Proceedings of the 2012 Workshop on Ambient Intelligence Infrastructures (WAmIi)

CS-Report 13-05

Proceedings of the 2012 Workshop on Ambient Intelligence Infrastructures (WAmIi)

T. Ozcelebi, A. Weffers - Albu and J.J. Lukkien

Technische Universiteit Eindhoven
Department of Mathematics and Computer Science

Proceedings of the 2012 Workshop on Ambient Intelligence Infrastructures (WAmIi)

T. Ozcelebi, A. Weffers - Albu and J.J. Lukkien

13/05

ISSN 0926-4515

Reports are available at:
http://library.tue.nl/catalog/TUEPublication.csp?Language=dut&Type=ComputerScienceReports&Sort=Author&level=1 and
http://library.tue.nl/catalog/TUEPublication.csp?Language=dut&Type=ComputerScienceReports&Sort=Year&Level=1

Computer Science Reports 13-05
Eindhoven, May 2013

Proceedings of the 2012 Workshop on

Ambient Intelligence Infrastructures

(WAmIi 2012) held in conjunction with AmI 2012, Pisa, Italy

Editors:

Tanir Ozcelebi, Alina Weffers-Albu, Johan J. Lukkien

PREFACE

This is a technical report including the papers presented at the Workshop on Ambient Intelligence Infrastructures (WAmIi) that took place in conjunction with the International Joint Conference on Ambient Intelligence (AmI) in Pisa, Italy on November 13, 2012.

The motivation for organizing the workshop was the wish to learn from past experience on Ambient Intelligence systems, and in particular, on the lessons learned on the system architecture of such systems. A significant number of European projects and other research have been performed, often with the goal of developing AmI technology to showcase AmI scenarios. We believe that for AmI to become further successfully accepted the system architecture is essential.

The workshop was held with 20-25 participants. A total of eight presenters, all of which are renowned for their work on Ambient Intelligence Infrastructures, presented the details of various local and international projects in the field and the lessons learned. The workshop program and the links to the individual presentations are given in the table below.

Presentation by	Affiliation	Topic	Projects
Maddy Janse (pdf)	Philips Research	AMIGO in hindsight - lessons learned	AMIGO
Marcus Ständer (pdf)	Technische Universität Darmstadt	Towards context-aware user guidance in smart environments	SmartProducts
Juha-Pekka Soininen (pdf)	VTT Finland	Opening embedded information of devices for intelligent applications	SOFIA
Monique Hendriks (pdf)	Philips Research	Inter-usability & intelligent communication: usability aspects in a multi device personal attentive system	SMARCOS
Boris de Ruyter (pdf)	Philips Research	Nomadic Media ITEA0219	Nomadic Media
Dietwig Lowet (pdf)	Philips Innovation Group Research	Florence - A multipurpose robotic platform to support elderly at home	Florence
Berardina De Carolis (pdf)	Univesity of Bari	A multiagent system providing situation-aware services in a smart environment	
Joelle Coutaz (pdf)	University of Grenoble, Immotronic	Infrastructure and architectural principles for plastic user interfaces	CAMELEON and ANR CONTINUUM
Closing by the workshop chair Johan Lukkien (pdf)	Eindhoven University of Technology	Summary and conclusions	

In this report, the reader will find the technical challenges faced in different Ambient Intelligence projects and the specific approaches taken. Researchers that aim to build AmI infrastructures can compare these and make use of this information to apply the most suitable methods.

Tanir Ozcelebi
Alina Weffers-Albu
Johan Lukkien

Amigo in Hindsight – Lessons Learned

Maddy D. Janse

Abstract—**This paper summarizes the experiences and the lessons learned from the Information Society Technologies (IST) Framework Programme 6 (FP6) project Amigo, finalized in 2008. Amigo provides a platform and building blocks for technologically advanced home environments that support all networked devices and services in the home using a service-oriented architecture, Web services and protocols such as Universal Plug and Play (UPnP).**

Index Terms—**Amigo, intelligent home environments, lessons learned, user-centered approach**

I. Introduction

MANY European and national research projects have addressed Ambient Intelligence infrastructures and their potential for developing applications for, among others, home automation, e-accessibility, e-inclusion, ambient assisted living, tele-care, personal health care, institutional health care, energy consumption. However, most of these applications exploit a subset of the available research experiences and results and leave the overall infrastructures as is. We might then question what are the hurdles for AI infrastructures? What are the lessons we can learn from the last 10 years of research in this domain? Although the Amigo project ended more than 4 years ago, the experiences and lessons learned during the course of the project and in the aftermath are still quite valid and provide insights for projects and applications in the AI domain. These experiences and lessons will be presented after a short introduction of the Amigo project.

M. D. Janse is with Philips Research in Eindhoven, The Netherlands. (e-mail: maddy.janse@philips.com).

II. Amigo in a Nutshell

The Amigo project started in September 2004 and ended in February 2008. The full name of the project is: Ambient Intelligence in the Networked Home Environment. The project was a collaboration of 15 European companies, research institutes and universities, partly funded by the European Commission in the 6th Framework Programme (Integrated Project-IST-2004-004182). Philips Research was the project coordinator. The major challenges addressed by the Amigo project were:

- To develop a consistent interoperable framework for the networked home environment that could be extended across different homes and locations
- To demonstrate compelling and sensible end-user applications for different usage domains

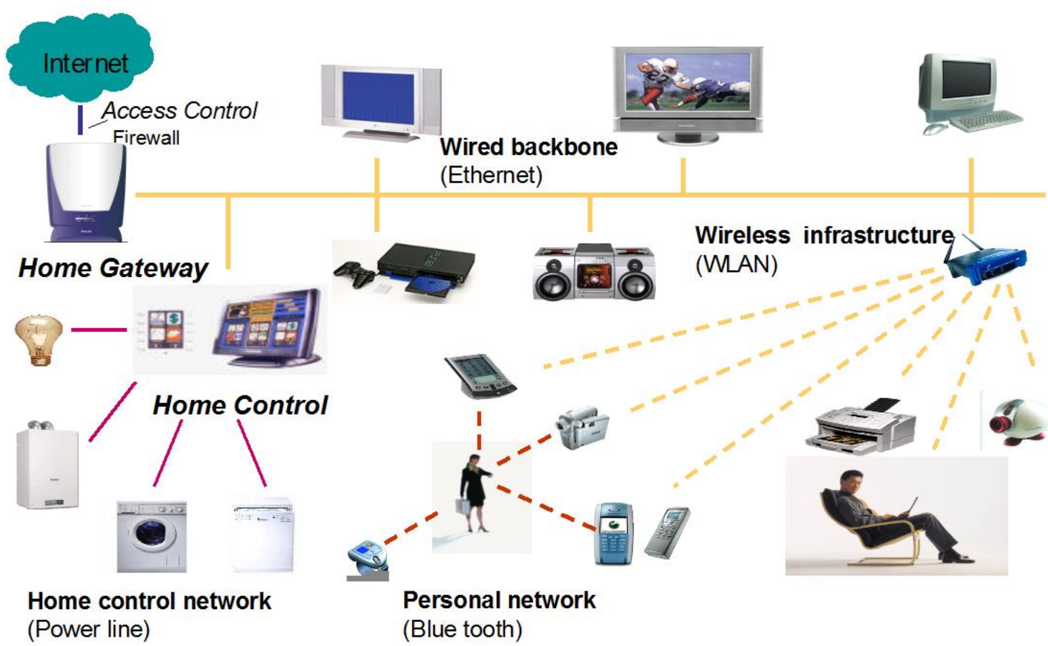

Fig. 1. Amigo networked home environment.

Fig. 1 shows the Amigo envisioned networked home environment.

At the start of the Amigo project, home networking had already emerged in specific applications such as PC to PC communication and home entertainment systems, but its ability to really change people's lives was inhibited by the lack of interoperability between different manufacturer's equipment, complex installation procedures, and the scarcity of compelling user services. These problems were addressed by developing middleware that dynamically integrates heterogeneous systems, adopting a user-centered development approach, and developing applications in different domains that show the potential for end-users and the benefits of the service oriented-middleware architecture for application developers. The Amigo interoperable middleware dynamically integrates services and devices. For example, home appliances, such as, heating and lighting systems, washing machines and refrigerators, multimedia players and personal devices, like mobile phones and PDA's were connected in the home network to work in an interoperable way. This interoperability across different application domains was also extended across different homes and locations. The Amigo architecture contains a base middleware layer, an intelligent user services layer, Amigo-aware applications, and a programming and deployment framework. The middleware layer and the user services layer provide the functionalities needed for a networked environment and an ambient in-home network, respectively. Amigo-aware applications and services form the top-layer of the architecture, and the programming and deployment framework allows developers to create applications and services (Fig. 2). The interoperable middleware operates across different application domains and across different homes and environments. This flexibility of the architecture ensures that the system can grow, as and when new devices and applications are added. Furthermore, the Amigo software is open source, which encourages further development of the system.

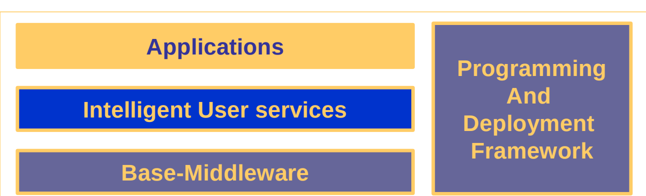

Fig. 2. Amigo Architecture

However, living in such a connected environment is very difficult to perceive and imagine for users, let alone, that they can explicitly specify their requirements and express their appreciation for such an intelligent environment. Furthermore, user requirements will change and evolve over time as people are becoming more familiar with connected environments. To facilitate working with the dynamics of such evolving user requirements the project adopted a user-centered development approach. Central to this approach was the development of a usage scenario to present the potential functionalities and implementations to users and developers.

Amigo applications were developed for three different domains: "Home Care and Safety", "Home Information and Entertainment", and the "Extended Home Environment" – in which multiple homes are connected. Different applications were developed for each of these domains. For example, a comfort management system that maintains environmental conditions that are adapted to user profiles, the different zones in the home and the time of day. Health management is another example of an integrated service that offers people in-home health monitoring and coaching. The integrated demonstrators for the Home Information and Entertainment domain use standard protocols that are widely used, like Wi-Fi, Ethernet or UPnP. Most of these applications are web-based, i.e., any device with a web browser can connect to the Amigo network and users can easily interact with the home devices. The Extended Home Environment applications provide end-users with services that enable them to share activities and experiences in an easy and personalized way. For example, parents who are on a business trip can still share daily activities with their children at home, tell them their bedtime stories, watch TV together, look at pictures or play a game with them. That is, they can share their presence independent of location and devices, for example, using TV with PC, TV with TV, or mobile with TV. Using a personal device, for example a mobile, in somebody else's home network for using the services in one's own home is another possibility. Such a device enables users to access services which are operating in their own home from other domains, for example, a friend's home, cinema, or office. This computing device travels with the user and 'binds' a visited domain to a home domain.

