confirm that activities rarely get specified up front and people become aware of the implications of an activity as they gather more information about it. Consequently, individuals often handle working spheres with different levels of maturity with respect to their definition of the constellations of people, the resources and time frameworks. Therefore, the individual does not have the same kind of information for all his working spheres. The lack of definition of a working sphere often results in problems whenever an information system demands many details in order to support activity management, such as specific deadlines, or specific people involved on it. Consequently, this element points out that different kinds of technology support are required depending on the degree of definition of the working sphere.

#### **Relationship between individual and collective perspectives**

Another aspect to consider is the impact of company project management styles on the way that individuals manage their working spheres. In particular, we need to define

---

<sup>1</sup> More details of the study and results can be found in: González, V. and G. Mark. "Constant, Constant, Multi-tasking Crazy": Managing Multiple Working Spheres. in Proceedings of CHI 2004. 2004. Vienna, Austria: ACM Press.

how individuals transform and define their own individual working spheres from company-level project goals. We have observed that what might be a single issue from a manager's perspective, becomes a set of working spheres once it is assigned to an individual. It is the individual who demarcates the resources required, the constellation of people involved and the time framework to achieve his working spheres. This element points out that activity management has to be approached both from a high level managerial perspective and from the individual perspective, and information technology should provide enough flexibility to allow that both perspectives to coexist.

#### **Handling of urgent vs. normal activities**

There is a difference between the strategies used by people to handle working spheres when there is a need to attend to it immediately than when it can be attended without urgency. We have observed that in such cases the support of tools might be peripheral given that individuals might not have time to engage in the extra effort of turning into a tool and enter information in it. We believe that this represents an opportunity to understand how urgency affects the interleaving of activities and the specific strategies used to resume those activities that have to be postponed due to interruptions. This element points out to the technological support in two main ways. On the one hand, we have seen that while handling urgent activities people often interact with individuals who are not part of the group of co-workers with who they normally cooperate. In those situations they turn the might turn to external people such as providers, clients, or consultants. Systems that let people to locate expertise in a prompt way would be useful. In the other hand, given that people often have to provide accounts of what was done to solve urgent problems and they do not have time to do it while solving them, systems that records the activities performed will facilitate the reconstruction of the actions taken.

#### **Evaluation process to reassume work**

Given the situated nature of their work, individuals often re-evaluate their circumstances and decide what activity in the short term is more relevant to be executed after an interruption. We have noticed that individuals have to enact this evaluation process whenever they have to reassume work, for example, after a long break or at the beginning of the day. However to some extent the same evaluation is done when work is reassumed after shorter interruptions. Close attention should be placed over the characteristics of the personal information environment that make activity cues more prominent and visible.

Looking at activities as working spheres let us understand from the individual's perspective and the *state* of the activity at a certain point of time. Inspired by the Interactionist Theory of Action, proposed by Strauss, we believe that considering each activity as a phenomena, the evolutionary process of each activity can be described as a *trajectory*: "as a course of action which embraces the interaction of multiple actors and the contingencies that may be unanticipated and not entirely manageable" [20:54]. Taking this approach, we can understand activity management in the long term as the handling of different working sphere trajectories, each with its own time frame, each with each own constellations of people and resources, and each evolving as the individual discover its meaning and the ways to get it done. Looking at the evolution of activities as trajectories helps us to understand that an activity remains alive or "on hold" even if the individual does not perform an action related with it. Progress will not be made on the activity until the individual resumes it. This let us understand that the individuals can hold a number of independent activities regardless of whether the individual will take care of them or not. The challenge for information

workers is then to keep constantly aware of those working sphere trajectories that they are handling. Given that not all the interactions with people and resources guarantee that those pending working spheres will be brought to their attention at the right time, the role of information technology becomes one of providing enough cues for those pending working sphere trajectories.

#### 4. Understanding activity management “in situ”

Analysis of our data has resulted in the identification of strategies for the management of working spheres. We found that our informants typically handle an average of five (5) working spheres per day for which they are responsible. The following comments from one of the financial analysts about how he manages his activities point out the rich collection of resources required for it:

*“ I would, you know, jot down notes, of what's need to be done, like sometimes I will have a piece of paper just for writing notes on, aside from my main list [email printout] which is like my tasks, my major tasks that need to be done. So in addition to that I will write like a to-do list per day usually [paper notepad]... and then I just go down there [MS Excel spreadsheet with items to be released by his team] to check things out and follow that for that day, or added back on the other list [MS Excel spreadsheet] if that list does not contain an item... I think I just prioritize, you know, based on this list [email printout], my current like, things, the set of meeting notices [MS Outlook calendar], and reminders to do certain things [MS Outlook reminder]... I also have my goals [MS Word document printout] up on my cube's wall now and I put them up there and I got like two goals that I have to have finished by the end of June, and I am going to get them done, absolutely... ”*

From the comments of this analyst, we can understand that activity management involves different kinds of resources: notepads, email printouts, spreadsheets, MS Outlook calendars and printouts hanging on walls. The scenario depicted by the analyst is one where many artifacts support the management of activities at different levels, where he has found for what level each artifact is more appropriate (awkward), and where a clear effort has been made to integrate all those artifacts into a functional *ensemble*. Overall, the analyst's experience allows us to see that the definition of the things to be done in a certain day involves a complex process which consolidates information from different sources and gradually let the individual focus on those working spheres which will become his current concerns for the day.

While it is clear that no single artifact support activity management in the long and medium term, when people experience fast-pace conditions and have to interleave quickly across different working spheres, we identified some artifacts that play a stronger role helping people to prioritize and maintain their attention over their working spheres. Given that switching among working sphere is constant (in average every twelve (12) minutes [8]), those artifacts act as container holding certain information about working spheres. They both *signal* the working sphere to be attended to and *describe* with some detail what has to be remembered. The artifact is often updated across the day with results when work within a sphere has to be postponed.

The artifact takes on different forms, each providing different properties in terms of the signaling or descriptive power. A common implementation uses a special inbox folder in the email client to contain messages related to working spheres to be attended to. Messages are from others or from the individual himself with summaries

of other messages or other documents. A second form of the artifact used by three of our informants is printouts of email messages. People keep piles on their desks with printouts of email messages or meeting notices generated by their team's scheduling system. On those printouts are annotations such as clarification notes or contact information. A third form of artifact relies on more traditional activity management tools. Two of our informants used planners extensively to manage their working spheres usually by integrating conversations or email messages into the planner. For each day, they listed the working spheres to cover and if necessary they transferred pending actions. A fourth form of artifact is the post-it note. One informant placed up to seven notes with references to actions to be taken in different working spheres. The notes stay on the monitor shelf for as long as the working sphere lasts and are annotated to indicate the status of the working sphere. For instance, while working on a software update for one of his colleagues, an informant wrote down comments regarding progress in that working sphere (e.g. "Test PB patch DAN's PC –Waiting for AL").

Both long and short term strategies involving the use of artifacts for activity management indicate the importance of understanding the way that individuals define their activities. The concept of working sphere helps us understand, differentiate, and delineate activities by looking at the different resources, persons or time frames involved in each. Using the concept of working sphere, we can perceive how the individual with the support of his artifacts is able to make sense of his circumstances, keep track of the different working sphere trajectories, and gradually advance to achieve the motivation behind each one.

## 5. Conclusions

Inspired by the notions brought by activity theory's scholars we argue that the successful definition of a conceptual framework will come from recognizing and placing *activity* as the unit of analysis under which human behavior is explained and objects, such as artifacts, are brought to real meaningfulness [15]. However we take a step further and argue that in order to understand the *activity* involved in *activity management* we have to bring into close inspection how individuals delineate and define their own activities. We argue that it is by understanding the processes behind how activities emerge, evolve, and mature to the point that are concluded or abandoned that we can understand the role of information resources required for their management.

## 6. References

1. Blandford, A.E. and T.R.G. Green, Group and individual time management tools: what you get is not what you need. *Personal and Ubiquitous Computing*, 2001. **5**(4): p. 213-230.
2. Bluedorn, A.C., C.F. Kaufman, and P.M. Lane, How many things do you like to do at once? An introduction to monochronic and polychronic time. *Academy of Management Executive*, 1992. **6**(4): p. 17-26.
3. Bonnie, N.A., S. Whittaker, E. Isaacs, M. Creech, J. Johnson, and J. Hainsworth, Integrating communication and information through ContactMap. *Communications of the ACM*, 2002. **45**(4): p. 89-95.
4. Cole, I. Human Aspects of Office Filing: Implications for the Electronic Office. in *Proceedings of the Human Factors Society - 26th Annual Meeting*. 1982.
5. Cypher, A., *The Structure of Users' Activities*, in *User Centered System Design*, D.A. Norman and S.W. Draper, Editors. 1986, Lawrence Erlbaum Assoc., Publishers: Hillsdale, New Jersey. p. 243-263.

