Users' Practices and Software Qualities: a Dialectical Stance

Alessandro Pollini
Interaction Design Area,
Communication Science Dpt.,
University of Siena
Via Roma 56, 53100, Siena
0039 0577 270565
pollini@media.unisi.it

ABSTRACT

The Ubiquitous Computing technology in practice is often characterized by users that experience recurring breakdowns, standards' incompatibility and a proliferation of interfaces when using, accessing and trying to connect different devices (e.g. PCs, cameras, printers, and phones). Such interconnected devices populate ordinary Ubiquitous Computing scenarios.

The focus of the present research is on how software architecture can support Ubiquitous Computing applications and how people might use these technologies to enhance their practices and reach personal goals. Architectural support is indeed needed for designing embedded, distributed, intelligent and interactive systems, which need communication through middleware components.

Use practices and Architectural Qualities have been investigated in the Active Surfaces case study. Active Surfaces is an embedded and modular system of tiles aimed at supporting therapeutic use practices and special needs. The design and developmental process is articulated on the relationship and the exchange between key users practices and architectural qualities.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Contructs and Features – *abstract data types, polymorphism, control structures.*

General Terms

Design, Performance, Experimentation.

Keywords

Software architecture, Ubiquitous computing, Usability, User requirement, Participatory Design.

1. INTRODUCTION

Design and development of software architectures for ubiquitous systems have been a major concern in academic research and industry [1] and how architectures impact *real use* and *usability* have also become issue of research interest. Usability benefits have been widely applied to individuals performing on a desktop computer but need now to be re-examined within the context of distributed, interactive, networked and embedded applications. Usability studies, which traditionally approach aspects specific to a given task or application, have to be reinterpreted and adapted to Ubiquitous Computing application systems, wherein networks of laptops, PDAs, wearable computers, mobiles and other distributed devices are constructed, de-constructed and integrated.

Designers and developers must also find ways in which sensitive, responsive and intelligent UbiComp technology can also become *usable*, i.e. noticeable, comprehensible, adaptable and easy to control. That is why usage and usability concerns need to be reconsidered outside of the desktop metaphor. Achieving usability traditionally depended on how the functions provided by the system were understandable and clearly visible through the user interface. In this paradigm users have many input and output peripheral devices and the overall system *interface* must be adequate for their needs. There is a multitude of interfaces and usability issues for each mobile device of the distributed and ubiquitous system, and this requires a unique and enabling *software architecture* that must be designed according to users' needs.

In this paper we primarily discuss the interplay between software architecture development and users practices by focusing on the *architectural qualities* peculiarity of designing ubiquitous systems for users with special needs and diverse abilities through the case of Active Surfaces, a modular system of tiles used for play and therapy in water.

Active Surfaces relies on the service-oriented architecture developed in the EU funded IP PalCom, Palpable Computing [2]. We will discuss the interplay between users' practices and software architecture development by experimenting with the Active Surfaces with therapists and children with special needs.

By focusing on those attributes that support *palpable use* of technology, that we henceforward call *Qualities*, we also consider the architectural attributes required by *usable* ubiquitous technology.

2. ARCHITECTURE AND USE

The software architecture has been explored and experimented in different application prototypes related to the Health Care and the Landscape Architecture domains [2]. Each *general scenario* is characterized by an application prototype in which the architecture, or part of it, has been experimented. The application prototypes served as testbeds for the development of the

I-USED'08, September 24, 2008, Pisa, Italy

architecture and as case studies that provide the requirements coming from the field studies.

The *architectural qualities* have been introduced and described as the meeting point between architecture and use/application. The peculiarity of these non-standard architectural qualities is that they both evolve and gain meaning through software development and investigation of key *user practices* to account for.

In fact the *architectural qualities* and the *user practices* are tightly coupled and represent the two perspectives adopted in this research: the software architecture engineering and the interaction design perspective.

In order to better focus on users and architecture it is necessary to describe the Active Surfaces application prototype. The concept, the system and the architecture are described below.