III. Lessons Learned

A. Tame technology

There is a huge difference in perception between user needs, user requirements, system requirements, system specifications and their ultimate implementation resulting in a system that fits all these perceptions. This is not a new problem, but the challenge is immense as the number of variables causes an exploratory explosion of problems.

> *"What is complex to users is also complex for developers of the system."*

B. Identify the target users and beneficiaries

In first instance, the project aimed at end-users/consumers as the primary beneficiaries. However, to make home systems much more attractive for end-users, the benefits for the end-user of a combined home system must be attractive and evident and offer a surplus over what is offered by current non-networked systems. Most of the Amigo services use knowledge of the world around the device like the other devices in the system and the user. This is only possible in a networked system. Such new services for the end-user are, for example, using a display that is in the neighborhood to display information instead of on a small PDA display, using the processing capabilities of the home server to do speech recognition and communicating the results back to the camera that has only small processing capabilities, i.e., combining user-related and context-related information. Such examples, albeit interesting, are not exemplars that get people in a normal household very excited.

> *"A towering engineering achievement is not equal to a killer application. Why would a consumer talk to his/her washing machine?"*

C. Adapt to changing application demands

In home automation and domotica people start to look more and more to Amigo-like results, not so much driven by the possible attractiveness of connected devices as envisioned by the project, but driven by, for example, energy consumption, sustainability concerns, and demographic changes, e.g. homes for the elderly. The project mainly adapted to these challenges by putting more emphasis in the dissemination activities on these applications that are particular relevant for these evolving consumer needs and showing how the Amigo interoperable middleware addresses these challenges. In due course, the project changed its policy and aimed at designers and developers of services and applications as the main target users of the Amigo software and applications. And, furthermore, at third-party developers in:

1) telecommunication, multimedia, informatics and consumer electronics convergence,
2) home automation and security, such as energy suppliers, household appliances, security and surveillance enterprises or health and assistance service providers, and
3) Internet service providers and entertainment companies.

> "*Timely, relevant and easy to understand application examples are a must."*

D. Address the needs of the application and system developers

Several applications developed on the Amigo platform show designers and developers, the advantages of using the Amigo software, of having the advantage to use the tools that they are most familiar with, of being able to build applications very quickly by just using a subset of the Amigo platform and by having the possibility to gradually extend the applications by exploiting the flexibility of the system. These benefits apply for application development by and for SME's and their customers, as well as for larger industrial organizations. The Amigo platform is very suitable to quickly build and show application possibilities to customers, to support the development of roadmaps and to customize for clients. The architecture is such that it ensures flexibility of the system to grow, as and when new devices and applications are added.

> *"Understand the context of work of application developers. They are not going to change their way of work if there isn't anything in for them."*

E. Entice stakeholders

Amigo provides a common framework for the networked home that ensures interoperability and opens a path to the home's inside for different stakeholders. It is an opportunity for the development of a new generation of telecommunication services that gravitate around the home and that integrate service providers and operators and home networks in a trend that could be named "from the PC, to the TV to the laundry". These services can become accessible from external networks through the Amigo services. This is not so innovative, but it is difficult to position the Amigo benefits in this context as the application domains are complex, overlapping, complementary and competing. For example, viable application domains for Amigo are, among others: Smart homes - new applications and services; Mobile services and content provision; Consumer electronics and Lifestyle – resource-constrained networked devices, gateways, dynamic updating of software on remotely managed networked embedded devices, remote building of automation services; Health and wellbeing - ambient assistance services, elderly care; Collaborative work and communication – sharing experiences, content, work

> *"Easy, understandable and enticing application examples are needed in all domains for impact."*

F. Technology moves on

The Amigo project started in 2004 with the technologies and expectations of that time. Furthermore, the preparations for the project started in 2003. Technology has advanced enormously since then. These technologies, for example sensor network technologies, could have been used differently in the project. Furthermore, the surge of Web 2.0 technologies that are now often used for social networking applications, just like many applications developed in Amigo are related to social networking. To stay up to speed with regard to such advances in technology and its applications, the project needs to be aware of the relevance of these developments for its activities and to adapt if appropriate and feasible. The Amigo application work packages started using such services as example to illustrate the adaptability and flexibility of the system. Other activities, not foreseen in the original proposal, were started, for example, the communication framework. For example, the choice of .NET Web interfaces was a risky decision in 2004. But, it is now very much at the forefront with regard to the devices' market.

> *"If you can't stay abreast of technology, change your strategy according to the times."*

G. Dissemination of the overall concepts and results

With regard to the overall dissemination of the project, the R&D expectations of 2003/2004 focusing on customer benefits of futuristic applications needed to be adjusted for the community of application development, emphasizing the modularity of the Amigo system and its flexibility in supporting the incremental development and integration of home automation applications. Both middleware and application elements appear to have different time to market. The influence and role of the existing infrastructures, i.e., including bricks and power lines, needs to be addressed and adapted for each potential stakeholder community. It should be noted that software production for the home, as Amigo propagates, is cutting edge with regard to current practice in the build environment.

“Starting from ‘scratch’ is only possible in Utopia.”

H. Take-up of technology is not about technology

The feedback given by the participants of the Amigo dedicated training program for SMEs provided very useful feedback with regard to current practices, state of the art in the application domain and obstacles to take-up. That is, SMEs current technology is typically “years behind” of the Amigo approaches. They are looking for out-of-the-box solutions and are not convinced that Amigo will survive as a whole. But, they see elements of the application prototypes easily being taken up. This is not a matter of technology, but of market demands, trends, and politics, and of distribution and installation channels. However, Amigo was perceived by the SME community as setting the tone for demonstrating innovative concepts and ideas and as a source for innovative components and solutions.

“Respect the hardships of crossing from research pastures to the wilds of business implementation.”

I. End-user acceptation and feedback

Most feedback of end- users was generated by means of the evaluation of the initial application scenarios and the final integrated prototypes. This feedback is not based on long-term usage in people’s homes but on presenting and interacting with the prototypes. In general people liked the applications related to health and wellness, and home automation. However, whatever the feedback of the end-users, it is always dependent on how and to what extent they can trust the applications, how reliable, robust and trustworthy the system is perceived. It is important for users to have similar or universal interfaces for all the applications. Furthermore, the logging of user data, content, and context are extremely sensitive issues that need to be addressed for any subsequent implementation or follow-up research. End-users don’t consider the different Amigo domains as separate domains. For them, it is much more important to maintain the overall comfort and social integrity of their home environment.

“Interoperability is obvious and natural for users.”

J. Sharing ambience and activities

The concept of ambience and activity sharing that was exemplified in the Amigo extended home applications extends the traditional views of remote communication between people and between locations. The take-up of this concept in home applications will require heavy investment in devices and it is a long-term perspective, but it constitutes very suitable applications for the broadband home. Potential short term applications could aim at, for example, informal communication in office environments, remote care, and remote assistance.

Digital sharing of ambience and activities is not the same as sharing pictures and text

References

[1] M. D. Janse, Amigo Final Project Report, ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/st-ds/far-v3_en.pdf

Towards Context-Aware User Guidance in Smart Environments

Marcus Ständer, Aristotelis Hadjakos, and Steven Luitjens

***Abstract*—Context-aware guidance systems need to observe and react to the user even if she does not always follow the instructions exactly. A human guide can easily adapt to such situations while a computer guide needs a model how to guide the user, has to work with unreliable sensor data to observe the user, and, in the case of multiple active processes, needs to assign the right sensor information to the right process. In this paper, we introduce two use cases for smart environments and use them to motivate unsolved challenges in the field of context-aware user guidance that we observed in a set of user experiments.**

***Index Terms*—Context awareness, Interactive systems, Ubiquitous computing, Computer aided instruction**

IV. Introduction

The saying “practice makes perfect” expresses, that humans are not born as masters but have to learn how to do things. It has always been the case that humans with more knowledge try to transfer it by either directly teaching others or by noting down their knowledge in order to save it and reach a broader community. Executing practical exercises usually follows a certain strategy: the teacher tells what to do, the student executes, and the teacher controls and comments. As it is neither possible to be a master in every discipline nor to always have a personal master available, written instructions are an important means to transfer knowledge. Written instructions like manuals or recipes are present everywhere. In manufacturing, for example, an engineer plans a machine and technicians have to build it based on the given instructions. Later, service technicians can maintain the machine based on maintenance-instructions. Another example everyone knows are recipes. Not everyone is a chef but one maybe wants to prepare a delicious meal. Thus, having a recipe that instructs which ingredient to use how and when is a good means to reach that goal even at the first try.

The question how *computer-assisted guidance* (CAG) can be supported and enhanced has been under research since the early days of computing. Between 1960 and 1980 especially computer assisted career guidance laid the foundation for the development of modern CAG systems [1] by using their knowledge to propose optimal decisions. Since these days CAG has been becoming important for many disciplines. It is no longer used for educational purposes only; the scope has been extended especially to ensure the quality of some work, for example in surgery [2], [3].

Following Moore’s law, the capacities of computing hardware evolved rapidly and computers become faster and smaller. Today users employ computers in merely every sector of their life, e.g. for getting access to modern communication systems, by enacting financial transactions on the stock market, or even by letting them control airplanes, buildings, or even infrastructures of whole cities. Together with computing hardware evolved sensor technology. Modern smartphones for example usually contain two cameras, a three axis accelerometer, and gyroscope, a proximity sensor, a compass, a barometer, a microphone, and even more. Miniaturization of sensors allows us to equip everyday tools like knifes with accelerometers to detect the kind of movement and thus, the activity that is performed with the knife (cutting, chopping, etc.). We refer to the information that represents the world state as *context* and the notification about changes in the world state as *events*. The context generated by the sensors informs the guide if and how the user performs the work so that the CAG can dynamically react, which we call *context-aware user guidance*. If a user is instructed to execute a task and she finishes it, the guide can automatically show the next instruction. If the user uses machine B instead the proposed machine A, the guide can adapt its description or notify the user if there would be disadvantages.

In the vision of ubiquitous computing, the combination of computation units and the multitude of sensors are seamlessly integrated into our environments, which we then refer to as *smart environments*. These smart environments allow CAG systems to instruct, sense, and react to the human user.

The motivation for computer-assisted context-aware guidance is manifold and comprises mainly the support of (I) learning or at least following new processes, (II) guaranteeing the quality of work done and (III) supporting users in tasks they cannot execute themselves completely but where a human user is required. This is for example the case when only partial automation is possible or when a responsible person needs to be present.

V. Use-Cases

In this paper, we focus on two smart environments of special importance for GAG: the smart factory and the smart

Manuscript received September 4, 2012.