6. Czerwinski, M., E. Horvitz, and S. Wilhite. A Dairy Study of Task Switching and Interruptions. in *Proceedings of CHI 2004*. 2004. Vienna, Austria: ACM Press.
7. Ducheneaut, N. and V. Bellotti. E-mail as Habitat: An Exploration of Embedded Personal Information Management. *Interactions*, 2001. **8**(5): p. 30-38.
8. González, V. and G. Mark. "Constant, Constant, Multi-tasking Crazyiness": Managing Multiple Working Spheres. in *Proceedings of CHI 2004*. 2004. Vienna, Austria: ACM Press.
9. Gruen, D.M., The Role of External Resources in the Management of Multiple Activities, in *Department of Cognitive Science*. 1996, University of California, San Diego: San Diego. p. 298.
10. Hudson, J.M., J. Christensen, W.A. Kellogg, and T. Erickson. "Td Be Overwhelmed, But It's Just One More Thing to Do:" Availability and Interruption in Research Management. in *Proceedings of CHI 2002*. 2002. Minneapolis, Minnesota. 97-104.
11. Kirsh, D., The Context of Work. *Human Computer Interaction*, 2001. **16**(2): p. 306-322.
12. Kraut, R.E., C. Egidio, and J. Galegher, *Patterns of Contact and Communication in Scientific Research Collaborations.*, in *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, J. Galegher, R.E. Kraut, and C. Egidio, Editors. 1990, Lawrence Erlbaum. p. 149-172.
13. Kraut, R.E., R. Fish, R. Root, and B. Chalfonte, *Informal communication in organizations: form, function, and technology*, in *Groupware and Computer-Supported Co-operative Work*, R. Baecker, Editor. 1993, Morgan Kaufmann, 1993. p. 287-314.
14. Landsale, M., Remembering about documents: memory for appearance, format and location. *Ergonomics*, 1991. **34**(8): p. 1161-1178.
15. Leontev, A., *Activity, Consciousness, and Personality*. 1978, Englewood Cliffs, N.J.: Prentice-Hall, Inc.
16. Leontev, A., *The Problem of Activity in Psychology*, in *The Concept of Activity in Soviet Psychology*, J. Wertsch, Editor. 1979, M.E. Sharpe, Inc.: Armonk, New York. p. 441.
17. Mintzberg, H., *The Nature of Managerial Work*. 1973, Englewood Cliffs N.J.: Prentice Hall.
18. Miyata, Y. and D.A. Norman, *Psychological Issues in Support of Multiple Activities*, in *User Centered System Design*, D.A. Norman and S.W. Draper, Editors. 1986, Lawrence Erlbaum Associates, Publishers: Hillsdale, New Jersey. p. 265-284.
19. Sproull, L.S., The Nature of Managerial Attention. *Advances in Information Processing in Organizations*, 1984. **1**: p. 9-27.
20. Strauss, A., *Continual Permutations of Action*. 1993, New York: Aldine De Gruyter.
21. Suchman, L., *Plans and Situated Actions: The problem of human machine communication*. 1987, Cambridge: Cambridge University Press. 203.
22. Suchman, L. and E. Wynn, Procedures and Problems in the Office. *Office: Technology and People*, 1984. **2**: p. 133-154.
23. Whittaker, S., Q. Jones, and L. Terveen. Contact Management: Identifying Contacts to Support Long-Term Communication. in *Proceedings of CSCS 2002*. 2002. New Orleans, Louisiana: ACM.

## About the authors

Victor M. González is a Ph.D. student at the University of California, Irvine. He is a member of the Interactive and Collaborative Technologies group, working with Dr. Gloria Mark. He has done research on Distributed Collaboration, Information Visualization Systems and Home Technologies using both qualitative and quantitative methods. Currently he conducts research on activity management within technology-rich environments. He received a M.S. in Information and Computer Science (2002) from UC Irvine, and a M.S. in Telecommunication and Information Systems (1997) from The University of Essex, UK.

Dr. Gloria Mark is a faculty member in the Interactive and Collaborative Technologies group at the University of California, Irvine. Prior to this, she had been a research scientist at the GMD in Bonn, Germany. Her work in CSCW involves studying the effects of technology use on group behaviors. Technologies she has studied include collaborative hypermedia, shared electronic workspaces, desktop conferencing, and HDTV video conferencing. Her main research interest is in studying the effects of virtual collocation.

# User-centered support to localized activities in ubiquitous computing environments

Helder Pinto and Rui José

University of Minho, Information Systems Department  
Campus de Azurém, 4800-058 Guimarães, Portugal  
{helder,rui}@dsi.uminho.pt

**Abstract.** The design of pervasive and ubiquitous computing systems must be centered on users' activity in order to bring computing systems closer to people. The adoption of an activity-centered approach to the design of pervasive and ubiquitous computing systems should consider: a) how humans naturally accomplish an activity; and b) how computing artifacts from both the local and personal domains should contribute to the accomplishment of an activity. This work particularly focuses on localized activities performed by occasional visitors, i.e., activities having a strong association with a specific physical environment, which may be visited by people who are not accustomed to it. We are investigating how ubiquitous computing environments can provide user-centered support to localized activities, by exploring activity specification models and mechanisms allowing for the integration between local and personal environments.

## 1 Introduction

Ubiquitous computing environments promise to transparently support people in their daily activities by leveraging computing resources existent in the physical environment. However, two major challenges have not yet been solved: firstly, mobile users still do not have much support in the accomplishment of activities that go beyond the virtual workspace, particularly activities that are strongly related with the physical environment in which they are performed; and, secondly, the application- and document-centered paradigms remain the central interaction models, despite their strong limitations for mobility scenarios. In order to bring computing closer to people, the design of ubiquitous and pervasive computing systems must be centered on the user, particularly on the user activity [1, 2]. Designing ubiquitous and pervasive computing systems so that user activity is handled as a first class object enables those systems to transparent and effortlessly meet user expectations, relieving users from the current burden of dealing with computer applications. The adoption of an activity-centered approach to the design of pervasive and ubiquitous computing systems should consider: a) how humans naturally accomplish an activity; and b) how computing resources from both the environmental and personal domains should contribute to the accomplishment of an activity.

Activity Theory [3] sets the ground for the psychological aspect of activity-centered computing by defining a theoretical framework for analyzing activities. We, as designers and developers of ubiquitous and pervasive computing systems, must understand how people perform their activities: which actions may compose an activity and what is their usual sequence; which operations and tools are required for the conditions in which an activity may unfold; how to design tools so that users are not distracted; how does context influence actions, operations, and required tools; and how to support the evolution of an activity and the different ways it may be performed by different people.

The following section details the association between computer-supported human activity and its physical environment, describing the concept of localized activity and the role of the integration between resources from both local and personal domains. Section 3 presents some of the previous research that relates with our current work. Section 4 is a simple scenario illustrating our vision and finally Sect. 5 describes the approach we are taking for accomplishing the support to localized activities performed by occasional users.

## 2 Computer-supported activity and its environment

Computer-supported activities and the physical environments in which they take place have different levels of association. Some activities can be performed everywhere, as long as there appropriate resources, and their relevance is not associated to the physical environment (e.g., managing e-mail or editing a report). The user may possibly need to explore some local resources (e.g., a display, a keyboard, connectivity, etc.), but these are not specific to the activity and can also be found elsewhere. On the contrary, other activities are closely related with specific physical environments. In this work, we are addressing *localized activities*, which we see as those activities that have a strong association with the physical environment, i.e., activities that can only be accomplished in specific places (e.g., visiting a relative at the hospital or visiting an exhibition at the museum). We are particularly interested in physical environments that may be occupied by people that are not used to live or work in that place – *occasional visitors*. These people may not have *a priori* knowledge about the environment, and they possibly want to know which activities can be accomplished in place, or how they can achieve them, or even how they can be supported by the ubiquitous computing environment. This does not happen with day-to-day home or work place activities, which, although possibly having a strong association with physical space, do not pose the same challenges referred above, i.e., people is accustomed with the physical environment, possible localized activities, and how the ubiquitous computing environment can support them.

In this work, two aspects of Activity Theory are particularly investigated with respect to ubiquitous computing support to localized activities: a) the flexible structure of an activity; and b) evolution of activity influenced by historical and social forces. A localized activity may be carried out in a variety of ways by employing different actions under different conditions. Individual characteristics

and changing local and personal context are the factors driving the structure of a localized activity. For example, the activity of visiting a museum may employ different actions and operations, depending on the visitor age (e.g., adults are mainly interested in observing artworks and learning about their details, while children would prefer to play an artwork-related game), preferences (e.g., devoting more visit time to roman artwork than to Visigothic pieces), available resources (e.g., viewing additional information about an artwork in a personal or local display vs. being directed to the museum's library when no display is available), or context (e.g., planning the visit route in function of available time or crowdedness). The specification of the support to a localized activity should thus provide that required flexibility, comprising the identification of the different combinations of goals (and respective actions) that may compose an activity, along with the possible conditions (and respective operations) under which each action may be executed. Decomposing an activity into different levels of granularity and identifying individual characteristics and context elements applying to each sub-component of activity should provide the means for specifying the support to a localized activity with the required flexibility.

Social interaction and historical background should also be considered in the support to a localized activity, especially in occasionally visited environments, where sharing experiences between users and recording experiences for future remembering are of special relevance. The way a localized activity is performed may develop along time. New types of resources may become available, better processes may be unravelled, etc. As stated by Bardram [4], an activity plan should be flexible enough to allow adaptation to variable circumstances and to integrate feedback from users. A localized activity evolves based on accumulated experience, and thus this experience should be kept for future accomplishments. Moreover, social interaction plays an important role in the dissemination of experience. Experience is generally shared between people (e.g., adults to children, teachers to students, etc.). Thus, besides mechanisms for experience memory, learning, and configurability, the support to a localized activity should also include mechanisms for experience to be shared among people.