2.1 Active Surfaces

Active Surfaces is a modular system constituted by physical and interactive units, the tiles. They are interactive modules that *activate* the surfaces of the swimming pool by making the environment featured with a network of distributed interactive components [3][4]. In particular, the prototype as developed in this research affords the horizontal configuration of the tiles on the water surface.

Figure 1. The current Active Surfaces prototype

The tiles constitute a network of physical (and software) objects that communicate and exchange data. Each Active Surfaces tile is thought of as a modular unit that can communicate with the others through its six sides. These entirely homogeneous devices, the tiles - which have exactly the same physical characteristics and computation and communication resources - are assembled. Each tile is an independent, physical, tangible object that can be picked up and moved around, and the interaction between the tiles is coherent and straightforward: all the tiles can communicate with their adjacent neighbours. They are, in fact, able to recognize their relative position as being essentially positioned and orientated in a sequence of tiles.

The Active Surfaces is highly scalable in respect to computational power and number of components. In fact it can scale up or down (vertically) by adding or removing resources to a single node in a system, typically involving the addition or removal of CPUs or memory to a single tile. Active Surfaces can also scale out (horizontally) by the addition of more nodes to a system, such as adding new tiles to the distributed system.

The concept emphasizes issues related both to the *use*, such as physical manipulation, positioning and emergent uses of the system, and the *architectural platform*, like the networking and dynamic assembly of tiles that is configured purposely [3][4].

3. SPECIAL NEEDS AND USERS' PRACTICES

In order to better focus on use practices as they emerge in the Active Surface application prototype it is thus necessary to describe the target users profiles - that is, the therapists and caregivers together with the disabled children - their needs, wishes and abilities [4].

Together with the study of the domain and a survey of the enabling technologies [5], fieldwork has been carried out with the aim of directly exploring the field of therapeutic intervention in water. The fieldwork has been conducted in two settings for psychomotor therapy in water, the Disabled Children Parents Association, Siena and the D. Chiossone Institute in Genova. We adopted ethnographic methods - such as field observation and interviews - and design methods - such as user workshops and creative brainstorming. The ethnographic activities attempted to observe and reveal relevant issues related to the environment (the features of the water, the physical structure of the swimming pool), the actors (therapists, disabled children, parents), the tools (objects, toys and water noodles) and, above all, the activities (the procedures, the different phases, the practices). We have addressed the whole practice starting from the *planning*, entering the *activity* and proceeding with the *evaluation* phase [5].

We will exclusively focus here on the overall description of users' needs and therapists practices in order to understand the implications they have on software architecture development.

Figure 2. Playing domino like games with Active Surfaces

The main actors of this therapeutic setting are the children with special needs. Children with very diverse profiles actually benefit from therapeutic play in the water. The users we have observed can be summarized in three main groups described below:

Autistic Spectrum Disorders and Other Affective and Socio-Relational Disturbances. People with autism have impaired social interaction and social communication and have a limited range of imaginative activities. People with autism have a tendency toward repetitive behaviour patterns and resistance to any change in routine. They need to be instructed and supported during the game, otherwise they very quickly return to their own solitary 'obsessive activities'.

Physical and Motor Disabilities and Cerebral Palsy. These children have limitation or an impossibility of movement, restrictions in force, abnormal postures, the presence of

neurological movement disorders such as dystonia, tremor, ataxia, etc. Children with cerebral palsy can be severely impaired in playing by their motor disability, but also by speech and communication disabilities, and sensory impairments (visual and/or hearing).

Mental Retardation/ Intellectual Disabilities/ Learning Disabilities. Children with mental retardation (also referred to as intellectual disabilities or learning disabilities, for example children with Down's syndrome), have a reduced capacity for attention and might not understand the meaning of the proposed activity. They might not understand the meaning of language and many of them have speech limitations too.

3.1 Key Practices

The therapists and trainers are the other main actors of this setting. They essentially have the role of facilitating the playful physical, social and emotional experience. They have to mediate the social relationships, the experience in the water and offer a reassuring presence to the child. They are the scaffolds that allow the child to express and freely explore the space of the pool. The therapists have to facilitate the activity, and not impose rules or, on the opposite extreme, abandon the child without a guide. Even when the child would like to explore by herself the therapist should also be present and support her independent action. The intervention is considered successful when the therapist interprets the meanings of the behaviors of the child. Having an intimate knowledge of the child is central to achieving this interpretation.