Marcus Ständer is with Telecooperation Lab, Technische Universität Darmstadt, Germany (email: staender@tk.informatik.tu-darmstadt.de)

Aristotelis Hadjakos is with Telecooperation Lab, Technische Universität Darmstadt, Germany (email: telis@tk.informatik.tu-darmstadt.de)

Steven Luitjens is with Philips Group Innovation, Research, High Tech Campus, Netherlands (email: steven.luitjens@philips.com)

home environment.

A. Smart Factory

Nowadays production can basically be sorted into one of the two categories: (i) standard production or (ii) highly individual production. Often also a combination of the two types is required like e.g. in aircraft manufacturing. Highly standardized components like engines, air conditioners, or aircraft bodies need to be combined, following the wishes of the customer. Interviewing experts from EADS, we found that in nowadays production sites technicians are guided by paper instructions and checklists where they have to sign e.g. that they have mounted certain screws with a certain torque. This kind of traceability is of high importance, especially in critical production sites. If an airplane runs into technical problems, it is important to be able to track back the roots of the problem.

In a smart factory with context-aware guidance, printed manuals can be replaced by electronic versions. Technicians then always have access to the latest required manuals. Making use of context allows automatically displaying the right instruction to the technician and switching to the next instruction when the activity is finished. Furthermore it is possible to pre-configure special tools automatically for the next activity, e.g. to configure the maximum torque of an electronic wrench. Also traceability is enhanced by collecting auditing information automatically, which is more reliable than manual input.

In consequence, technicians can work more efficiently, production gets more reliable and traceable, and managers always have an overview about the real-time state of the production. This allows spotting possible production problems with low management costs.

B. Smart Home Environments

A second type of a smart environment is much more user-oriented: the *smart home environment*. The goal of such environments is usually to support users with activities of daily live [4], [5], handling the rising complexity of the environment [6–9], or monitoring inhabitants [10] if they could otherwise not live alone any longer. Thus, the target group ranges from young people over the average family to elderly people.

Smart home environments offer a diversity of interesting research areas, e.g. ease interaction with the environment due to constantly rising complexity of multimedia electronics [11], or performing activity recognition to control e.g. that inhabitants with the Alzheimer’s disease do not forget to take their medicaments [12].

As we focus on context-aware user guidance, we have chosen the kitchen as our main sub use-case. A large set of related work is focused on the kitchen [13–21] as in many homes cooking a nice and healthy meal is an activity which is carried out regularly. A smart kitchen environment could help consumers to have an enjoyable experience in preparing a healthy meal. Using suitable recipes and healthy ingredients could help to prevent obesity and coronary disease. In that way a healthier lifestyle would be enabled. As the kitchen also hosts many different appliances, it is also a good scenario for providing guidance in the sense of technical manuals, e.g. for descaling a coffee machine.

C. Example

In order to point out and illustrate the gaps in research about context-aware user guidance in smart environments, we have chosen a rather simple multiuser example: Two users prepare a salad and a raspberry ice with chocolate in parallel. The salad is a cucumber salad and the recipe contains the following activities: Wash cucumbers, slice cucumbers, add cucumbers, herbs, and dressing to a bowl, and finally mix the salad. The recipe for the ice contains these activities: heat raspberries, cut chocolate pieces, add vanilla ice to plate, and finally add raspberries and chocolate pieces to the plate.

VI. Challenges

A. The Problem of Modeling and Learning

In literature, there exist two categories of approaches that try to cope with dynamic processes: *model-based* and *learner-based* approaches. Model-based approaches provide explicit process models that have to be created by developers, thus, they have the benefit of being instantly applicable. Learner-based approaches first have to learn the process model. They need large training sets to learn processes and the outcome is a priori unknown. A clear disadvantage of model-based approaches is the limited flexibility, as designers need to foresee all possible situations and reflect them by means of making the model more detailed and complicated to understand. However, even if learning processes would work perfectly, the learned process would highly depend on the profile of the user and the sensor environment. If another user has different habits and executes the process in another way or if different sensors are used, this will require additional learning. And especially for user guidance, a clearly defined user interface (UI) is required. However, generating a suitable UI for learned processes is an open issue. Thus, neither modeling nor learning seems to be the perfect fit for the problem of describing user guidance. Instead a modeling approach with some characteristics of the learning approach, like allowing the user to deviate from the model to some degree, could be promising.

B. Coping with User Actions

Coming back to the teacher explaining a student how to perform a certain task, probably everyone made the experience that we sometimes do not exactly follow the teachers’ instructions. Often this is caused by misunderstandings and sometimes also by different opinions but usually the master reacts and corrects or discusses the deviation.

Having a guidance system, users tend to act very similar. The problem here is that the guidance system can just not react as dynamic as the teacher could. In a small experiment where we observed how users would use an interactive, context-aware guidance system, groups of two users prepared the salad and raspberry ice, using a guidance system that showed step-by-step instructions. We observed that users tend to read more than one instruction before they start to work. Especially the more experienced users started to execute the process differently then proposed by the system, e.g. they cut cucumbers earlier than expected. We observed the following situations that caused problems in recognizing correct process states.

Firstly, users deviated from the proposed solution. One reason for the deviation was simply that the proposed solution was just not possible, e.g. if there was no salad bowl, users decided to use a cooking pot. A more interesting reason for deviations is of psychological type. If users think that a different solution is better suitable for them, e.g. more efficient or easier to perform, they adapt the proposed process. In the example this could be washing and cutting the herbs first, as they are for example already lying on the chopping board. Another example most readers will know are navigation systems. A navigation system sometimes proposes a route, which the driver will not follow, even if it is optimal. This happens because she thinks her choice is better or if she just does not like one of the streets on the proposed route. March's theory about human decision taking [24] explains such a behavior as a matter of thinking about alternatives, raising expectations, obeying user preferences and finally applying decision rules. Depending on the outcome, users decide for one or the other alternative.

Apart from user deviations, the second reason for problems in detecting the current process state is related to the **activity** recognition and the sensors themselves. To date activity recognition is a highly probabilistic process so that sometimes activities are labeled wrongly. Sensor signals from e.g. cutting and slicing might be very similar and when the wrong activity is reported, the guidance system might not react to that situation so that further progress is not possible without human intervention (e.g. manually acknowledging that this activity has been performed). Finally it is always possible that hardware problems like malicious sensors or temporal network problems might cause context to not be propagated properly.

C. Recognizing Concurrent Processes

In the previous section we have introduced the problem of one user who does not follow the instructions exactly or whose actions are not recognized correctly. Considering the case of several users raises even more new problems: (1) assignment of events to processes and (2) assignment of events to users. Coming back to the two users preparing the salad and the raspberry ice, problem (1) occurs if both are using the guidance system in parallel, each one executing her own process. When the classifier recognizes an activity, how can it know to which process it belongs without further information? In literature (e.g. [25]), it is often assumed that this assignment is given but in real smart environments with a multitude of publicly usable utensils and appliances, this assumption cannot hold. Also problem (2) is not solved easily. Some approaches in literature implicitly solve the assignment of events to users, e.g. by using body-worn sensors. However, often sensors are connected to utensils like a knife, which can be used by every user in the kitchen. And even if we would equip users with technology like a body worn RFID reader and attach RFID sensors to the utensils, it would not be clear which user really works on which process. In our initial study we observed, that one user was doing the cutting for the other user, just because she had the knife in her hand anyways.

Thus, the guide must try to find the most probable explanation for the events. It needs to examine the following possibilities: (1) It is unclear if the event belongs to one of the processes at all, thus, ignoring it must be possible. (2) If more than one process would expect that kind of event the guide should assign the event to the process with the highest probability. However, this probabilistic assignment leads directly to an optimization problem. The local optimum that is chosen by the guide does not necessarily correspond to the global optimum. In the current example this case can be observed if the system notices that the raspberries are heated. Now *"cutting"* is recognized and assigned to the ice making process. This is more probable as for the cucumber salad the cucumbers need to be washed first. However, then the system senses, that cucumbers, herbs and dressing is added to the bowl. Now at this point, the system needs to recalculate the probabilities and might figure out that this cutting activity might have belonged to the salad process and washing the cucumbers has either just not taken place or has not been sensed. In essence, neither the assignment of events to processes or to users nor users to processes can always be assumed to be known in a real smart environment with different inhabitants.

VII. Contributions of SmartProducts

In the scope of the SmartProducts project, we created a platform for realizing smart environments[1]. The platform contains a multitude of components, e.g. to guide the recipe choice based on personal requirements and preferences [22], to keep track of the kind and weight of ingredients with a special scale [23], and many others that can be found on the project website[2]. However, the most important component for this paper is the SmartProducts user guidance system.

In the scope of the project, we introduced *XPDL4USE* [26], a lightweight process description language that allows adding semantic information to processes. These annotations are e.g. used for describing which appliances are required for an activity or what the purpose of a process is, e.g. if it represents the recipe for a salad. Further we introduced a method for allowing developers to choose from a continuum between simple but static processes and more sophisticated but harder to design planner methods, depending on the degree of runtime flexibility they need [27]. Regarding the dynamicity of the environment, we are able to semantically select suitable processes from the set of processes that are available in the environment and use context to influence the control flow of active processes. Based on the given processes, we also allow deviations that follow a certain pattern, e.g. when a different

[1] http://www.smartproducts-project.eu/mainpage/smartproducts-platform
[2] http://www.smartproducts-project.eu

execution order was detected, the system tries to reorder the activities. Multiple processes and multiple users are in principle possible but only if the context can be clearly assigned to a certain process or user. **Table 1** shows a summary about which of the most important guidance features are supported by the SmartProducts platform. *"X"* stands for *"addressed"*, *"O"* for *"partially addressed"* and *"–"* for *"not (yet) addressed"*.

Model			Dynamicity			Multiple Processes		
Semantics	Planning	Learning	Runtime Selection	Context Usage	Probability Model	Multiple Processes	Multiple Users	Probability Model
X	O	-	X	X	O	O	O	-

Table 1: Overview about the guidance features supported by the SmartProducts platform

VIII. Conclusion

In this paper we have motivated context-aware user guidance. We have introduced our two main use-cases. Based on an example we have pointed out challenges that still have not been solved in research. It is unclear how guidance processes should be created, how to best cope with not perfect users and environments and how a computer guide can differentiate between contexts of different processes caused by the same environment. In SmartProducts we made some progress but there are still open challenges. In our future work we want to focus on the missing parts, especially regarding multi-process recognition, trying to define a methodology for better context-aware user guidance for smart environments.

IX. Acknowledgment

This research was partly funded within the SmartProducts project funded as part of the Seventh Framework Programme of the EU (grant number 231204).

References

[1] M. R. Katz and L. Shatkin, "Computer-Assisted Guidance: Concepts and Practices.," 1980.

[2] R. Mischkowski and M. Zinser, "Comparison of static and dynamic computer-assisted guidance methods in implantology.," *International journal*, 2006.