In our view, providing support to localized activities in a user-centered manner (personalized, context-aware, and without obliging users to considerable efforts) requires a thorough knowledge about both the local and personal environments, and can only be accomplished with a strong integration between both these environments. Personal environments – devices, applications, preferences, context information, etc. – are the driver for the user-centered aspect of the support to localized activities, whereas local environments provide the association with the elements in the physical environment of the activity. Personal environments cannot be prepared beforehand to support all possible activities users may perform in the many different physical environments to which they can move. Activities associated to a specific place have particular characteristics that cannot be foreseen by someone developing pervasive computing systems to be used in any location. Additionally, details about activity unrolling that may differ between individuals (e.g., supporting an activity performed by a blind per-

son) are better determined locally than by some external personal entity. It is thus reasonable to expect that support to a localized activity is better managed by the local environment (or by some local entity, e.g., a city council centrally supporting activities that may be unrolled in the city area), which owns most of these knowledge and resources, than by the personal environment, which knows the user domain but is hardly aware of details regarding activities that may be locally unrolled.

### 3 Related work

Existing location-based services, e.g., mobile network operator (MNO) portals, can be seen as a form of support to localized activities in that they can provide information that is related with the user's current location. However, they are too generic to provide real value to satisfy specific user needs and are generally not able to explore the resources that may be available in the physical environment. Location-aware systems that are targeted at specific environments (e.g., museum assistants) can be made to fully explore the capabilities of the local infrastructure to support a localized activity. However, they lack the integration with personal resources, either treating users anonymously, thus hindering personalization, or offering their own user identification mechanism, obliging users to deal with possibly different interaction mechanisms and to manage several duplicates of their personal data across different environments.

Project Aura [2] implements the concept of task-driven computing by capturing user intent and mapping it into a task corresponding to a set of abstract services, which are further concretized by the environment infrastructure providing continuous support to user tasks regardless of the environment in which the user is. What mainly distinguishes the work presented in this paper from Aura is the location scope of an activity and how much the user is accustomed with the activity in that location. While Aura addresses daily routine activities that may span more than one location, our work is targeted at activities with a strong association to a specific, occasionally visited location.

Christensen and Bardram [1] also grounded on Activity Theory to develop a pervasive computing system supporting collaborative activities within specific environments. However, unlike our work, their effort is centered on environments where users are well-known and not for situations where the user population is dynamic and unknown, which requires a special attention to issues regarding integration with personal environments.

Our previous work in the VADE project [5] introduced the concept of Value Added Environment as an administrative and physical domain where the locally available computing facilities can be combined with the personal environment of visiting users. The overall scenario is that when entering a VADE, mobile users are provided with functionality that corresponds to the dynamic combination of predefined preferences, currently active applications, current user context and locally available services and applications. This approach successfully attains some level of integration between personal and local environments. However, the

system does not consider the concept of activity in the functionality provided to users. It would be valuable to enrich the integration possibilities to other types of personal environments (not only web portals) and to broaden the user interaction means beyond the personal device (e.g., using local displays).

## 4 Scenario

A scientist - João - was invited for a talk in a seminar on computer science taking place at the University of Minho. The seminar organization deployed a localized activity support through which seminar participants are provided with assistance during the seminar activity. Each speaker was previously sent an invitation code through SMS, which is further used for identification purposes when arriving at the University of Minho. The support to the seminar activity is composed of a variable sequence of actions: going to the seminar room, attending to seminar talks, making a talk, meeting seminar participants, etc. Each action is decomposed in a set of operations (e.g., setting up the personal laptop, controlling the overhead projector, etc., for making a talk) requiring different resources from both the local and personal environments. Depending on user context, some of the actions/operations may not be performed or their order may vary, and required resources may be allocated between local and personal environments, depending on their availability and adequateness to context.

When arriving at the university main hall, João sees a public LCD display which he guesses may provide him with help. Just below the display, he sees an infra-red receiver to which he beams the seminar invitation code. The University of Minho ubiquitous computing infrastructure associates that code to the seminar activity and shows through the display some initial support to João, e.g., instructions explaining him how to go to the room where the seminar takes place. João is told to browse in his PDA through his MNO portal for more advanced help. He logs in the portal and is now presented with an additional portal functionality called “Seminar at University of Minho” corresponding to the support to that localized activity. The support begins with detailed guidance to the seminar room; when in the seminar room, João browses through the scheduled talks and reads detailed information about talks and respective speakers; when the time comes to his talk, the activity support allows him to control the overhead projector; after the end of the talk and discussion, João can save in his PDA a sound record of the discussion which he may analyze later; at the conclusion of the seminar, João is asked to fill in a survey about the quality of the seminar. The activity support always include directions to the bar, toilets, and, at lunch time, to the restaurant (including the menu).

## 5 Proposed approach

The vision supporting this work poses essentially two categories of challenges: activity management (specifying an activity and managing its accomplishment) and resource management (composing and coordinating the resources needed

for the activity unfolding). These research tasks are further described, along with our proposal of an architecture for supporting localized activities, which we intend to instantiate in our initial demonstrative scenario.

### 5.1 Activity modelling

Although not intending to contribute especially to the field of activity specification or task-description languages, this work has to explore previous results in the field [2, 6, 7] in order to define a comprehensive model for the representation of a localized activity, considering different purposes:

- identifying which localized activity the user may be interested in accomplishing - this involves matching activity descriptions with personal profile, preferences, context, and resources.
- describing the activity plan - which actions and operations compose the activity, how context influences activity, which resources are required in each stage of activity, etc.
- interrupting and further resuming activity - here, activity description must be more concrete, defining who was unrolling the activity, what had been done before interruption, or what is needed (resources, context, etc.) for resuming activity.
- remembering or sharing an activity experience - this activity description must be preferably in a human-understandable language, although it may also include a machine-understandable version so that the experience may be re-instantiated in the local infrastructure by the same person or by whom it was shared with.
- customizing activity experience - this involves defining which parts of the activity structure may be customizable and how the customization can be done. The end result of an activity customization could be the machine-understandable version of the description for remembering or sharing purposes.

### 5.2 Integration with personal environment

The integration between local and personal environments begins at the moment when a new user enters in the physical environment and ends when the user leaves. The first integration stage is the detection of the personal environment and interaction with it in order to determine the user's profile, preferences, context, and available resources. This work considers that a ubiquitous computing environment should not impose any special technology to its users, but rather cover a wide range of heterogeneous personal environment types. Therefore, different mechanisms should be devised for detecting users, as long as interacting with different types of context providers, profile types, resources, etc., which may last for the whole activity experience (e.g., keeping information about user context or available resources up to date throughout the activity). Secondly, advertising possible activities and supporting the activities themselves, possibly requires from the local infrastructure to integrate the user interaction mechanisms

into different types of personal environments, unless the interaction is performed uniquely through local means. To deal with this issue, this work builds on existing solutions for multi-modal and multi-channel interfaces [8]. Both these issues require a technical survey of existing types of personal environments, studying their communication protocols, APIs, information representation formats for context, profile, and preferences, presentation languages, etc.

### 5.3 Architecture

Aiming at dealing with both issues of activity management and integration of local and personal environments, we propose an architecture for supporting localized activities (see Fig. 1), which imports some ideas from the Aura architecture [2], and is also designed to separate user interaction from data and logic.

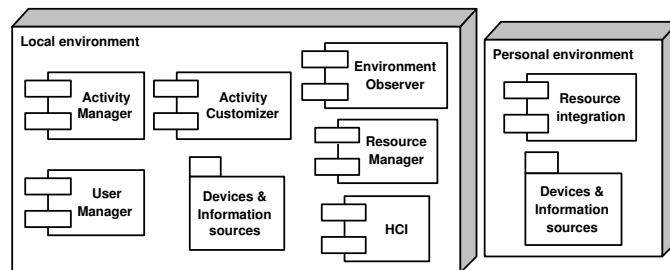


Fig. 1. Architecture

**Activity Manager** – keeps specifications of one or more localized activities and, each time a user enters the environment, is responsible for checking which activities can be unrolled; manages the sequence of actions/operations, depending on the user and local context; manages activity interruption and later resumption; is responsible for providing feedback to users.

**Activity Customizer** – provides mechanisms for managing customization, remembering, and sharing of localized activities by users.

**Resource Manager** – allocates local resources to activities, managing possible situations of concurrency between multiple users; evaluates resource characteristics in order to select the optimal configuration; discovers available personal resources and manages the access to them.

**Environment Observer** – detects users entrance, changes in users' context, or user interaction signals, either by means of local sensors or through operations executed by users.

**User Manager** – keeps users' information specific to localized activities they have been carrying out (e.g., activity state for later resumption, activity description for further remembering, etc.).

**Human-Computer Interaction component** – presents information to and accepts input from users, either through local devices or through the personal environment.

**Local devices and information sources** – local resources managed by the Resource Manager.

**Resource integration** – mechanisms allowing for the discovery (by the local Resource Manager), access, and integration of personal resources into the flow of localized activities.