The outcomes of this activity resulted in key observations that have informed the whole design process. They can be summarized as follows:

Looking for creative solutions: The therapists usually deal with dynamic settings and changing conditions. This implies the ability to manage and rearrange the available resources in purposeful and creative ways.

Dynamic configuration of the tools: In dealing with continuously changing conditions and rehabilitation demands, the therapists should always find new solutions for adapting their tools and the environment to the patients and for maintaining their attention throughout the session. Consequently a core characteristic is that the tools have to be easily re-configurable and adaptable to this evolving situation.

Resource availability and opportunities for action: The therapist needs to feel in control of the available resources and how they might be adopted, changed and exploited. As in many workplaces, since their attention is exclusively directed to the patients, the resources the therapists use have to be ready at hand and immediately understandable.

Exploration and performance: This practice facilitates and encourages exploratory experimentation by users. Tools have to be used, customized and altered according to established degrees of freedom and constraints.

The key therapist practices are among the outcomes of the field exploration of the application sites and have continuously informed the development of the software architecture.

4. RESEARCH METHODOLOGY

Dealing with diverse and special users requires that methods and experimental environments would be appropriate, i.e. non-obtrusive, able to be personalized, adaptable, and capable of anticipating emerging user needs [6].

A wide variety of methods have been used throughout the iterative design life cycle [5]. These methods pertain to Human Computer Interaction, Participatory Design and Software Architecture Engineering. In particular we integrated a participatory design perspective with a co-evolutionary approach to interaction design and we explored this methodology in the domain of software architecture design. The process is co-evolutionary since architectural development, site exploration, activity analysis and concept design have been carried out in parallel so that each path of the process can inform, without constraining, the others.

We especially highlight on how the use of scenarios helped the structuring of data gathered through activity analysis, the envisioning of the role and functionalities of the system, and the assessing and validating the envisioned solutions from an architectural perspective (see [7][8] for scenario-based evaluation methods).

Throughout this research the scenarios are used to step through the software architecture and to document the consequences of architectural solutions from a user perspective. Different kinds of scenarios drove the research process: Activity scenarios, Envisioning scenarios, Prototype scenarios and Qualities scenarios [5]. We will focus here on Activity and Qualities scenarios that better represent the dialogue between Application and Architecture.

Activity scenarios stem from the fieldwork and activity analysis. They are grounded and built on data collected with ethnographic observation and user research. Activity scenarios account for concrete use episodes and key practices. We used the Activity Scenarios to understand, as thoroughly as possible, what is relevant and appropriate in the specific domains of use, which in this case study was the therapeutic practice in the water. These issues have thus been evolved into user requirements that informed the definition of the envisioned solutions at the software architectural level.

The key User Practices also were the criteria to define the experimental plan with the architectural prototype and the evaluation framework. In fact in this research *experimental architectural prototypes* have been used to conduct experiment on the architectural qualities that we have analyzed, in particular those observable at run-time (like performance) [9]. The experimental architectural prototypes allowed concrete measurements to be made under a range of different situations that might be also defined in terms of Qualities scenarios. They will be described in Par. 6.1.

Qualities scenarios consist of a slight adaptation of the quality attribute scenarios [1][10] that are a way to make the Qualities for palpable systems operational. They are short technical scenarios referred to specific Qualities. Qualities scenarios provide a way to concretely measure whether the architecture fulfils the requirements of the scenario. It states measurable properties of an architecture by defining metrics to be used in performance testing of the architecture. These scenarios allowed us to experiment with and evaluate specific features of the technology by testing the Qualities of the software architecture.

5. ARCHITECTURAL QUALITIES

In the multiple iterative cycles of the process followed in this research, scenarios have been used to bridge the use practices and the architectural development. The key practices have been discussed in terms of system use and from the software architecture perspective. The Architectural Qualities are summarized below:

USERS PRACTICES	*ARCHITECTURAL QUALITIES*
Looking for creative solutions	Assemblability
Dynamic configuration of the tools	Adaptability
Resource availability and opportunities for action	Resource Awareness
Exploration and performance	Experimentability

Table 1. From Users Practices to Architectural Qualities

Each Quality comes from an iterative design and development in which user participation and technological challenges were interwoven strands of the whole process.