[3] R. Haynes, "Computer-aided quality assurance: a critical appraisal," *Archives of internal medicine*, 1987.

[4] P. Rashidi, D. Cook, and L. Holder, "Discovering activities to recognize and track in a smart environment," *IEEE Transactions on*, vol. 23, no. 4, pp. 527–539, Jan. 2010.

[5] E. M. Tapia, S. S. Intille, and K. Larson, "Activity Recognition in the Home Using Simple and Ubiquitous Sensors," *Sensors (Peterborough, NH)*, pp. 158–175, 2004.

[6] M. Hellenschmidt and R. Wichert, "Goal-oriented assistance in ambient intelligence," in *Workshop on Experience Research in Ambient Intelligence. Eindhoven, The Netherlands*, 2005, no. June, pp. 1–5.

[7] K. Richter and M. Hellenschmidt, "Interacting with the ambience: Multimodal interaction and ambient intelligence," *Interaction*, Jun-2004. .

[8] M. Hellenschmidt and T. Kirste, "SodaPop: A software infrastructure supporting self-organization in intelligent environments," in *2nd IEEE International Conference on Industrial Informatics, 2004. INDIN '04. 2004*, 2004, pp. 479–486.

[9] T. Kirste, "DynAmite - Dynamisch Adaptive Multimodale IT-Ensembles," in *Tagungsband: Forschungsoffensive" Software Engineering 2006*, 2007, p. 63.

[10] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, "Accurate activity recognition in a home setting," *Proceedings of the 10th international conference on Ubiquitous computing - UbiComp '08*, p. 1, 2008.

[11] M. Hellenschmidt and T. Kirste, "A generic topology for ambient intelligence," *Ambient Intelligence*, pp. 112–123, 2004.

[12] G. Singla, D. J. Cook, and M. Schmitter-Edgecombe, "Recognizing independent and joint activities among multiple residents in smart environments," *Journal of ambient intelligence and humanized computing*, vol. 1, no. 1, pp. 57–63, Mar. 2010.

[13] M. Kranz, P. Holleis, and A. Schmidt, "Embedded Interaction: Interacting with the Internet of Things," *IEEE Internet Computing*, vol. 14, no. 2, pp. 46–53, Mar. 2010.

[14] M. Kranz, A. Schmidt, A. Maldonado, R. B. Rusu, M. Beetz, B. Hörnler, and G. Rigoll, "Context-aware kitchen utilities," in *Proceedings of the 1st international conference on Tangible and embedded interaction - TEI '07*, 2007, pp. 213–214.

[15] D. J. Patterson, D. Fox, and H. Kautz, "Fine-grained activity recognition by aggregating abstract object usage," in *Ninth IEEE International Symposium on Wearable Computers, 2005. Proceedings.*, 2005, pp. 44–51.

[16] C. Pham and P. Olivier, "Slice\&Dice: Recognizing Food Preparation Activities Using Embedded Accelerometers," *Ambient Intelligence: European Conference, Ami 2009, Salzburg, Austria, November 18-21, 2009. Proceedings*, p. 34, 2009.

[17] J. Chen, P. Chi, and H. Chu, "A Smart Kitchen for Nutrition-Aware Cooking," *Pervasive Computing*, pp. 58–65, 2010.

[18] G. Panger, "Kinect in the kitchen: testing depth camera interactions in practical home environments," *ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 1985–1990, 2012.

[19] L. Bonanni and C. Lee, "CounterIntelligence : Augmented Reality Kitchen," in *Proc. CHI*, 2005, pp. 2239–2245.

[20] W. Ju, R. Hurwitz, and T. Judd, "CounterActive : An Interactive Cookbook for the Kitchen Counter CiH12001," in *CHI'01 extended abstracts on*, 2001, pp. 269–270.

[21] M. Beetz, F. Stulp, and B. Radig, "The Assistive Kitchen — A Demonstration Scenario for Cognitive Technical Systems," *RO-MAN 2008. The*, pp. 1–8, 2008.

[22] Y. van Pinxteren, G. Geleijnse, and P. Kamsteeg, "Deriving a recipe similarity measure for recommending healthful meals," 2011.

[23] Toet, E., B. Meerbeek, and J. Hoonhout, "Supporting mindful eating: InBalance chopping board," in *Workshop on Promoting and Supporting Healthy Living by Design*, 2011, pp. 49–52.

[24] J. March and C. Heath, "A primer on decision making: How decisions happen," 1994.

[25] C. P. Kunze, S. Zaplata, and W. Lamersdorf, "Mobile process description and execution," in *Proceedings of the 6th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2006)*, 2006, vol. 4025, pp. 32 – 47.

[26] P. Webster, V. Uren, and M. Ständer, "Shaken not Stirred: Mixing Semantics into XPDL," *5th International Workshop on Semantic Business*, 2010.

[27] M. Staender, A. Hadjakos, and D. Schreiber, "Adaptive Workflows in Smart Environments : Combining Imperative and Declarative Models," in *DOMe-IoT 2012 aligned with Ubicomp 2012*, 2012, p. 4.

Opening Embedded Information of Devices for Intelligent Applications

Juha-Pekka Soininen

Abstract—**Developing intelligent applications creating added value in particular physical places would benefit from having access to information and services existing in embedded systems and devices around. This paper presents a brief summary of M3 local information sharing solution targeted for embedded systems and discusses its characteristics based on use experiences in various projects. The approach extends the sematic web oriented approaches to embedded domain and has potential to become an Internet-like infrastructure solution enabling mash-up applications based on embedded systems.**

Index Terms—**Semantic interoperability, smart space, open data, embedded systems**

I. Introduction

During last decades the research community has been struggling to develop methods and tools for adding intelligence into our environment, in physical places we live. There have been various goals from automating routines, modifying the conditions, saving resources, to supporting our living, for example. From technology perspective this has meant improvements in the areas of sensing, analyzing sensor data, minimizing the cost and resource consumption of computation, and in understanding the contexts and intentions of people in places in question..

The visions of ambient intelligence, ubiquitous computing and smart homes, buildings or cities, i.e. smart environment ecosystem, have not been implemented in a large scale in spite of all possibilities. The problem is the lack of interoperability of devices and appliances. In most current products the aim has been in fulfilling the specific purpose of the product and the additional cost due to interoperability has been considered too high. The optimization for the purpose has also led the huge diversity of communication and service level solutions. Partly this is also due to the need to protect the businesses of companies with proprietary solutions. The smart environment extending to all possible domains is also too big challenge for any single company to handle.

Standard interfaces such as DLNA have been useful in specific domains. The problem is the standardization processes are slow resulting to incompatibility problems with evolving technology and new features and services. Another approach to conquer diversity problem is to use gateways and servers for providing communication and processing capabilities. To limited number of interfaces this is feasible, but it may also introduce new partners into the value chain of the end-user's service. It may also be a major challenge for privacy and security.

The problem in the emergence of smart environment ecosystem is how to provide information and services from currently used devices for new mash-up based intelligent applications with minimal interference to current devices. This involves both technical and business related challenges. In this paper we will give a brief introduction to a solution called M3 that allows sharing the embedded information of devices [1] and summarize the experiences on its use in various demonstrations and pilots [2, 3]. The key idea in M3 is to have a semantic information sharing services for opening the information for other devices without restricting the purpose of use of the information. This extends the ideas of semantic web [4] into embedded system domain.

The paper is structured as follows. In section II the vision of interoperable devices is described. Section III briefly presents the M3 concept. Section IV summarize our experiences in the use of M3 and discusses its main characteristics. Sections V give the conclusions.

II. Vision of Interoperable Devices

The vision behind M3 is to provide best information, services and resources for applications in a particular local physical place through collaboration of devices capable for computation and communication in that place. This would enable the optimization of applications against desired quality characteristics such as performance, energy consumption or user experience, and extension of applications by means of increased capabilities to exploit available information and resources.

The vision is to have a solution that can expand across domains, platforms, languages, vendors, and business ecosystems. The idea is to be able to use information and services from all devices in the physical place regardless from their original purpose or origin. This would allow application that mash-up capabilities from multiples domains and devices is a same way that done in the Internet today. So the M3 would act as glue and interface between various service-

Manuscript received September 4, 2012.

This work has been done in Artemis Sofia project supported by EU and national funding organizations of partner countries. in Finnish national DIEM, OPUTE, and MIOTE projects supported by Tivit and Tekes, in EU FP7 project iCore.

J-P. Soininen is with VTT Technical Research Centre of Finland, (e-mail: Juha-Pekka.Soininen@vtt.fi)

oriented architecture solutions, business ecosystems, and application domains. Examples of this vision could be home control systems exploiting user information from mobile devices, forecasts and energy cost from Internet, activity information from other home appliances or media systems that uses media sources both in home and mobile devices, user preferences, lights, TVs, and entertainments systems.

III. M3 SOLUTION

The M3 solution is basically a simple semantic information sharing service that can be embedded into different service networks and that can provide simultaneously multiple interfaces to service networks. With the support for implementing additional functionalities and for development of applications it extends to a platform for intelligent applications in physical places.

The separation of interoperability concerns in one the key issues in M3 concept. In order the create interoperability between devices we have to able to handle three levels. First, we have to transmit bits between devices requiring common physical layer solutions. Secondly, we have to be able to access services across device boundaries which require shared communication and message protocols. Thirdly, we have to understand the meaning of information in the same way at both ends.

The real world is extremely complex. The variety of communication and service protocol solutions is enormous. Therefore in M3 we decided to focus on the third level only and provide possibility to use existing solutions at lower levels. In M3 the common understanding of information is achieved using common ontology model based semantic presentations of information. Approach is the same as in semantic web, i.e. to model the information as an RDF graph with subject, predicate, and object triples with references to underlying ontology models.

A. *Design guidelines*

The main principle of M3 is to focus on opening and sharing of information. It is a blackboard software architecture with a publish/subscribe capabilities. Applications can subscribe to information and they are alerted when changes or updates happen. Second main issue is that the interoperability agreement between participating devices (or device manufacturers) is done at the information level. Companies have to agree only on sharing the information using common ontology and semantic web based uniform format through M3 semantic information broker (SIB) service to be able enable intelligent applications using their devices.

Other development guidelines have been to keep the idea and implementation as simple as possible, so that the cost of implementing the service or interfaces to service is low. The service itself is kept agnostic with respect to ontology, application programming language, service platform, communication layer, and hosting devices/systems. The idea has been that M3 implementations can support simultaneously multiple intelligent applications spanning across multiple domains and business ecosystems.

Real-life applications require a lot of supporting features related to evolvability, security, trust, privacy, scalability, and quality characteristics in general. The complete list of design principles can be found in [2].