**Personal devices and information sources** – personal resources to be managed by the local Resource Manager.

#### 5.4 Validation scenarios

We are developing a group of representative scenarios, in which people are more or less equipped and their personal environments embody different types (e.g., a personal profile in a web portal, a personal ICQ number, etc.). In our initial prototype, we are building on the VADE infrastructure [5], which sets a basis for the integration between local and personal environments, seeking to augment it with an activity-centered approach, by adapting it to the architecture here defined. In our initial scenario (see Sect. 4), the local environment corresponds to the University of Minho ubiquitous computing infrastructure and is characterized as follows. The Activity Manager is responsible for following the sequence of actions/operations during the seminar, depending on the user and local context; it is also responsible for providing feedback to users about how the activity is being unrolled. The Resource Manager only knows about João's PDA and personal service environment and manages local resources, either for local use, e.g., controlling the overhead projector, or for integration with the MNO portal, e.g., providing the access to the local functionality supporting the activity. The Environment Observer processes information coming from the infra-red receiver and both from user interaction and context information. The HCI component is constituted by markup code locally generated and integrated into the portal page. Local devices and information sources include the public LCD display, the overhead projector, the sound recorder, maps, instructions, the restaurant menu, and seminar details. João's personal environment is composed of the invitation code, a PDA with connectivity, his MNO portal and personal service environment [9]. The integration of the support to the seminar activity into João's MNO portal is made possible by providing the seminar invitation code to the portal service, which is thus able to integrate local functionality when João enters the University of Minho area.

Further versions of the prototype will concretize more advanced scenarios, in which we will improve the support to heterogeneity, by developing the integration with other types of personal environments other than portal services. We will also develop our activity model by including the support to activity interruption/resumption, remembering, sharing, and customization.

## 6 Conclusion

This work investigates how ubiquitous computing environments can offer user-centered support to localized activities performed by occasional visitors, i.e., activities having a strong association with a specific physical environment, which may be visited by people who are not accustomed to it. In our view, this can only be accomplished by: a) understanding the way humans achieve their activities; b) by applying this knowledge to the modelling of the ubiquitous computing support to activities; and c) by integrating the local infrastructure with the personal environment, so that the support is user-centered. Our work thus explores activity specification models which are adequate to the support to localized activities and mechanisms allowing for the integration between local and personal environments. An initial prototype is being developed, putting into practice our previous experience in the integration between local and personal environments and extending it with the concept of activity.

## 7 Acknowledgements

We are thankful to the Portuguese Foundation for Science and Technology which has been funding this work.

## References

1. Christensen, H., Bardram, J.: Supporting human activities - exploring activity-centered computing. In: Ubiquitous Computing 2002 (UbiComp 2002). Volume 2498., Göteborg, Sweden, Springer-Verlag (2002) 107–116
2. Sousa, J.P., Garlan, D.: Aura: an architectural framework for user mobility in ubiquitous computing environments. In: 3rd Working IEEE/IFIP Conference on Software Architecture, Montreal, Canada, Kluwer Academic Publishers (2002) 29–43
3. Leontiev, A.N.: The problem of activity in psychology. In Wertsch, J., ed.: The Concept of Activity in Soviet Psychology. Sharpe, Armonk, USA (1981) 37–71
4. Bardram, J.: Plans as situated action: An activity theory approach to workflow systems. In: Fifth European Conference on Computer Supported Cooperative Work (ECSCW '97), Lancaster, United Kingdom, Kluwer Academic Publishers (1997) 17–32
5. José, R., Pinto, H., Meneses, F., Vilas Boas, N., Rodrigues, H., Moreira, A.: System support for integrated ubiquitous computing environments. In: System Support for Ubiquitous Computing Workshop at the Fifth Annual Conference on Ubiquitous Computing (UbiComp 2003), Seattle, USA (2003)
6. Paolucci, M., Shehory, O., Sycara, K., Kalp, D., Pannu, A.: A planning component for RETSINA agents. In Wooldridge, M., Lesperance, Y., eds.: Lecture Notes in Artificial Intelligence, Intelligent Agents VI. MIT Press, Cambridge, USA (1999)
7. Simmons, R., Apfelbaum, D.: A task description language for robot control. In: IEEE/RSJ International Conference on Intelligent Robotics and Systems, Vancouver, Canada (1998)
8. W3C: Device independence activity (2004) <http://www.w3.org/2001/di/>.
9. 3GPP: 3rd Generation Partnership Project (2004) <http://www.3gpp.org/>.

# Designing for Physical-Virtual Activities

Thomas Pederson

Department of Computing Science, Umeå University  
SE-90187 Umeå, Sweden  
top@cs.umu.se

## 1 The Emergence of Physical-Virtual Activities

Computers, embedded in the “background” as well as more obtrusive artefacts (e.g. PCs, PDAs, cellular phones), play an increasingly important role in human activity. However, there are still things that most people would prefer to do “off-screen” in the physical (real) world, such as having parties, reading long text documents, or spending vacation. I argue that there exists a class of activities that are neither physical or virtual, but “physical-virtual” [2]. People frequently do parts of an activity in the physical world (e.g. proof-reading a text document under construction) and parts in the virtual world (e.g. adjusting paragraphs within “the same” document in a word processing environment). This behaviour is likely to become more common. Hence, future environments should be designed with such physical-virtual activities in mind.

### 1.1 Goal and Approach

For this purpose, I propose a physical-virtual design framework to deal with the gap between the physical and the virtual world<sup>1</sup>, and facilitate the exploration of designing information technology for helping human agents bridging it. The assumption is that a reduced physical-virtual gap means less “friction” for physical-virtual activities. Physical and virtual space is modelled together, and automatic mechanisms for synchronising related phenomena in both worlds are offered. By viewing the physical and virtual worlds as one, I believe the chance to *make* them one increases.

## 2 A Conceptual Framework for Physical-Virtual Design

The need to take into account the physical world when studying and designing for Human-Computer Interaction (HCI) has gained increased recognition for the past 10 years. The proposed physical-virtual design framework rides the same wave of “HCI field expansion” but takes a quite extreme stance by viewing the physical and the virtual world as existing in parallel and of *equal importance* for interaction.

We limit ourselves to activities that a) have a clear meaning, b) are observable by a human agent, and c) are observable by an artificial agent. Although this narrows the scope of the model significantly, e.g. it leaves out pure cognitive and social processes) we believe that for our purposes, the gain in modeling power compensates for it. A dis-

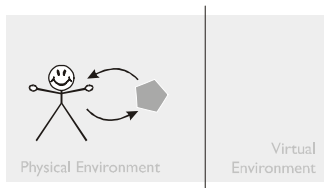
---

<sup>1</sup>Dimensions of the gap between the physical and virtual worlds are explored in [2].

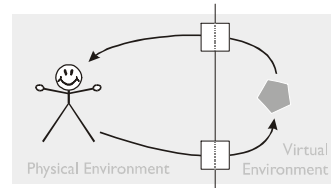
tion is made between physical, virtual, and physical-virtual activities. The notion of activity is furthermore divided into operations, actions and activities depending on the level of abstraction.

## 2.1 Outset: Physical, Virtual, and Physical-Virtual Actions

Adopting the physical-virtual design perspective involves abstracting away the classical HCI concepts of input and output devices, giving them a background role as Inter-World Event Mediators (IWEMs). Fig. 1 and 2 illustrate the basic cases of physical and virtual human action (object manipulation). IWEMs are shown as white squares.



**Fig. 1.** Physical action [2]



**Fig. 2.** Virtual action [2]

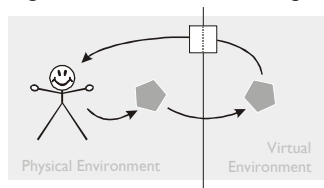
In order to arrive at a definition of physical-virtual activity I have found it useful to define human border-bridging activity on a lower level of abstraction first:

**DEFINITION 1:** A *physical-virtual action pair* consists of two actions belonging to the same activity and often time-wise adjacent, where the first action is constrained (by lack of action support in the current environment) or chosen (e.g. based on individual preferences) to be performed in the physical world and the other action is constrained/chosen to be performed in the virtual world, or vice versa. [2]

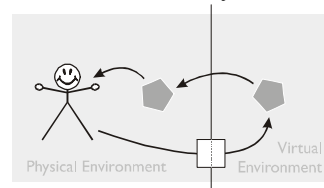
**Physical-Virtual Artefacts.** Among physical-virtual action pairs we can sometimes identify one or several information-mediating objects that are subject to indirect or direct human manipulation in *both* actions, objects that transcend the physical-virtual border by being present in both worlds. Such objects are referred to as Physical-Virtual Artefacts (PVAs) and for denoting the presentations of them in the two different worlds, the term PVA manifestation is used. A text document presented in both the physical (e.g. printed on paper) and the virtual world (e.g. within a word processing environment) would serve as a good example of a PVA, where each manifestation affords different kinds of manipulation.