Assemblability. Each Active Surfaces tile is identical and interchangeable and can run any piece of code that is passed to it through a neighbour, included the game logics. They can be assembled in many different formations that take into account the tiles' communication capabilities and the surfaces on which they have to be placed. Each formation of tiles is instantiated as a functional and physical Assembly of devices and services. The Assembly takes form as the users construct it by means of the Assembler Tile. The Assembly can then be dynamically altered and adapted over time. Despite the stability it has when it is created, the Assemblies can be easily deconstructed and re-constructed in a different formation being supported by flexible ad-hoc networks that can be controlled and configured by end users.

Adaptability. The Active Surfaces system consists of a set of tiny, resource constrained computers that can be arranged together to create a physical network. Because the tiles can only communicate with their close neighbours, there is an explicit and consistent discovery and communication framework underpinning the whole system. The tiles can be arranged in three-dimensional patterns, like squares in a crossword puzzle, and tiles, which are stacked one on top of the other, communicate through the top and the bottom. The network can be easily reconfigured by picking up a tile and moving it; this movement immediately changes the feedback that is provided.

Resource Awareness. The tiles are embedded systems with powerful and limited resources at the same time, such as available energy, available memory or communication bandwidth. Because of the limitations of these devices they represent a concrete challenge for the developers of the software architecture. In Active Surfaces a game application can exist within a network, rather than on a single unit or a central mainframe. Through the networking among the tiles and the instantiation of the assembly, they can discover the resources present in the system and debug the behaviour of such resources in order to overlook malfunctions or degraded individual or generalized performance. The resources are monitored and managed throughout time.

Experimentability. Active Surfaces can be thought of as a toy problem to experiment the software architecture because of its peculiar characteristics, as a modular system made of small easy to handle units. The tiles can be experimented with and tested without altering the structure of the system or causing any malfunctions or error. Indeed, Active Surfaces has to operate even despite the presence of an error in the use. An error is a condition of exception resulting from some deviation from the expected behaviour, which leads to a fault or failure, and the design of the architecture aims at minimizing the eventual adverse consequences of accidental or unintended actions.

These Qualities should not be considered in isolation, but rather as interwoven contributory factors that exhibit dependencies and influences on one another. The purpose of the Qualities is to capture the essence of what defines the nature of usable, easily perceivable and understandable (in a word, *palpable*) ubiquitous computing applications.

6. EXPERIMENTING WITH THE SOFTWARE ARCHITECTURE

The goal of this experimental phase is to describe the behaviour of the Active Surfaces system by measuring the performance of the architectural prototypes. The Qualities scenarios help in describing the performance in terms of more informative detailed statements. These statements allow *quantifiable arguments* about a system to be made [10].

Empirical testing is possible when relevant requirements and architectural components have been identified and prototypes have been developed. In particular the Active Surfaces architectural prototypes, described in the following paragraph, were used to observe, explore and evaluate the Architectural Qualities.

6.1 Architectural Prototypes

Prototypes of software components with different levels of accuracy and completeness have been used throughout the process. Their usage in architectural development provided the opportunity to have intermediate embodiments of the systems' functionality even if not supposed to represent any final or complete stage.

The Active Surfaces system underwent a concurrent development either within the Simulation Framework and the Hardware Platform. The hardware platform selected for the Embedded Architectural prototype is the UNC20 microcontroller. With such small microprocessor only the PalVM, the Virtual Machine developed within the PalCom project [2], is supported as a runtime engine.

The embedded architectural prototype has been built to learn about the PalVM platform and the serial communication over IR. The testing aims at discriminating whether there are restrictions in the PalCom open architecture or if the constraints are due to the current hardware implementation (e.g IR communication implemented over serial port).