B. *Logical architecture for information sharing*

The logical architecture of M3 based system is very simple (Fig. 1). The core is semantic information broker (SIB) service that implements interfaces to desired SOA solutions which then have their own physical communication interfaces. The SIB can be accessed using smart space access protocol (SSAP) that contains a set of simple functions such as join, leave, insert, remove, query, and subscribe for manipulation of information graph. The SIB service is essentially a RDF database and depending on the implementation it can support different query languages. Recently the SPARQL has been most popular.

The applications or services that access the SIB are called knowledge processors (KP). They have to implement the SSAP protocol and conversion of information to uniform RDF triple presentation. When common ontology models as a basis of information are used in several KPs the interoperability and common understanding of information becomes possible.

C. *M3 based systems*

M3 concept is a very simple approach for enabling only the information sharing between applications, services, and devices. It is based on using existing, legacy service frameworks and communication solutions to support complete system construction. The SSAP protocol has an authentication and security capabilities, and they can be extended with external solutions.

The complete system requires additional supporting functionalities. Examples are elimination of obsolete information in database, automatic additions of information to published information, e.g. time, source, context awareness, reliability of information, run-time adaptations, etc. The solution in all of these in M3 is additional KPs that augment or modify the stored RDF graph. Several solutions supporting system implementations have been reported in [2, 3].

The M3 concept does not limit the implementation architecture possibilities from M3 based systems also called smart space applications. The system can be built around a single SIB or it is possible distribute the shared information into several SIBs in case information ownership, mobility of devices, or some restrictions with communication capabilities require it. It is also possible to create a single search extend from several SIBs [2].

D. *Programming mash-up applications*

A key criterion for any system or application development method is the ease and simplicity of programming. In M3 the use of common model based uniform RFD format for shared information and the possibility to use dedicated methods needed for programming the devices are basic requirements.

The M3 offers a possibility to share information and services in a very flexible way resulting to different kind of applications and development flows. It also gives freedom for developer to decide whether to put the knowledge of ontology of application into the SIB or on KP side.

The KPs are in principle independent agents that voluntarily participate into overall functionality. So, we need to support devices own programing environments and focus on the interfacing through information that is supporting the publishing and accessing the information in SIB to/from device-specific applications. The creative part is then how the new user experiences emerge from environment through ad-hoc mash-up of available information. An approach focusing on hiding ontology models from programmers is in [6].

It is also possible to design complete systems so that all devices and sharing of information is designed at the same time. Due to the multi-device, multi-platform, multi-language nature of M3 heterogeneity of design methods and tools need to be supported. In this case the basic application development flow consists from creation or reuse of common ontology models needed for application, from the creation of APIs to support the RDF format from/to programs in the participating devices, and from designing the actual logic in all of them. The ontology model based design flow for M3 is presented in [2] and a more complete description for smart space development in [7].

IV. EXPERIENCES

There are three main reference implementations of M3 concept. Smart-M3 [5] is a Linux-based open source implementation available under BSD license from SourgeForge [8]. RDF Information Base Solution (RIBS) is targeted to be more portable and more suitable for low capacity devices [9]. RIBS also implement security solutions and a light-weight version of SSAP. Third implementation has been done into OSGi framework as a part of application development kit (ADK-SIB) [2].

M3 concept has been used in various pilots in Sofia project where devices from different domains interact through M3. Different ecosystems (car and mobile media) were linked through M3 for enhanced features in Media Follows user pilot. Complex operational processes were created involving devices and systems from different domains in Smart Maintenance, Smart Maintenance on Move, and Video Surveillance pilots. Information from various sources were combined and displayed in Virtual Wall and Virtual Graffiti pilots. M3 smart spaces were connected and technology mash-ups were created in SUM-S and Smart Home pilots. In all of these the use of common ontology models and creation of information based on them enabled the enrichment of functionalities that expand across domain and business boundaries. The application development was supported by KP library and ADK supporting the reuse. [2]

Another branch of M3 development has been to take semantic information processing also to low capacity devices. M3 SIB was in a WLAN access point where it acted as core of Green House control system [10]. The sensors were connected to M3 via gateways and the actors were designed so that they reacted on information in SIB instead of specific control interfaces. In [11] the idea was pushed even further. The semantic interface (KP) was implemented in active tag attached to a sensor. The active tag was capable to publish its own information in RIBS, but also capable to react on information published by others. M3 has been used in connecting tagging system with local information servers in DineTender application [12]. The idea was to create a very local smart place with a possibility to share data that is meant only for people in close proximity.

The basic feedback on experiences so far is that interoperability in diverse environment can be achieved using information sharing, developing mash-up applications is simple and easy, and that the idea of using semantic information scales from servers and desktops also to mobile devices and embedded systems, and in a restricted form also to ubiquitous computing and sensors.

The main benefit of M3 concept is the very low barrier in adoption of the technology. The reference implementation and the specifications of the SIB and SSAP are available as open source. They can be taken into use without any legal operations or negotiations thanks to BSD licensing model. The implementation of the SIB and KPs are straightforward. Making own versions with added features is relatively easy and there is on-going work in this area. The RDF format and ontology models are used in semantic web which means that there is lot of active R&D in this area and also mature results that can be exploited.

In case the M3 based system requires multiple companies the collaboration can be achieved with simple agreements. Agreement on common ontology model, how devices and appliances react to information, and what legacy systems are used in accessing the SIBs is enough. This is simpler than agreeing on service interfaces, protocols, and access to functions or resources. Ontology model based approach is also more flexible, since it leaves open the possibility to innovate new functionality. Since all actors are independent agents they can be enhanced without compromising the collaborative functionality. When there is no need to discuss about interfaces, the possibility to faster development and incremental improvement of systems in a mash-up way becomes a reality.

In the use cases developed the M3 proved to be a very flexible way of linking independent systems and devices together. The amount of new software needed in the participating devices was small. In pilot implementations even products of companies that aim for proprietary solutions were connected on application level to M3 and to collaboration with other business ecosystems (i.e. Apple, Nokia, Google for example). In some cases the incremental development and possibility to innovate new features to the system happened even unintentionally.

The Green House use case had several development cycles. Initially, it consisted of a wireless sensor network, manual control interface, and a couple actors. Later it evolved to an automatic control system. Tag-based automation was included, and finally web-based plant growing instructions and more intelligence were added. During the time the core ontology remained untouched and the initial devices were not modified at all. The intelligence was added through extension of ontology and through new KPs that manipulated the information. This kind of model gives new possibilities to industry. Network based design becomes more easy due to simpler interfacing. Responsibilities of systems functionalities can be given to companies that are truly focused on them. Eventually this will result to more optimal parts and improved overall system quality. It also gives room for easy adoption of new innovations.

There are restrictions in the M3 concept. Since the communication occurs at the information level, i.e. through the SIB, it is not suitable for hard real-time systems. Also the semantic format of information causes overhead of putting large chunks of data into the SIB. Large images, audio and video files must currently be transferred in other ways. In M3 based systems this is naturally possible, since it does not exclude the use of any service or physical level interoperability solutions.

The real M3 systems require additional functionalities that must be explicitly created. Solutions based on common middleware typically have a set of supporting functions that take care of reliability of data, security of system, clean-up of data base, etc. In M3 these responsibilities have been left to external KPs that have to be created. The motivation has been to maximize the flexibility.

The final challenge relates to ontologies. Ontology models can enable global interoperability, but they can also act as ways to protect businesses and to enable diversity. Ontologies also evolve with the world they present. Ontology governance is a major issue. When ontologies are used in embedded domain or in ubiquitous computing those parts cannot be easily modified after production, which must be taken account in the future.

V.CONCLUSIONS

Opening embedded information of devices with embedded systems and even ubiquitous computing devices gives new possibilities in creation of smart applications and in exploiting the Internet of Things ideas. The proposed M3 solution is a simple alternative. Its strength is in simplicity, portability, scalability, and flexibility. It is infrastructure solution that can develop to a real interoperability platform.

The success of M3 idea depends on how industry adopts the idea of opening embedded information. Information has value and open information idea is against the traditional business thinking that focuses in protecting everything. Open data and open information has some excellent success stories in the Internet. The question is that can this potential be seen also in embedded domain. Other challenges relate to ontology governance issues, industrial collaboration required to networking in embedded domain, and technical maturity and quality characteristics of SIB solutions for embedded devices. The experiences with M3 show that these are solvable issues.

ACKNOWLEDGMENT

The credit from the research results belong to the researchers in Artemis SOFIA project, Finnish DIEM, OPUTE, and MIOTE projects, and in EU FP7 iCore project.

REFERENCES

[1] Liuha, P., Soininen, J-P, Otalea, R. “SOFIA: Opening embedded information for smart applications”, Embedded Systems Conference, Nürnberg, Germany, 2010
[2] Sofia project. http://www.sofia-project.eu/
[3] DIEM project website, http://www.diem.fi/
[4] T. Berners-Lee, J. Hendler, O. Lassila, “The Semantic Web,” Scientific American Magazine, May 17, 2001.
[5] Honkola, J. et al, "Smart-M3 information sharing platform," IEEE Symposium on Computers and Communications (ISCC), 2010 , 22-25 June 2010, pp.1041-1046
[6] Kaustell, A. et al, , “Framework for Smart Space Application Development”. In: Salvatore Flavio Pileggi (Ed.), International Workshop on Semantic Interoperability (IWSI 2011), 10, SciTePress, 2011
[7] Ovaska,E., Salmon Cinotti, T., Toninelli, A. ” The design principles and practices of interoperable smart spaces”. Advanced design approaches to emerging software systems : principles, methodologies, and tools. Xiaodong Liu and Yang Li, editors., pp. 18-47, 2012
[8] Smart-M3 project. Smart-M3 www-pages at SourceForge. 2010, 05/07, Available: http://sourceforge.net/projects/smart-m3/.
[9] Suomalainen, J., Hyttinen, P., Tarvainen, P. “Secure information sharing between heterogeneous embedded devices”, ECSA’10 Proceedings of the Fourth European Conference on Software Architecture: Companion volume, ACM, NY, USA, 2010, pp. 205-212
[10] Kiljander, J. et al , "Opening information of low capacity embedded systems for Smart Spaces," *Intelligent Solutions in Embedded Systems (WISES), 2010 8th Workshop on* , vol., no., pp.23-28, 8-9 July 2010
[11] Ylisaukko-Oja, A, et al, “Semantic interface for resource constrained wireless sensors”. Proceedings of the International Conference on Knowledge Engineering and Ontology Development, KEOD 2011, 26 - 29 October 2011, pp. 505 - 511
[12] Open-M3 website, http://www.open-m3.org/

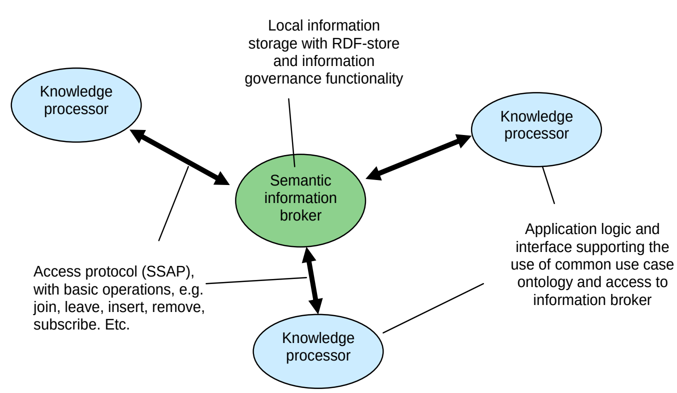

Fig. 1. Information level view to M3.