**DEFINITION 2:** A *physical-virtual action* is an action on a PVA where both the physical and virtual manifestations are directly controlled and/or monitored by the agent. [2]

Fig. 3 and 4 illustrate two possible kinds of physical-virtual actions. Finally, the concept of physical-virtual activity is defined as follows:



**Fig. 3.** Physical? ? virtual action [2]



**Fig. 4.** Virtual? physical action [2]

cept of physical-virtual activity is defined as follows:

**DEFINITION 3:** A *physical-virtual activity* is an activity consisting of a sequence of actions containing a) at least one physical-virtual action pair or b) at least one physical-virtual action. [2]

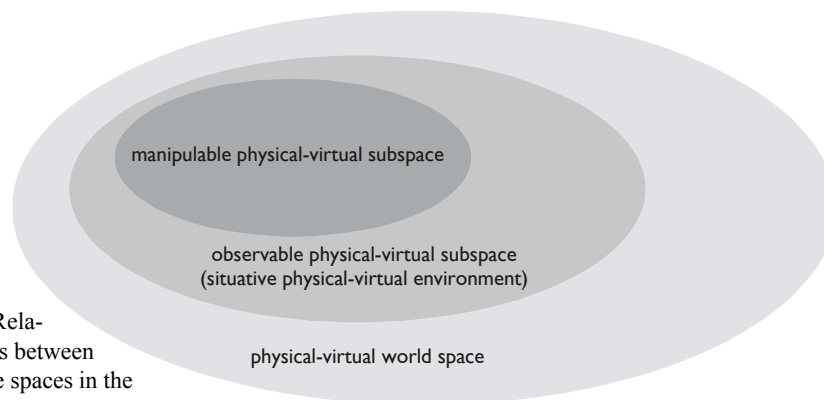
## 2.2 One Space, One Magnifying Glass

By viewing the physical and the virtual worlds as equally important for human activity, the proposed design framework makes terms with implicit virtual-world bias like “context awareness” become obsolete. It also expands the meaning of “location tracking” (currently having an implicit physical-world bias) to include also space and place in the virtual world. It invites the viewing of the relationship between physical and virtual environments from unconventional angles. For instance, why should not the current (local) state of the virtual world influence how activity in the physical world is interpreted? Could it not be of use for physical-world “applications” to be aware of their virtual-world context? And why is virtual-world location tracking (e.g. web pages browsed by a human agent) not considered when designing interactive environments?

The proposed physical-virtual design perspective is based on a series of world-neutral definitions of space that allow us to disregard whether common human actions such as navigation and object translation take place in the physical or the virtual world. Two models developed so far are briefly presented below.

## 2.3 A Situative Physical-Virtual Space Model

At any given point in time, a specific human agent is able to (visually) observe only parts of the physical and the virtual world. Furthermore, only parts of these observable “sub-worlds” contain objects that can be manipulated. Thus, in any given situation, three inclusion-related spaces can be distinguished in the physical and the virtual world respectively, as pictured in Fig. 5. While the general “object translation pattern” looks similar for both the physical and the virtual world as to where “hot” and “cold” [4] objects can be found, there are differences when it comes to the frequency and dynamics of object translation. In the physical world, objects often enter and leave the observable subspace in large “chunks” as the human agent navigates in the physical world space, e.g. by leaving one room for another. In the virtual world, by contrast, objects are often moved into and out of the observable subspace one at a time.

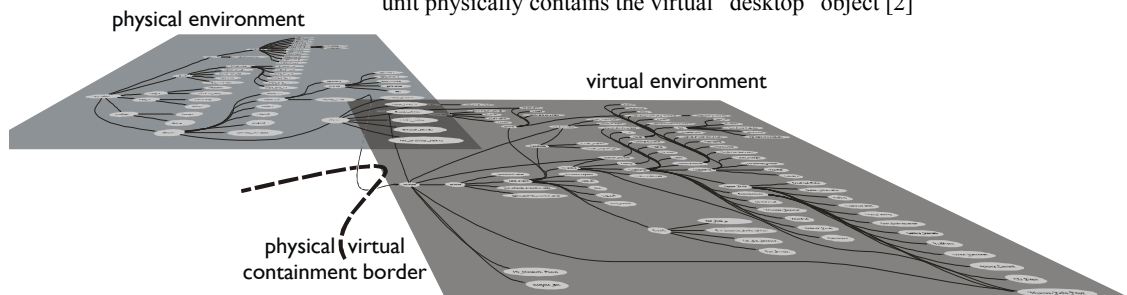


**Fig. 5.** Relationships between the three spaces in the proposed situative physical-virtual (PV) space model: A small subspace of the physical-virtual world space is observable, and a part of that observable subspace is also manipulable, by a human agent at a given point in time [2]

## 2.4 A Hierarchical Model of Physical-Virtual Space

Both physical and virtual environments can be modeled as hierarchies based on the objects situated in them and containment relationships between those objects. However, because the physical and virtual worlds typically differ structurally, "containment" cannot mean exactly the same thing in both worlds. Furthermore, in the virtual world, cheap "cloning" of objects as well as independency from laws of nature opens up for a more irregular structure compared to the physical world. Fig. 6 shows a particular physical-virtual environment structured around containment-relationships.

**Fig. 6.** A physical and a virtual environment visualized as a physical-virtual containment hierarchy. The border between the physical and the virtual world is at the point where the PC display unit physically contains the virtual "desktop" object [2]

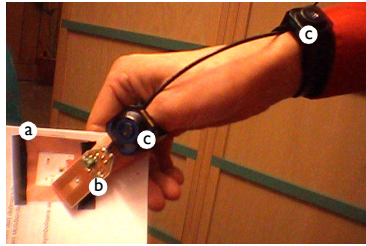


## 3 Magic Touch — A Physical-Virtual Prototype System

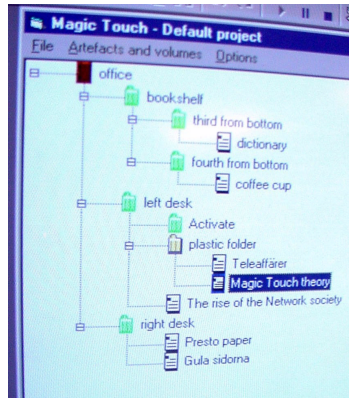
Many of the concepts within the physical-virtual design framework have been inspired by, and has inspired, the development of the physical-virtual prototype system Magic Touch [3]. The system is primarily focused on enabling the definition of PVAs and to mediate basic manipulation of physical PVA manifestations (e.g. paper documents) to their virtual counterparts (e.g. web pages). Fig. 8 shows the conceptual system architecture, Fig. 9 shows the wearable object identification and location tracking unit, and Fig. 10 shows a visualisation of the real-world office environment in which the system is installed, based on containment relationships between physical objects.

## 4 The Physical-Virtual Framework and Activity-Centric Design

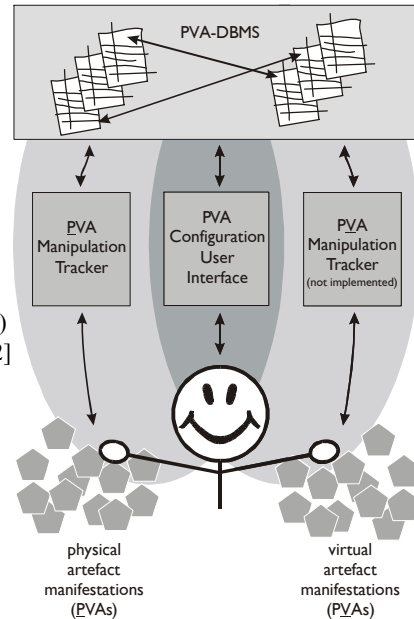
I believe that when moving from a classical software application-centric view towards an activity-centric view of HCI settings (as proposed in [1]), one would gain in modelling power by incorporating relevant parts and states of the physical world. If done properly, physical-world and virtual-world prerequisites for successfully performing a specific physical-virtual activity (i.e. all necessary physical and virtual tools and data are available) can be modelled in a straight-forward fashion. If computing systems would maintain such physical-virtual models they could suggest alternative ways of performing tasks depending on available resources, *across the physical-virtual border*. The system could suggest to substitute a missing virtual object with an available physical one, or give hints to the human actor on ways of performing an intended activity



**Fig. 9.** Magic Touch PVA Manipulation Tracker v.0.51 in action. The photo shows (a) an RF/ID tag attached to a paper document, (b) a stiff antenna, and (c) position transmitters [2]



**Fig. 10.** Parts of the PVA Configuration UI of Magic Touch 1.0 showing a hierarchical virtual representation of objects in a physical environment based on containment relationships [2]



**Fig. 8.** Magic Touch conceptual system architecture. The PVA-DBMS linking the physical (left) and virtual (right) environments together by keeping track of artefact manifestation changes done by the human agent in any of the two environments [2]

more efficiently by switching from the physical to the virtual world or vice versa. Incorporating real-world objects and phenomena into the interaction model also opens for an intuitive inclusion of mechanisms and phenomena commonly referred to as “interaction context” in the Context-Awareness communities. In fact, “context” phenomena would no longer be external to the activity, but simply part of it.