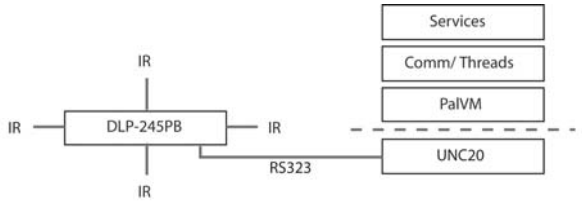

Figure 3. PalCom tile stack

The middleware management layer, which consists of managers handling resources, services, assemblies, and contingencies, requires too great a memory footprint to fit into the 8MB memory of the UNC20. Therefore, the software for the tiles has been developed to run on a standard PC with simulated infrared communication in concurrence with the development of the hardware for the tiles and the optimization of the middleware management layer. On the desktop machine the simulated framework runs on top of Sun's JavaVM.

The tiles deployed as simulated devices on a desktop machine are expected to have an optimal performance and can still exhibit a certain level of experimentability through the simulated game with a graphical user interface. In fact the therapists had the valuable opportunity to exploit the opportunities provided by the middleware managers, even if within the simulation framework The architecture experienced on the Simulated Framework was likely to inform the development of the embedded applications.

6.2 Performance Testing

The Performance Testing have been organized around tasks designed in order to translate the Qualities, and therefore with a relation to the Users' Practices, in measures observable via execution. The tasks aim at demonstrating how the existing architectural components would behave in performing the Active Surfaces scenario, e.g. performing the assigned activities.

The performance testing is based on a user-oriented perspective and assumes human practice in the therapeutic setting. In particular time responses, delays or frequency of errors have been observed with respect to the requirements coming from the activity analysis. For what regards timeliness, the major requirements from the therapeutic activity in the water are the duration of the whole session (45 minutes), the pace of the interaction (cycles of 3 to 5 minutes games to the utmost) intervened by the restless time pauses (2-3 minutes). These data allowed us to define the baseline for the experiments [5].

In order to determine whether there are restrictions in the software architecture or if the eventual constraints are due to the current hardware implementation, we have organized testing around two different conditions: 1) Tasks in which the performance is influenced mainly by the software architecture currently running; 2) Tasks in which the performance is both influenced by the architecture and mostly by the current hardware implementation [10].

In particular the experimental tasks can be grouped into the following areas. Each area represents a way to translate the Architectural Qualities (in brackets) into less conceptual and more verifiable evaluation tasks.

Communication and Discovery (Assemblability and Resource Awareness)

Task (a), (a1): 1+1 tiles, one is still, the other is rotated to reach the correct orientation for the side connection. In one case (a) Two tiles are put together, in the other (a1) two correctly connected tiles are kept apart.

Task (b), (b1): 1+2 tiles, one is still, the other two are rotated to reach the correct orientation at the same time. In (b) three tiles are put together, in (b1) three correctly connected tiles are kept apart.

Task (c), (c1): 1+3 tiles, one is still, the other three are rotated to reach the correct orientation at the same time. In (c) four tiles are put together, in (c1) four correctly connected tiles are kept apart.

The tasks are designed as two series each consisting of 10 repetitions of the tasks. In the first series the tasks are interrupted by re-boot of the game services (Re-boot series), in the other series the tasks are carried out continuously over time (Over time series). The former case represent the normal performance the tiles have on these tasks. The latter evidences how the performance in these specific tests varies over time.

Re-configuration (Adaptability)

The tiles currently can run either *fixed* GameServices, like the Jigsaw Puzzle Fish game (see Figure 1) and the Domino game (see Figure 2); or *open* GameServices where the tiles are in programming mode and learn how to configure by physical programming-by-example. The tiles also run FeedbackServices, like the actual LEDService or the possible VibrationService and SoundService that can be developed in the future.

The Re-Configuration tasks can either mean: choosing among existing pre-defined GameServices or the flexible use of single services related to game configuration, e.g. tiles' sequence, sensing and feedback.

In one case the system should allow shifting between pre-defined GameServices, i.e. different games that have already been configured. In the second case the system should allow running more services at the same time

That's why we launched different services in parallel simulating the two conditions described above. We are able to compare the task under two different conditions represented by the *ist.palcom.tiles.test.fish.prc* services, which involves IR communication among the tiles; and *ist.palcom.tiles.test.timer.prc* which doesn't involve the use of IR communication.