Inter-usability and Intelligent Communication: Usability Aspects in a Multidevice Personal Attentive System

Monique Hendriks, Ville Antila and Tine Lavrysen

***Abstract*—Ambient intelligent systems often consist of multiple, interconnected devices. Therefore, in designing the user interaction of such systems, one should ensure a seamless user experience across devices (inter-usability). In the SMARCOS project, a multi-device personal attentive system was designed to help users obtain and maintain a healthy lifestyle. In the design of the user interaction of this system, inter-usability aspects needed to be taken into account, but the system also needs to communicate with the user in an intelligent manner; it needs to provide advice and motivation to the user which is adapted to the user's personal situation. Moreover, besides the contents of each message, timing, choice of device and presentation modality (text, audio, or video) are of importance. We describe these two user interaction challenges and how they were dealt with in the design of a prototype for the personal attentive system.**

***Index Terms*—User interaction, interusability, embedded intelligence, context awareness, lifestyle coaching**

I. Introduction

THE SMARCOS project (Smart Composite Human-Computer Interfaces) is aimed at helping users of interconnected embedded systems by ensuring their inter-usability. New challenges have come up for user interaction: multiple platforms, multi-user applications, Internet synchronization, and application and service adaptation to the changing situational contexts. Inter-usability entails a seamless user experience across devices, platforms and situations.

The use case of the development of a personal attentive system in the SMARCOS project has provided insight into the design process of an intelligent, context aware coaching system. This use case was aimed at developing an intelligent system that motivates and supports healthy consumers and chronic patients in their daily life, while ensuring that these users experience inter-usability across devices. Two target groups were defined at a high level; office workers and diabetes type II patients in the early stages of the disease, who can still regulate glucose levels fairly well using pills and being physically active.

Users of the personal attentive system use a number of devices, among others an activity monitor, a pill dispenser, a glucose meter, a smartphone, a laptop and a TV. All of these devices provide input to the system; some of them contain sensors that measure the user's behavior (e.g. the activity monitor, the pill dispenser, the GPS of a smartphone), others provide usage information (whether the device is switched on, whether the user is currently interacting with the device, etc). A subset of these devices can also be used to communicate information to the user; not only screen-based devices, but also for example the activity monitor which contains lights that indicate how active the user has been.

The system uses these types of available information:

• The user's behavior as measured by the sensors (physical activity and medication intake)

• The user's manual input of glucose measurement values and food intake (which can for example be analyzed from pictures taken with the smartphone's camera)

• Situational information gathered from sensors such as the GPS in the smartphone, keyboard typing frequency from a laptop or pc, or simple usage information such as when a device was turned on, and when it was turned off

• Personal characteristics, either directly provided by the user, inferred from information already available on one of the devices, or learned by the system over time

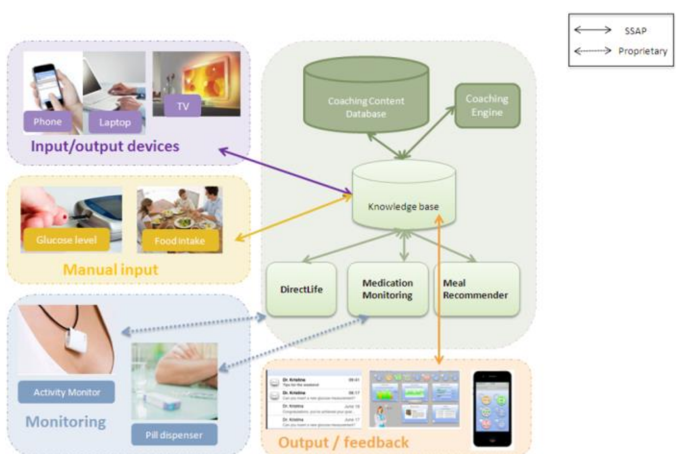

Fig. 1. The SMARCOS Personal Attentive System.

This information is used to provide assistance to the user in

Manuscript received September 4, 2012.

M. Hendriks is with Philips Research, Eindhoven, The Netherlands. E-mail: monique.hendriks@philips.com.

V. Antila is with VTT, Finland. E-mail:Ville.Antila@vtt.fi

T. Lavrysen is with Human Interface Group, Mechelen, Belgium. E- mail: tine.lavrysen@higroup.com

obtaining or maintaining a healthy lifestyle. The situational information is used to infer opportune times and opportune situations to send feedback on his behavior, practical tips and motivational messages to the user. Fig. 1 gives a graphical overview of the system. See [5] for an extensive description of the use cases described here and the corresponding requirements that preceded the design of this system.

The system described above presents two challenges in the area of user interaction:

• Inter-usability: Designing the interaction with multiple devices in such a way that the user experiences the system as a coherent whole and full use is made of the capabilities of each device.

• Intelligent communication: The user interaction does not only entail usage of the system's devices in a re- active sense, but also proactive communication of the system with the user via multiple devices. Moreover, this communication deals with a sensitive subject; the system provides feedback on the user's unhealthy lifestyle. This communication needs to have the right content, it needs to be delivered on the right device and at the right time (see [4], for evidence of the effect of timing of motivational messages).

Below, the approach that was taken to tackle these issues is described, for both the issues related to inter-usability in general and the issues related to the main task of the multi-device system: communicating with the user regarding his lifestyle.

II. Interusability

Attention should be paid to the fact that the user is interacting with a system that consists of multiple devices. These devices may have different input and output modalities (e.g. keyboard, touchscreen, audio, video, text, etc). Furthermore, the timing and situations in which the user may interact with each device, may differ per device (e.g. a mobile phone may be used while riding the bus, but a desktop PC will only be used at its fixed location, a pill dispenser may be used at work, but not while in a meeting, etc). Finally, how the user interacts with each device can also differ among these devices (e.g. a user will interact differently with a pill dispenser than with a mobile phone). Despite these differences though, the user should feel that he is interacting with one system.

It should be clear to the user

• what the capabilities of each device are and what functionalities are available on each device

• what data is available on each device

• what the role is of each device in the overarching system

• whether there is functional modularity in the system: is there a subset of devices that can still provide some limited service when specific devices are unavailable, and if so, what is this subset and what can it do?

• what the behavior of the system will be: how predictable is the behavior of the system on a certain device, is it comparable to other services on the same device, or to similar services on another device?

The ISO 924-11 guideline defines usability as a function of characteristics of the user, the user's tasks and goals and the environment (or context) in which a device is being used (see Fig. 2) [3]. Metrics were defined for measuring the extent of usability;

• Effectiveness: how many errors do users make when performing a task on the system, how quickly can they recover from errors?

• Learnability: how quickly can users learn to use (do basic tasks on) the system after first encounter?

• Efficiency: how quickly can users perform tasks once the learning phase is over?

• Memorability: after a period of non-use, how quickly do users regain their ability to use the system?

• Satisfaction: is it perceived as pleasant to use the system, or is it a burden?

These metrics can be reused as metrics for usability in a multi-device system. However, since the interaction between the different devices of the system raises new issues w.r.t. usability, these metrics need to be extended with guidelines w.r.t. the inter-usability issues listed above: the capabilities of each device, the availability of data, the role of each device, options for functional modularity and predictability of behavior on each device w.r.t. other services on that device.

In the SMARCOS project, guidelines for designing inter-usable system are currently still being composed. They cover the following aspects of the design of inter-usable systems [2], [1], [6]:

• Role of devices (Composition) - Within one system, different roles can be assigned to devices. Devices might have the role to gather specific data or provide certain functionalities that other devices are not capable of

• Modularity (Composition) - In order to offer a set of core and/or additional functionalities, a combination of multiple devices might be needed. By ensuring the modularity, the system can incorporate additional devices which contribute specific functionality to the system, but does not necessarily replicate all the existing functionalities as well

• Consistency - The system should be consistent across devices. The consistency can be perceptual, lexical, syntactic or semantic

• Knowledge Continuity - The knowledge continuity is ensured by using the common (platform-specific) metaphors and design guidelines consistently, but also use the lexical, syntactic and semantic consistency across the service

• Migration (Task continuity) - In a multi-device system the user hand-over becomes an important design issue. When the user switches from a device to another, the system should maintain the state of interaction and synchronize the content

• Context-adaptation (Task continuity) - Moreover, when a system is used in different contexts of use the system should adapt appropriately

When the SMARCOS Personal Attentive System is analyzed according to these aspects, these are the results:

• Role of devices - In the Personal Attentive System, different devices have different roles, taking into account their capabilities. For example, the activity monitor is dedicated to gathering activity data and the pill dispenser gathers medication intake data. Other devices have overlapping roles. Both a TV and a smartphone can give feedback about the current progress or communicate messages to the user.

• Modularity (Composition) - The Personal Attentive System is modular in that users can decide which set of core and/or additional functionalities they want to use. For example if they purchase an activity monitor, but no pill dispenser, they will not be able to receive feedback on their medication intake, but they will still be able to get feedback on their activity level.

• Consistency - While designing the different parts of the Personal Attentive System, lots of attention has been paid to aligning them and providing a holistic system. The terms that are used, interaction patterns, icons and graphics are all designed to be as consistent as possible.

• Knowledge Continuity - Through consistency in workflow and interaction patterns across devices, the predictability of system interactions is improved when users move from one device to another. This also increases the learnability.

• Migration (Task continuity) - In the Personal Attentive System, interactions occur in short bursts. Users check their progress or get a short message. As these tasks are short, we do not expect users to want to migrate from one device to another in the middle of it. Therefore currently no migration from one device to another is supported.

• Context-adaptation (Task continuity) - Based on the user's context, the Personal Attentive System adjusts the content of the messages sent, their modality and the device to which it is sent. This is described in more detail below, in the section "Intelligent Communication".

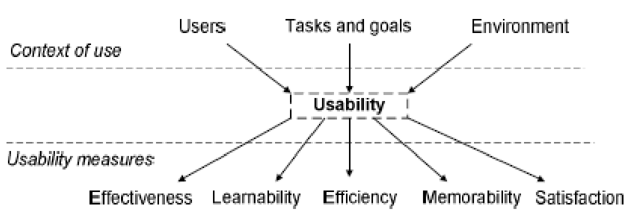

Fig. 2. Framework of usability, taken from D1.2 [3] of the SMARCOS project.