## References

1. Christensen, H. B., Bardram, J.: Supporting Human Activities - Exploring Activity-Centered Computing. In Borriello, G., Holmquist, L. E. (eds.), Proceedings of UBIComp 2002, Lecture Notes in Computer Science **2498** 107–116, Springer Verlag (2002)
2. Pederson, T.: From Conceptual Links to Causal Relations — Physical-Virtual Artefacts in Mixed-Reality Space. PhD thesis, Dept. of Computing Science, Umeå university, report UMINF-03.14, ISSN 0348-0542, ISBN 91-7305-556-5 (2003)  
Permanent URL: <http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-137>
3. Pederson, T.: Magic Touch: A Simple Object Location Tracking System Enabling the Development of Physical-Virtual Artefacts in Office Environments. Journal of Personal and Ubiquitous Computing **5** 54-57, Springer Verlag (2001)
4. Sellen, A. J., & Harper, R. H. R.: The myth of the paperless office. MIT Press (2002)

## **Author Biography**

- 2003 PhD thesis defended. Faculty opponent: Professor William Buxton, Univ. of Toronto.
- 1998-2003 PhD studies and teaching at Department of Computing Science, Umeå University having Professor Lars-Erik Janlert as supervisor. Member of the Cognitive Computing Lab within the UCIT (Umeå Centre for Interaction Technology) consortia.
- 1998 Guest research position at Fraunhofer-IPSI (previously GMD-IPSI), Darmstadt.
- 1997 MSc thesis project at Ericsson Media Lab, Stockholm.

# Three Desirable Properties of Activity-Oriented Intelligent Systems

Fahd Al-Bin-Ali<sup>1</sup> and Nigel Davies<sup>1,2</sup>

<sup>1</sup> Computer Science Department, University of Arizona,  
Tucson, AZ 85721, USA  
[albinali@cs.arizona.edu](mailto:albinali@cs.arizona.edu)

<sup>2</sup> Computing Department, Lancaster University,  
Lancaster, LA1 4YR, UK  
[nigel@cs.arizona.edu](mailto:nigel@cs.arizona.edu)

**Abstract.** Much effort has gone into building ubiquitous systems that react to users' context. However, relatively little effort has gone into building systems that are aware of users' activities. Here, we identify three desirable properties of intelligent techniques for supporting activity awareness in ubiquitous systems: comprehensibility, adaptability and accuracy. We explain what these characteristics are, why they are important and how systems can support them. In addition, we describe a prototype system that we are building to infer users' activities.

## 1 Introduction

Context-Aware computing has occupied a prominent place in ubiquitous computing research. The importance of context awareness stems from a need to adapt services and applications in ways that take advantage of users' context such as location, proximity to objects or even users' mood. Earlier research in ubiquitous computing has typically focused on establishing simple relationships between tangible aspects of context and appropriate actions, for example, switching on and off devices based on user proximity. Intangible aspects of context (that are not directly measurable) such as activities, human moods and human intentions have not received significant attention to date. However, to engage users in richer and indeed more meaningful interactions, systems should have an understanding of both tangible and intangible aspects of context.

One novel computing paradigm that promises better understanding of context is activity-based computing in which systems interpret behaviors by considering users' activity as the fundamental unit of analysis. Such systems will require (at least) three functionalities including: monitoring activities using sensors, analyzing sensor data to identify activity patterns and inferential relationships and modifying systems' functionality to adapt to changes in activity patterns.

In this paper, we identify three desirable properties of intelligent techniques for supporting activity awareness in ubiquitous systems: comprehensibility, adaptability and accuracy. The remainder of this paper explains what these concepts are, why they are important and how systems can support them. We hypothesize

that these characteristics will be central to the design and to the deployment of ubiquitous systems that understand users' activities and therefore can engage users in more intimate interactions.

## 2 Comprehensibility

### 2.1 What it is.

Activity aware systems will enable users to delegate tasks to systems when certain activities take place. That means that users will have to trust systems' decisions and ability to correctly recognize activities and systems will have to generally produce reasonable results. When systems produce errors, users will disagree with systems and will require means to deactivate and isolate malfunctioning components and to cancel their actions. Moreover, users should be able to initiate dialogues with systems to correct errors. Hence, we believe that building comprehensible systems is essential and will require supporting (at least) two levels of comprehensibility:

1. Execution: systems seem to generally produce reasonable results.
2. Explanation: systems can produce a meaningful account of their actions.

### 2.2 How to support it.

First, systems must use training datasets that reflect behaviors and actions indicative of users' activities. Generally speaking, intelligent techniques will implicitly assume that training datasets are product of rational user behavior, whereas in reality, user behavior might be based on arbitrary factors. Previous work [2] has proposed controlled methods for training systems (e.g. via demonstration). Such approaches eliminate doubts about the validity of the data, however they require frequent user involvement which can be a source of distraction. In contrast, approaches that continuously monitor spaces and do not require user involvement will have to distinguish between data that reflect rational behavior and data that do not.

Second, activity patterns will vary across time and space: morning activities will differ from evening activities and public spaces such as coffee shops are likely to host more dynamic activities than private spaces such as homes. Different tools will be able to model such patterns making it critical to choose a solution that provides simplicity and translucency. For example, simple parametric methods (when suitable) can provide explanations at a general level using coefficients and signs of coefficients thus producing explanations to the general user.

Finally, systems need to have an ability to explain their actions. This requires extracting causal relationships between inputs and responses. Here, a distinction should be drawn between 3 similar but different concepts: correlation, dependency and causality. Correlation is a special case of dependency in which factors vary linearly. Dependency includes all observable variations. Causality on the other hand encompass the key to understanding and tracing causes and effects.

Neither correlation nor dependency imply causality, for example, a cooking activity might positively correlate with the temperature in the kitchen, this does not mean though that an increase in temperature causes a cooking activity but in fact the opposite. Determining the exact causal relationships between various factors will be very important for systems to initiate and sustain dialogues with end-users. We note however that this will require extensive analysis of user traces and might possibly require involving the end-user.

### 3 Adaptability

#### 3.1 What it is.

Adaptability provides systems with an ability to cope with the highly dynamic nature of ubiquitous environments. Such adaptability must apply to both physical reconfiguration of spaces (e.g. changes in the availability of sensors) and to changes in activity patterns within these spaces. Different systems will require different forms of adaptability including off-line adaptability in which sensor data is logged for later analysis and on-line adaptability in which sensor data is examined and adaptation is performed while the system is in use. On-line adaptability is likely to be harder than off-line adaptability.

#### 3.2 How to support it.

The simplest approach to adaptation is to retrain the system. For example, in parametric models, this will require reestimating coefficients and in some cases might require changing the parametric model. This approach though does not seem to suit the stochastic nature of the data collected in ubiquitous environments. Alternatively, systems can store intermediate states to perform incremental training or systems can use stochastic techniques in which data is included in models as soon as they arrive [5]. A clear challenge though is to identify what data to include or exclude in the modeling process.

Another possible way for supporting adaptation is through exchanging knowledge about activity patterns across spaces. Clearly, it is important that knowledge about users and their activity patterns can be moved between ubiquitous environments, reflecting user mobility inherent in the real world. With an ability to disseminate knowledge about activity patterns, systems will be able to accelerate training and indeed rapidly adapt to unfamiliar activities. In practice, disseminating knowledge about users' activities will require a clear separation between models and the system-specific assumptions and mechanisms. This raises many technical challenges (e.g. determining the equivalence between sensors in different environments) and non-technical challenges in areas such as legal and social ethics (e.g. can models about activity patterns be exchanged between private and public places without violating the privacy of people?).

Finally, reducing the dimensionality of input data can potentially accelerate adaptation making it especially suitable for on-line adaptability. We note however that data reduction techniques result in information loss, therefore, it is

critical to preserve the information necessary to produce reliable inferences. In addition, it seems that there is a tradeoff between data reduction and data interpretation: most data reduction techniques apply transformations to datasets to produce new datasets of lower dimensionality. This often complicates interpreting the resulting quantities.

## 4 Accuracy

### 4.1 What it is.

Clearly, achieving high accuracy in terms of identifying activities is crucial. However, it should be noted that the exact requirements in terms of accuracy are influenced by the significance of the actions that will be triggered: users will perceive identifying activities as accurate and indeed as invisible when the system's reactions are correct. This does not necessarily mean that the system has identified the user's activities correctly. For example, imagine a user having a nap while watching TV. A ubiquitous system might detect a reduction in the overall mobility in the space and therefore infers that no one is in the room; resulting in switching off the TV or the lights. Clearly, the system misdiagnosed the activity, but the outcome is still considered accurate by the user.

### 4.2 How to support it.

We summarize three important factors that will enable systems to accurately infer activities. We note however that these factors might not be applicable in every situation. First, the intelligent technique employed should be suitable for the activity patterns under investigation. For example, when dealing with activities (that are inherently categorical), quantitative techniques such as linear regression will fail to analyze the data while techniques such as logistic regression are suitable. Second, training datasets should span the whole decision space in order to reflect all possible outcomes and to allow for optimal placement of decision boundaries. Third, training datasets should encompass data of good and consistent quality to minimize the potential for error. In general, systems that can follow these guidelines will produce better results.

## 5 System Overview

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 and analyzes various behavioral patterns. We therefore built an activity classifier system and installed it in a domestic space. The classifier goes through a training phase where users specify to the system their activities (e.g. cooking, exercising, dancing) as they participate in them using wireless handheld devices. Our activity classifier is part of a larger system that provides services in a smart space. Next, we give a brief overview of our system components.

## 5.1 The Sensing Subsystem

The sensing subsystem has 5 smart-its boards. A smart-its board [3] 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 every 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.