Performance (Experimentability)

Performance comprises 1 task performed under both the experimental conditions, *with* and *without* the use of communication. Thus there is a set of 2 tasks that consist of observing two GameServices running for 30 min.

As mentioned above, the overall session lasts 45 minutes and the duration of a single game situation can be assumed to be 30 minutes at the very most. In fact even if it is possible that children find some games very engaging, it is very hard to carry out the same game for almost the whole session. Furthermore game dynamics usually last few minutes.

7. RESULTS

In this paragraph a short summary of the gathered data is presented. For an extensive overview of the results see [5].

The results related to Communication and Discovery are presented regarding the two series of gathered data (Re-boot and

Over Time series), the two main actions (Put Together and Put Apart) and the scalability factor represented by the number of tiles utilized (2, 3 or 4 tiles).

Conditions	*Tasks*	*2 Tiles*	*3 Tiles*	*4 Tiles*
Re-Boot	Put together	3.2	7	6.9
	Put apart	7.6	9.8	12.5
Over Time	Put together	3.5	7.6	7.5
	Put apart	8.1	10.6	13.3

Table 2. Communication and Discovery. Summary of Results

The comparison among *Communication and Discovery* between 2 Tiles, among 3 Tiles and 4 Tiles, also gives a quantitative measure of how horizontal scalability affects the performance of the tiles system. Active Surfaces is conceived and designed as a modular system that in future implementation will be made of 12 units. The experimental data suggest that the performance of PalVM and the PalCom Communication components should be improved to meet the requirements of a highly scalable system and guarantee acceptable time responses as the number of the modules increase.

Tasks related to *Re-configuration* show how the system supports several services running in parallel and also creative combinations and adaptations of the tiles system. This can be done by shifting among these pre-defined solutions or by flexibly combining single services related to game configuration (e.g. game logics, sensing and feedback).

The eventual shifting among GameServices would be affected by the time required by new services to start, about 10 sec. As we observed through the activity analysis, the pace of the activity in the Active Surfaces scenario would impose a quicker response time for the re-configuration of the system. It is estimated to be no more than 10 sec in order to really provide the user with the experience of ready-at-hand tools. The results show that there is still not adequate support for the multiple services combination, i.e. more than three services running).

Regarding the combination of services, all the VM versions well support two services running in tandem both in tasks involving the use of IR communication or not. Simultaneously running two services, the system coherently exhibits the behaviours defined by the two services. Three services running in parallel are also supported but it seems to affect the behaviour of the tiles by decreasing the overall performance of the PalVM-release. These results are close to what happen with running one service alone and this could prove valuable support for re-configuration.

Tasks regarding the *Performance* over long periods of time show that the current implementation restricts the overall performance of the tiles. In fact, the performance through the LightUp GameService proved to be optimal, while tasks involving the communication modules resulted in a series of malfunctions that negatively affected the overall performance.

The results of the experiments allowed us to revise and elaborate on the initial formulation of the Qualities. For the Architectural Qualities revised see [5].

8. CONCLUSION

In discussing software architecture development and users' practices we have described the integration among the traditional ethnographic studies, participatory design methods and naturalistic experiments to inspire, inform and evaluate the design of software architectures [9]. This approach has already been adopted for the design of ubiquitous computing technologies [11] while it seems to be still fully appreciated in software architecture design [11].

Recently there has been a growing interest in understanding specific evaluation problems that arise from the use of Ubiquitous Computing systems [12]. In such paradigm software and hardware resources are distributed throughout the physical world and this impacts individual and social behaviours. Different evaluation criteria have been outlined, user attention (focus and overhead), the adoption of the system (value and availability) and the qualities of the interaction (physically embeddedness, dynamic input/ output, multiple devices, multiple users). Criteria related to the use and the person, such as understanding, control, accuracy, appropriateness, and customization, are also discussed.