III. INTELLIGENT COMMUNICATION

Using multiple devices allows for gathering lots of data. However, this data gathering should be unobtrusive, it should not be a burden for the user. The devices should share their knowledge in the background. For this to happen, a number of steps need to be taken:

1. The data gathered by the devices should be translated to a common data format

2. The data should be transformed to useful knowledge about the user through some form of reasoning, e.g. GPS coordinates should be translated into a classification of a location such as "home", "office" or "supermarket"

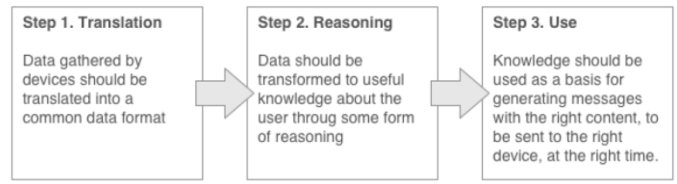

Fig. 3. The three steps of gathering knowledge.

3. This knowledge should be used as a basis for generating messages with the right content, to be sent to the right device, at the right time

See Fig. 3 for a graphical overview of these steps.

Since ambient intelligent systems are user centered, an important part of any such connected system, and therefore also of our personal attentive system, is user modeling. We have chosen to model user knowledge in the form of an ontology, which we annotated with information regarding which device is responsible for delivering specific data. The advantage of this model, is that in setting up the reasoning rules for step 2 and 3 together with domain experts, we can abstract away from which device delivers which data and in which format, and just assume that it is there. Furthermore, ontologies allow for ease of communication regarding the data model.

We have defined two types of reasoning rules: update rules and intervention rules. Update rules describe the reasoning that takes place in step 2. These rules specify the conditions under which a certain fact of knowledge may be assumed to be true for the user. For example, an update rule can describe the range of GPS coordinates that should be considered as the user's home. Intervention rules describe the reasoning that takes place in step 3. These rules specify which messages should be sent to the user, at what time and to which device.

From an online questionnaire conducted with 15 diabetics and 49 office workers on their preferences for receiving messages from the Personal Attentive System, we among others concluded the following:

• Respondents prefer to receive messages on their smartphone, computer or TV, but they are for example also open to receive a message like "You have been sitting for a long time, time for a walk?" through a color-changing lamp

• Respondents mostly preferred receiving messages when they were relaxing in front of the TV or having a break. Respondents least preferred receiving messages when they were in a hurry or when they were busy doing sports

• Overall there were plenty of differences between respondents, which indicates that for sending messages to users their specific preferences should be taken into account

Besides such user research, intervention rules need to be based on the knowledge of a domain expert; in this case a lifestyle coach. As described in the previous section, in choosing the right device for the message, its perceived role in the system, the user's expectations and the capabilities of the device should be taken into account. A user might for example not expect to have to read a message on the TV, so messages sent to the TV should preferably be in audio or video modality. Use of the mobile phone might be interesting because it provides the option to contact the user while he is on the move, but we should take into account that in certain situations, the user might not want to be disturbed by the system (e.g. when in a meeting).

The setup we have chosen allows system designers to start at the specification of messages and reason backwards toward the conditions for either sending or not sending certain messages and for the choice of a specific device. From these conditions one can then reason backwards toward the required data model and how this data could be obtained through sensors and usage information of the devices in the system. This design process therefore allows for heavy involvement of domain experts in coaching as well as interaction design from the start of the process. And so by design, we obtain a user friendly personal attentive system that gathers data unobtrusively, provides motivating and personalized advice tailored to the situation, and is polite and convenient by choosing the right method of communication (device and modality).

IV. Conclusion

The SMARCOS project has identified the need for developing key metrics for inter-usability, as an extension to the available metrics for usability in general. Existing metrics such as effectiveness, learnability, efficiency, memorability and satisfaction may be reused on a multi-device system, but they are not sufficient. User research to evaluate the concept of the Personal Attentive System described here, shows that in user interaction design of multi-device systems, there is indeed a need for paying attention to the different roles of each device in the system. Attention also has to be paid to possibilities for modular design such that devices may be included or excluded from the system, a consistent ‘look and feel’ across devices, consistency of knowledge across devices and adaptation of interaction patterns to the context of the user. The use case of the Personal Attentive system did not uncover the need for migration of tasks, but in situations where interaction patterns typically have a longer duration, one can imagine the need for task migration.

Through the analysis of the challenges in designing and developing the Personal Attentive System and other similar application concepts in SMARCOS project, we have identified a set of guidelines for designing inter-usable systems. Based on the important aspects of inter-usability described in this paper, these guidelines are targeted towards guiding the designers in making design decisions, such as: what should be considered when distributing of functionalities across devices in an interconnected system, or what should be taken into account when adding new functionalities or devices to the system. While these guidelines are not yet finished and will be discussed more in the future, we see them as an important starting point for identifying the key aspects in designing an inter-usable system as well as specifying metrics for evaluating inter-usability in addition to the already available and established metrics for usability in general.

Acknowledgements

This research was funded by the European Commission, within the framework of Artemis JU SP8 SMARCOS project 100249.

References

[1] Charles Denis and Laurent Karsenty. Inter-Usability of Multi-Device Systems A Conceptual Framework, pages 373–385. John Wiley & Sons, Ltd, 2005.

[2] T. Lavrysen et al. Smarcos deliverable d1.8: Analysis of ui design in hybrid embedded devices, interconnected and cloud-based service solutions. 2012. http://www.smarcos-project.eu/smarcosPublications.html.

[3] V. Antila et al. Smarcos deliverable d1.2: Prevailing ui design methodologies for interconnected embedded systems. 2010. http://www.smarcos-project.eu/smarcosPublications.html.

[4] S. van Dantzig, G. Geleijnse, and A.T. van Halteren. Toward a persuasive mobile application to reduce sedentary behavior. Journal on PERSONAL AND UBIQUITOUS COMPUTING, 2012.

[5] Aart van Halteren et al. Smarcos deliverable d4.2: Requirements and feedback models for healthy living. 2010. http://www.smarcos-project.eu/deliverables/D4.2 Requirements and Feddback models for healthy living.pdf.

[6] Minna Wäljas, Katarina Segerståhl, Kaisa Väänänen-Vainio-Mattila, and Harri Oinas-Kukkonen. Cross-platform service user experience: a field study and an initial framework. In Proceedings of the 12th international conference on Human computer interaction with mobile devices and services, MobileHCI ’10, pages 219–228, New York, NY, USA, 2010. ACM.

Nomadic Media
ITEA0219

Boris de Ruyter

***Abstract*—The quality of interaction that people enjoy when they are using networked services will be determined by the quality of the devices they use, and the content. The aim of the ITEA (Information Technology for European Advancement) project Nomadic Media is to reduce restrictions imposed by the platforms and devices that consumers will soon be using to access services and content in both physical and virtual home environments.**

***Index Terms*— Mobile device, Ambient Intelligence**

I. Introduction

THE vision that computers will 'disappear' and they will become totally embedded in the way we live is fast being realized. As the information age takes shape, we are witnessing an extraordinary and unprecedented growth in the appetite for access to services and use of information.

Many opportunities are emerging for consumers to access network- delivered content and interactive services in many locations one of the most important of which is the home. Major business drivers are the use of Audio & Video (AV) leisure and entertainment content and related digital information services, public utility content, and interpersonal communication services.

Nomadic Media addresses the intersection between (1) enabling and infrastructure technologies, (2) the services and content, and (3) people using content and services in their 'networked homes'.

Nomadic Media focuses on investigating technologies in the areas of situational awareness, human system interaction, interoperability and content management.

Manuscript received September 4, 2012.

B. de Ruyter is with Philips Research Europe, Eindhoven, The Netherlands. (e-mail: boris.de.ruyter@philips.com).

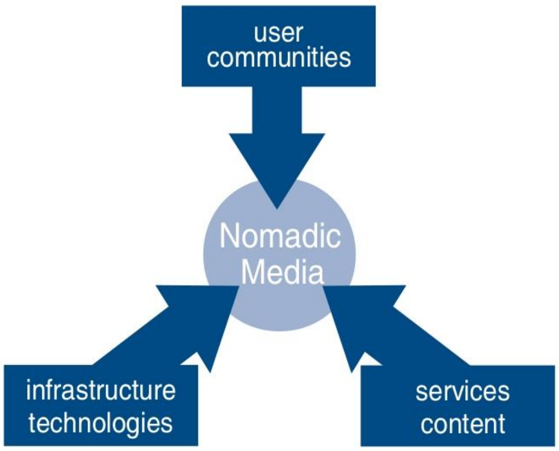

Fig. 2. Nomadic Media is at the intersection.

II. The virtual Home

The concept of home includes both a 'physical' space and the 'virtual' home – the personal world we carry around with us as we move from place to place. The anticipated proliferation of wireless communication and portable devices offers consumers increasing flexibility in where and when they can use content and services. So the notion of the home is rapidly evolving towards a combination of physical and virtual space. Also, consumers increasingly demand personalized delivery of content and services. These developments bring many benefits that people can enjoy at home and on the move – in fact, wherever and whenever they wish. But do we have the technology to realize these expectations? Today consumers are confronted with a diverse range of independent, disconnected solutions. There are significant limitations to moving content from place to place or from device to device. Many devices cannot readily interchange data. Use of interactive services tends to be limited to particular 'device and service' combinations. Moreover, many people find consumer electronic devices difficult to use.

III. Creating the right solutions

Nomadic Media was set up to develop innovative solutions that will allow consumers to use the devices that best suit their circumstances. The solutions should adapt more readily to their personal preferences and needs, be enjoyable to use, and provide low-entry thresholds for all sections of society.

Fig. 3. Understand the users and their needs

Finding the right design solutions that fit consumers' lifestyles is crucial to realizing the full potential of our networked futures. A major challenge for European industry is therefore to create solutions that are attractive to consumers because of the benefits they offer, and because they are enjoyable to use. To do this industry needs better insights into how devices will be used in practice, and how to design solutions accordingly. Nomadic Media will improve the capability of the partners to apply 'Usage-Centered-Design' (UCD) principles and so create solutions that are both useful and enjoyable.

Fig. 4. Investigating the user experience in a usability laboratory.

IV. Links to the 'value chain'

Networked systems will only work if content and services can easily be used in diverse locations using a variety of devices. Enabling flexible use, often on an ad-hoc basis, is key to ensuring that the value chains, and thereby the revenue streams, are intact and can develop at the levels required to ensure business success. Two key links in the chain are (1) to ensure that the function offered is of value to the consumer, and (2) that the device(s) used to access a service or function are designed to maximize benefits for the consumer. If either link is too weak, the whole value chain will be threatened. Nomadic Media addresses both of these links.

V. Project Results

The project obtained the following results:

- architecture solutions for interoperability and content management, and proposed extensions to relevant standards to cope with emerging requirements
- innovative Human System Interaction concepts for networked devices
- a proof of concept demonstration
- an architecture for content management and interoperability in multiple device environments, and data format definition
- User Centered Design guidelines for process, methods and techniques.