Each board is also equipped with a radiometrix BIM3 chip that transmits and receives data wirelessly over a short range. 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 predicts the activities by fitting a Multinomial Logistic Regression (MLR) to the training data. MLR classifies inputs based on responses with the largest expected probability for a factor/covariate pattern. The model assumes that the probability of occurrence of activities follows a multinomial distribution and that the conditional probability of each activity given the sensor data is represented by the following logistic formula:

$$\pi_{ij} = \frac{e^{x_i^T \beta_j}}{1 + \sum_{k \neq j^*} e^{x_i^T \beta_k}} \quad (1)$$

where  $\pi_{ij}$  is the conditional probability of activity  $j$  given  $x_i$ ,  $x_i$  is a vector of  $p$  sensor readings and a constant term  $x_{i0} = 1$ ,  $j^*$  is a reference activity (excluded from the model as its conditional probability can be computed trivially under the assumption that the activities are collectively exhaustive) and  $\beta_j$  is a vector of coefficients of size  $p + 1$ .

Our model uses maximum likelihood estimation to estimate the coefficients of the logistic formulae and subsequently classifies the activities. Details of MLR can be found in [4].

Initial experiments with MLR provide reasonable classification accuracy when the temporal structure in the data is not critical. Furthermore, odds ratios (that explain the relationship between coefficients of sensors and activities) can easily be computed from the logistic formulae thus potentially providing explanations to end-users at a general level. More details about our experiments can be found in [1].

## 6 Conclusion

In this paper, we identified three desirable properties of intelligent techniques for supporting activity awareness in ubiquitous systems: comprehensibility, adaptability and accuracy. We described a prototype system that classifies activities using multinomial logistic regression. Currently, we are investigating several enhancements to our system including temporal and spatial modeling and relaying explanations to end-users. We hope that our future work will provide better understanding of activity awareness in ubiquitous environments.

## References

1. Al-Bin-Ali, F., Boddupalli, P., Davies, N., Friday, A.: Correlating Sensors and Activities in an Intelligent Environment: A Logistic Regression Approach. In: Proc. EUSAI: Lecture Notes in Computer Science, Vol. 2875. Springer-Verlag, Berlin Heidelberg New York (2003)
2. Brumitt, B., Meyers, B., Krumm, J., Kern, A., Shafer, A.: EasyLiving: Technologies for Intelligent Environments In: HUC: Bristol, UK (2000)
3. Holmquist, L., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., and Gellersen, H.: Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artifacts. In: Proc. Ubicomp 2001. Atlanta, GA, 2001.
4. Hosmer, D., and Lemeshow, S.: Applied Logistic Regression. John Wiley & Sons, 2nd Edition (2000).
5. Oliver, N., Horvitz, E. and Garg A.: Layered Representations for Human Activity Recognition. In Proc. of the 4<sup>th</sup> IEEE International Conf. on Multimodal Interaction, Pittsburgh, PA, October 2002.

**Fahd Al-Bin-Ali** is a PhD student and a research associate at the Computer Science Department at the University of Arizona. His research interests include ubiquitous computing, intelligent environments and statistical analysis of sensor data.

**Nigel Davies** is a professor of computing at Lancaster University and an associate professor of computer science at the University of Arizona. His research interests include systems support for mobile and pervasive computing. Davies is also an associate editor of IEEE Transactions on Mobile Computing and IEEE Pervasive Computing.

# Open Issues in Activity-Based and Task-Level Computing

Jakob E. Bardram and Henrik Bærbak Christensen

Centre for Pervasive Computing  
Department of Computer Science, University of Aarhus  
Aabogade 34, 8200 Århus N, Denmark  
{bardram,hbc}@daimi.au.dk

**Abstract.** The prevailing computer paradigms do a poor job at meeting their human users at their level of abstractions. Humans organize work and leisure in more or less well defined tasks and activities. Computers organize computing in terms of applications, files, networks, etc. This abstraction gap becomes a big problem in situations characterized by mobility, frequent interruptions, and collaboration—situations that pervasive computing is intended to support. A new paradigm, task-level or activity-based computing, has been proposed to lessen the abstraction gap and provide a better platform for pervasive computing. In this paper, we highlight some of the open issues that remain to be addressed in this paradigm.

## 1 Introduction

Activity-based or Task-level computing is a new paradigm for computing that tries to provide computational support for humans at a familiar level of abstraction; namely in terms of the tasks and activities the human is involved in. This is in contrast to the prevailing desktop computing paradigm that defines computing in terms of files, documents and applications, not tasks. To illustrate the difference, you may ask researcher what he is doing, and he may answer that he is writing a position paper for a workshop. However, if you look at the process from the computer's point of view, he is running a couple of acroread applications showing some PDF documents, an Emacs editor, and a shell with a history of  $\LaTeX$  commands. The computer has no notion of the “writing paper” activity. Hence, there is a gap between the abstraction level of the human and the abstraction level supported by the computing system. The result is that much time is used on opening and closing applications, on navigating user interfaces, and on reestablishing working context, when the user shifts from one task to another. In a pervasive computing environment, characterized by nomadic work, ad-hoc collaboration, and use of multiple computing devices, this is simply infeasible.

Over the years a number of systems have been designed and described that goes some way in narrowing this abstraction gap. ROOMS [5] allows the user to switch swiftly between a number of virtual desktops; each desktop was then ideal to compose the set of applications for a particular task. The SunRay system [10] introduced mobility on top of this theme. However, these systems do not tackle the basic abstraction gap;

the computing system still has no clue to the human activity, it is only the graphical user interface that goes some way in supporting activity classification.

Several research groups have focused on direct computational support for the human level. Project Aura [6, 9] introduces Task-level computing and outlines an architecture that allows computing context to migrate between devices. The ABC: Activity Based Computing project [4, 1] have designed and implemented a prototype architecture and limited end user functionality for healthcare activities, aimed at hospital clinicians. Clinicians from Danish hospitals have evaluated the proposals during a number of workshops.

ABC defines a middleware system and a programmers interface that manages activities. Activities and Activity-Based Computing are defined as:

**Activity:** An abstract, but comprehensive, description of the run-time state of a set of computational services. An activity is collaborative, i.e. it has a number of participants.

**Activity-Based Computing:** A computing infrastructure, which support suspending and resuming activities across heterogeneous execution environments and supports activity-based collaboration

In activity-based computing, however, a lot of open issues still remain. This position paper forwards a number of issues we have identified in our work with ABC, and discuss how related task-based systems, like Aura, handles these issues. We acknowledge that our primary inspiration has been case studies within the health care domain; however, we believe that the observations made are valid inspiration for all platforms that support activity based computing.

## 2 Computational Activities versus Human Intent

The core conceptual challenge in activity-based computing is the issue of *intent* – if a computer system is to support human activity, should the system have an understanding of the human intent or purpose of this activity? And what does the word ‘understand’ imply in this context?

### 2.1 Capturing User Intent

Ideally, computational activities reflect user intent: a “prescribe medicine for patient Hansen” activity will, when resumed on any available device, reestablish the complete computational working context associated with this tasks, that is: start applications, find data for patient, show proper views, ect. Thus, the physician can just continue working on prescribing medicine for Hansen. The question arises, however, who defines the intent—and when and how? Sousa et al. [9, §2.3] states that given a more sophisticated context awareness monitoring *the less Prism has to rely on explicit indications from a user concerning their intentions*. There are actually two points made in this statement. First, users may explicitly state intentions. Second, context awareness can provide all or part of the user’s intention.

We have discussed the first point with clinicians at our workshops and find it problematic if users are to give lengthy descriptions whenever they define a new activity. Even the relatively simple action of *naming* an activity when it is created is often too distracting. An extreme scenario is the physician that is called for immediate help with a patient suffering heart-attack, rushing to the nearest computer to get vital information only to be met by a friendly dialog-box: “Please enter a name for the activity”. While this is of course an extreme situation, it does show that the problem is not trivial. Experiments with default names helped a little, especially when naming according to contextual information, like the patient nearby. But it did not address the fundamental problem of revealing intent to the computer system – defining a name is only a clue for humans as to what the intention is.

This leads to the second point, namely that context awareness can define user intention. We have experimented with contextual triggering of activities [3]. Our experience is that it is very difficult to correctly infer a satisfactory complete set of attributes to define user intent uniquely from environment monitoring and context awareness—even in what first appears as obvious cases. As an example, one of our workshops had the theme: prescription of medicine. We initially envisioned that some environmental triggers, like location, time, or nearness of a set of artefacts, could define that a physician is most likely to want to initiate a “prescribe medicine for patient X” activity. However, in practice physicians prescribe medicine at odd hours and in all sorts of places. Our conclusion was that a medicine prescription activity is triggered by sources too complex to capture except by explicit indication by the user.

## 2.2 Intent or Light-weight Activities

This leads to another open issue: Humans have intentions with their tasks. Do computational activities have intent that must be modelled in order for them to be useful? Sousa et al. [9, §4] also raises the issue: ...*Aura should prove useful even with no deeper knowledge of the task* ...

A recurrent debate at our workshops with clinicians was the question on whether activities should mirror human tasks. On one hand, if the computing infrastructure has accurate knowledge of defined tasks, then transferring tasks from one clinician to another could be supported directly by the infrastructure. This happens all the time in a hospital, during the shifts when for instance the evening nurse takes over from the day nurse. Thus instead of talking to the next nurse on guard (potentially forgetting some important tasks), the nurse leaving could simply transfer the list of pending tasks to the next nurse.