This study helped us to figure out the complexity of such intricate stage where persons and computational resources influence one each other. With this research we wanted to highlight on multifaceted aspects interwoven in the *interplay* between real use and software development.

We observed that the introduction of UbiComp technology affected and changed users' activities and that, at the same time; they became responsible for maintaining, controlling and changing it. The system *architecture* / *use* relationship is dialectical since on one hand, technology enhance certain practices by enabling novel use opportunities, on the other hand user-specific dynamics provoke, inspire and inform the emergence of unpredicted architectural solutions.

In this paper we showed how such interplay took place through the whole research process, i.e. through design and development strategies that accounted for the special needs of the involved users and challenged the development of the system architecture. We wanted to give a feeling of this multiplexed process by describing the design of the experiments and the results. Data gathered during the activity analysis and activity modeling provided the backbone to define the experimental plan and the baseline for the evaluation of the system.

We empirically investigated the dialogue between user studies and software development by means of operative choices. We tried to bridge these two different fields and to take advantage of the methods of each domain. This study also resulted in the investigation of newly emergent interwoven processes that make *use* and *architecture* meeting at the edge, where software Qualities and Users' Practices juxtapose and evolve tightly coupled.

9. ACKNOWLEDGMENTS

Thanks to Prof. Patrizia Marti, Alessia Rullo and Erik Grönvall as they were a part of the research group for the duration of the PalCom project. Thanks to the collegues at the Computer Science Dpt, Univeristy of Aarhus that played a fundemental role in the software architecture development.

10. REFERENCES

[1] John, B. E.; Bass, L. (2001) *Usability and software architecture*. In Behaviour and Information Technology, 20 (5) pp. 329-338

[2] Palcom Project Website, http://www.ist-palcom.org/

[3] Grönvall, E., Marti, P., Pollini, A., Rullo, A. (2006) *Active surfaces: a novel concept for end user composition*, NordiCHI 2006, Oslo, Norway, 14-18 October, 2006.

[4] Pollini A., Grönvall E. (2006) *Constructing assemblies for purposeful interactions*. In Proceedings of Mobile Interaction in the Real World Workshop, MUIA06 at MobileHCI 2006, 8th International Conference on Human Computer Interaction with Mobile Devices and Services. 12 September 2006. Espoo, Finland.

[5] Pollini A., (2008) *Experimenting with an Ubiquitous Computing Open Architecture.* Ph.D. Thesis, University of Florence, Italy, 2008.

[6] Emiliani, P. L., Stephanidis, C. (2005) *Universal access to ambient intelligence environments: opportunities and challenges for people with disabilities*. IBM Syst. J. 44, 3 (Aug. 2005), 605-619.

[7] Kazman, R., Barbacci, M., Klein, M., Carriere, S. J., Woods, S. G. (1999) *Experience with Performing Architecture Tradeoff Analysis*, In proc. of the 1999 International Conference on Software Engineering, pp. 54-63, 1999.

[8] Bengtsson, PO. (2002) *Architecture-Level Modifiability Analysis.* ISBN: 91-7295-007-2, Blekinge Institute of Technology, Dissertation Series No 2002-2, 2002.

[9] Bardram, J. E., Christensen, H. B., and Hansen, K. M. (2004) *Architectural Prototyping: An Approach for Grounding Architectural Design and Learning*. In Proc. 4th Working IEEE/IFIP Conference on Software Architecture, pp. 15-24, 2004.

[10] Bass, L., John, B. E. (2003) *Linking usability to software architecture patterns through general scenarios*. Journal of Systems and Software, 66 (3), 187-197.

[11] Edwards, W. K.; Bellotti, V.; Dey, A. K.; Newman, M. (2003) Stuck in the Middle: The challenges of user-centered design and evaluation for infrastructure. ACM Conference on Human Factors in Computing Systems (CHI 2003); 2003 April 5-10; Fort Lauderdale; FL. NY: ACM; 2003; 297-304

[12] Scholtz, J., Consolvo, S. (2004) Toward a Framework for Evaluating Ubiquitous Computing Applications. IEEE Pervasive Computing Magazine, Vol. 3, No. 2 (Apr-Jun 2004), pp. 82-8.