We found a lot of problems with this seemingly appealing idea. First, our clinicians did not like this idea. Human communication is important, and speech is much faster than keyboard exercises on the computer. Second, our activity infrastructure drifted towards a workflow system. Workflow systems dictate how work is done; we in contrast want to support current work practices where interruptions are not an evil but key tools for communication and learning.

The current approach is to let a computational activity snapshot and reestablish composite state information in a set of applications. The link to the intent of the activity is left for the users, or their organizational setting, to establish. We hence use the term

“light-weight” about the activity because there is no explicit link between a computational activity and a human intent. The infrastructure supports fast activity switching, seamless migration to new devices, and ability to transfer computational activities to other persons as *part of* handing over responsibility and *not* as the mechanism to do so.

### 2.3 Activity Lifecycle

Another important issue is a activity’s life-cycle. Both ABC and Aura view activities as disjoint entities where one activity is “alive”/resumed and all others are paused. To humans, however, activities are not disjoint but more characterized as a fluid continuum. Consider a nurse’s activity pouring medicine for a patient into a medicine tray. This activity computationally embody the electronic patient record service running on the proper patient and having views on the medicine schema showing medicine prescribed and perhaps a web browser showing the medicine handbook. Next, the nurse needs to bring the medicine tray to the patient and document that the patient has indeed taken the prescribed medicine. The latter may logically seem another activity; but the nurse may view it as one activity fluidly turning into the next. From the computation point of view, it is properly also the same information and services that are needed. Maybe the light-weight interpretation of activities, outlined in the previous section, is more in line with the fluid way humans perceive their activities.

However, it poses a problem with how to denote and label activities. We need some kind of handle to allow us to manipulate the set of activities that we have presently paused in our computing system. If we label activities with text strings to ease browsing (as done in ABC), then what is the proper label for such activities that fluidly alters contents and intention? Text labels seem like a poor choice, but the problem is what the replacement is?

### 2.4 Organizing and Managing Activities

A physician or nurse is handling a large set of patients. For each patient, there are a number of tasks to perform. Thus, we can readily envision that each user must manage a large number of computational activities, and the question arises how to support management and browsing in a way that does not overwhelm the user. At our workshops, we have experimented with both hierarchical and linear organization of activities allowing them to be browsed. However, as stated earlier, the limited scope of our workshop did not allow us to assess browsing problems. Here the work done on user-interfaces to activity-based computing system are of interest. The Kimura system [8] provides mechanisms to organize your work in related tasks and to organize them on large wall-sized display. You then have a peripheral awareness of what is going on in the activities which ‘runs in the background’. The WORKSpace project [2] similarly have ways of organizing related work in a ‘workspace’ which can be shared with others.

### 2.5 Automation in Pervasive Computing

An interesting issue often raised is the ability for a computing system to react intelligently on context and execute tasks on behalf of the user without asking. In a task oriented system, human activities may then be inferred from context information. Project

Aura discusses a scenario where Aura is taking action based on inferred activity: Fred is rushing to a conference room, thus Aura downloads files and software to the computer in the conference room and turns the projector on.

At one of the first workshops, we presented our vision of helping nurses to *automatically* record medicine given to patients in the electronic patient record simply based on location tracking of medicine trays, nurses, and patients. We found it appealing that this was done simply when placing the medicine tray on the patient's bed table. However, the clinicians were very upset with this idea. First of all, they pointed out that medical decisions were now made by the computing system, not by trained, clinical, staff. Second, the actual action to take is not uniquely defined by the external triggers we kept track of, such as time schedules, and location of people and things. Indeed, complex human decision making based upon knowledge of the patient and the situation. Third, even the same context triggers only define a space for user intent, not an exact point. Even given the pretty comprehensive context triggers: Nurse Berg is near patient Anderson; the 12 o'clock medicine for Anderson is not yet given; it is around 12 o'clock; Anderson is in her bed; the medicine tray of Anderson is on the bed table near Anderson; we could not infer whether Berg wants to document that all medicine has been given to Anderson or just some of it (unless we track each individual tablet!). We can infer that nurse Berg probably wants to do something about medicine recording for Anderson but not precisely.

Our proposal was thus to 1) infer a set of likely user intents 2) define activities for each in the set and 3) present these suggested activities *non-intrusively* for the user (see also [3]). We concluded that 'automation' in term of higher-level activities should be done with caution and suggestive. We are currently investigate automation for more low-level technical issues, like adaptation to changes in network or execution environment.

### 3 Programming Activity-Enabled Applications

From an architectural and software engineering perspective one of the central issues in activity-based computing is, what is required by programmers in order to write services for such a task level/activity based computing infrastructures.

Aura uses XML formats to capture activity state and services/applications are responsible for creating and parsing such descriptions during when a task is suspended and resumed. A similar approach is taken in ABC by storing state information as Java objects as key/value pairs. Helper classes and UI-wrappers that automatically does state handling exists in the ABC Framework, but in principle creating and parsing application-specific state information is the responsibility of the applications.

The downside is that state check-pointing and parsing is entirely left to the programmer, which also means that access to the source-code is necessary. Hence, it is difficult to use existing application in an activity-based computing infrastructure. Plugins, macros, and other kind of code need to be written for MS Word, Emacs, and other application that participate in an activity. Suggestion to use Virtual Machines (like the Java VM) to support weak or strong migration have been suggested [7]. Our experience so far is, however, that we want the state information to be (i) very light-weight because

we use it in synchronous collaborative application sharing, and (ii) more abstract than migrating code, because we want to be able to migrate between highly heterogeneous devices, like a wall-display and a mobile phone. Clearly a Java VM exists on both platforms, but the execution environment is so different that it makes little sense to migrate running code. Hence, there is a need for a simple, small, and abstract state description.

We therefore think that it is important to find techniques that relieve the burden on a programmer to program generic state handling in an ABC-service. For instance, remote method invocations schemes allow the programmer to define just an interface, and tools create stubs, skeletons, marshalling, and network code automatically. We are currently addressing this problem. An initial idea is to annotate relevant object attributes as ‘state’ attributes that must be part of an activity’s description. This meta information may be used actively by the activity based computing infrastructure at run-time. Another challenge is to find techniques and methods for making activity-based computing support ‘application transparent’, i.e. that an application can participate in activity-based computing without changing it programmatically.

## 4 Conclusion

Activity based computing has the promise of better supporting humans working in context characterized by mobility, frequent change of what computing device to use, and frequent interruptions. As such, it presents itself as an interesting paradigm on which to base pervasive computing systems. However, the ideas and proposals are still very new, and there are a lot of questions that are still open. We have outlined some of these, and hope to generate interest and discussion at the workshop, and hopefully get some insight and new inspiration.

## References

1. ABC: Activity Based Computing. [www.pervasive.dk/abc](http://www.pervasive.dk/abc).
2. M. Büscher, M. Christensen, P. Mogensen, D. Shapiro, and P. Ørbæk. Creativity, Complexity, and Precision: Information Visualization for Landscape Architecture. In *Proceeding of ACM International Symposium on Information Visualization*, pages 167–171, Salt Lake City, USA, Oct. 2000.
3. H. B. Christensen. Using Logic Programming to Detect Activities in Pervasive Healthcare. In *18th International Conference on Logic Programming ICLP 2002*, pages 421–436, Copenhagen, Denmark, Aug. 2002. LNCS 2401, Springer Verlag.
4. H. B. Christensen and J. E. Bardram. Supporting Human Activities — Exploring Activity-Centered Computing. In *Proceedings of Fourth International Conference on Ubiquitous Computing, UbiComp 2002*, Göteborg, Sweden, 2002.
5. J. D. Austin Henderson. Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics*, 5(3):211–243, July 1986.
6. D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. Project Aura: Towards Distraction-Free Pervasive Computing. *IEEE Pervasive Computing*, 1(2):22–31, 2002.
7. M. Kozuch and M. Satyanarayanan. Internet Suspend/Resume. In *Proceeding of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA’02)*, pages 1–7, 2002.

8. B. MacIntyre, E. D. Mynatt, S. Vodia, K. M. Hansen, J. Tullio, and G. M. Support for Multitasking and Background Awareness Using Interactive Peripheral Displays. In *Proceeding of ACM User Interface Software and Technology 2001 (UIST01)*, pages 11–14, Orlando, Florida, USA, Nov. 2001.
9. J. P. Sousa and D. Garlan. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. In *WICSA*, 2002.
10. Using Smart Cards With the Sun Ray 1 Enterprise Appliance. Whitepaper from Sun Microsystems, available from [www.sun.com](http://www.sun.com), Sept. 1999.

## About the Authors

Jakob E. Bardram's main research areas are pervasive computing, distributed system, software architecture, and computer supported cooperative work (CSCW). His main focus currently is 'Pervasive Healthcare' and is conducting research into technologies of future health - both at hospitals and in the patient's home. He is a principal architect of the Activity-Based Computing (ABC) Framework which includes support for activity migration and shared activity collaboration, among other things. His current interests include software architectures and frameworks for activity-based computing and collaboration, context-awareness, and distributed computing in a heterogeneous, pervasive computing environment.

Henrik B. Christensen's research interests include software architecture, software engineering, pervasive computing, and teaching. As researcher at Centre for Pervasive Healthcare, he has been architect on the ABC framework that experiments with computer support for human activities and collaboration. His research includes conceptual frameworks, architectures, and programmers API's for activity based computing, proactive discovery of activities, and techniques and tools for engineering of reliable pervasive services.