Nomadic Media benefits from the knowledge generated in several ITEA projects (Beyond, Ambience, HomeNet2Run, VHE Middleware and @Terminals).

The consortium includes partners who represent the various interests in the total technology and application development cycle and are used to working together.

Acknowledgment

The project was coordinated by McClelland and following organizations participated in making the project successful: Atos-Origin, Cefriel, CiaoLab, Cybelius Software, Euskaltel, Nokia, ORGA Systems, Philips Electronics Belgium, Philips Electronics Netherlands, Radiolinja, Siemens c-lab, SysOpen, University of Paderborn, University of Oulu, Vodafone Omnitel, VTT. .

References

[1] http://www.hitech-projects.com/euprojects/nomadic-media/

Florence – A Multipurpose Robotic Platform to Support Elderly at Home

Dietwig Lowet and Frank van Heesch

***Abstract*— Elderly want to live independently at home longer. In the Florence project, we investigate to what extent low-cost state of the art robot technologies can support elderly to meet this wish. The Florence robot provides several robotic services including, robotic telepresence, monitoring and coaching. In this article, we describe the Florence architecture that has the goal to simplify the development of applications for third parties, i.e. without having an in depth knowledge of robot technologies. We further motivate how the Florence robot can be integrated with smart home technologies.**

***Index Terms*— AAL, ambient intelligence, elderly, robotics**

I. Introduction

THE aim of the Florence project is to improve the well-being of elderly (and that of their loved ones) as well as improve the efficiency in care through Ambient Assisted Living (AAL) applications supported by a general-purpose mobile robot platform. During its early design phase, the Florence project has identified four main requirements on which the design of the Florence robot is based: low cost, support for social connectedness, safety and coaching, high user acceptance, and extendibility by third parties.

Fig. 1. The first version of the Florence robot.

The low-cost aspect (a target price of $1000) implies the technology needs to be based on the current state of the art. This eliminates the use of humanoid robots and limits us to a robot hardware design with a mobile base on wheels and a (touch) screen interface on top. Fig. 1 provides a picture of our first prototype.

This type of robot is suitable to address the needs of elderly with respect to three important AAL application areas: telepresence, monitoring and coaching. In these three application areas, we have developed a number of applications, as a proof of concept:

- *Keeping in touch*: communication via robotic telepresence
- *Fall handling*: fall detection and robotic telepresence to assess the severity and assist the elderly
- *Lifestyle management*: providing coaching advise to remain physical active
- *Home interface*: the robot as an easy metaphor to interact with the smart home
- *Agenda reminder*: providing a gentle way of reminding people of medications and appointments
- *Monitoring*: vital signs data and logging it for later data mining.

Fig. 2. Overview of the Florence proof-of-concept applications.

The aspect of user acceptance is very important. The robot should be usable for fun and lifestyle services e.g. listening to music, or controlling your home to prevent the robot to become negatively associated with a user’s reducing ability to be independent. After the early design phase, the project has followed a highly user-centered process, in which the Florence system has been tested with users at three points in time: Wizard of Oz testing at the start of the development, followed by a user test in controlled home environments half way through the project and a user tests at the end of the project at elderly’s homes.

The Florence robot is a multi-functional device, comparable to a PC or a smart phone, for which after product launch multiple applications can be deployed based on user

Manuscript received September 17, 2012.

Research supported by the 7th Framework Programme by the EC, grant ICT-2009-248730.

D. Lowet is with Philips Innovation Group Research, Eindhoven, 5656 AE Netherlands (dietwig.lowet@philips.com)

F. van Heesch is with Philips Innovation Group Research, Eindhoven, 5656 AE Netherlands (frank.van.heesch@philips.com).

preferences and available care provider services. Therefore, a platform approach has been used that supports easy development of third party applications.

This paper is organized as follows. We first provide an overview of the Florence platform in Section II. Subsequently, in Section III, we discuss how the Florence robot operates in conjunction with a smart home. We conclude by summarizing our findings in Section IV.

II. The Florence Platform

The Florence robot is built up out of a mobile base with wheels and an interactive touch screen brought to a height of 1.5m. The robot senses its environment with a 2D laser scanner, bumper sensors, a gyroscope, a wide-angle camera, and a 3D depth camera.

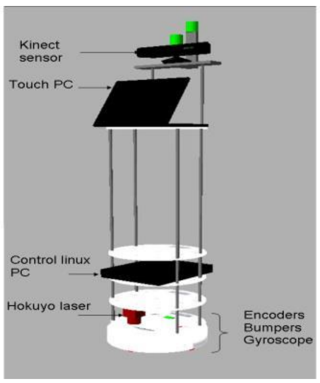

Fig. 3. Schematic overview of the 2nd version of the Florence robot

The robot contains two computing nodes; a Linux laptop at the base that handles all robot related functions and a windows Touch PC at the top that handles all human-robot interaction. The Florence robot prototype has a cost-price of around $2000 (which is close to our target of $1000).

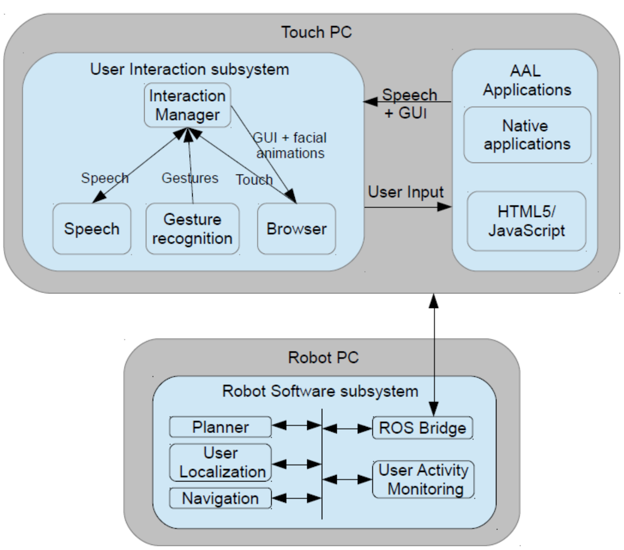

Fig. 4. The Florence software overview.

The Florence robot platform [2] provides functionality to AAL application developers such that a consistent behavior across applications is obtained, and to avoid that developers have to deal with low-level robot technologies and interaction modalities. This is achieved by providing high-level API's, for autonomous navigation and scheduling tasks and interacting with the user, such as: "*goToKitchen*", "*goToUser*", "*remindUser*" etc. Abbreviations and Acronyms

A. User Interaction Subsystem

The User Interaction Subsystem provides a consistent user interaction experience across applications. It also abstracts the communication for the different UI modalities that are supported: gestures, speech, and touch screen. The GUI part of applications is implemented in HTML5. The application logic of the AAL applications can also be implemented in HTM5/JavaScript, depending on the performance requirement. An application can equally be implemented in native code, if a high performance is required.

Applications interact with the components of the Florence platform via websockets [4], while they interact with the user through an Interaction Manager. This Interaction Manager has been developed using the Ravenclaw dialog-management system [5] as a basis.

To interact with users, an application provides a "dialog-tree" to the Interaction Manager. In its simplest form, a "dialog tree" is just a combination of one "output" (e.g. "Did you already take your medication?") and a list of possible "inputs" (e.g. "yes" or "no"). Dialogs with the user are rendered using, an animated face combined with text-to-speech, as well as text on the touch screen. The interaction manager contains the Elckerlyc animation framework [5] that synchronizes the movement of the lips with speech, while the spoken text is also displayed in a text balloon. The user can choose how he/she answers questions or provides input, either with speech (e.g. saying "yes", or "no"), gestures (e.g. nodding "yes" or nodding "no") or pressing the "yes" or "no" button on the touch screen. The communication modality is transparent to the applications; they only register to the inputs of a dialog tree.

To insert images in *Word*, position the cursor at the insertion point and either use Insert | Picture | From File or copy the image to the Windows clipboard and then Edit | Paste Special | Picture (with "float over text" unchecked).

IEEE will do the final formatting of your paper. If your paper is intended for a conference, please observe the conference page limits.

B. Robot Software Subsystem

The Robot Software Subsystem is based on the de facto standard, Robotic Operating System [3], and provides a high-level interface towards low-level robot functionality for autonomous indoor navigation, user localization and user activity monitoring. It contains also a planner for scheduling applications that can take user preferences and application requirements into account. In Florence, the communications between software components is socket-based and enables easy development and deployment.

C. Florence Planner

Since the robot is intended for multiple applications that possibly run simultaneously, a component that arbitrates which application can control the robot at any one time needs to be present. Note that in principle, the AAL applications can always run in the background, for example to process sensor data and monitor the elderly, however at any one moment in time, there can be only one application that controls the robot's actuators, i.e. the wheels and content on the screen.

Ideally, such a Planner component would know exactly when to schedule different applications. For scheduling the AAL applications, however, we take a cooperative approach, where applications indicate their scheduling requirements and release voluntarily control when asked for. The AAL applications indicate the time slot in which they would like to run, the urgency, the importance and the expected duration of the task. For indicating the timeslot, the Planner supports "soft timings" like, "after dinner", "before going to bed". The Planner receives all scheduling requests and finds a suitable schedule. When an application is due, the Planner component sends a "start" message to the application. After the application has finished, it sends an "ActivityStopped" in return to the Planner component. The Planner also takes the expected user schedule into account and tries to schedule each AAL application such that user activities like "watching TV", "taking a nap" are interrupted as little as possible.

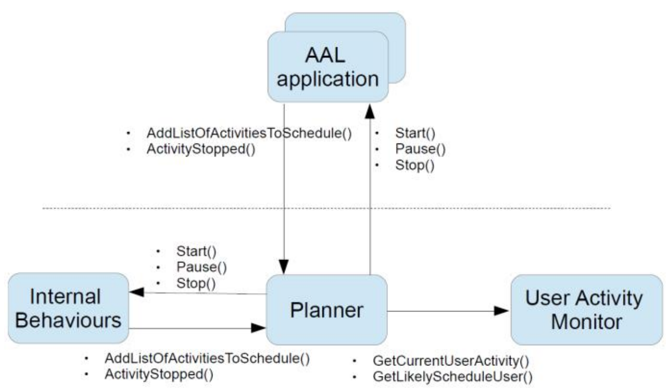

Fig. 5. Overview of Planner interfacing with other software components.

III. Robot Integration with Smart Home

In the Florence vision, the robot is a self-contained system that should be able to operate in any home, also traditional homes without smart home technology. The robot only requires internet connectivity for robotic telepresence. However, when smart home technology is available in a home, the robot system should be able to make use of it and integrate with the smart home technology. We can assume that future homes of elderly will be equipped with sensors such as. PIR sensors, emergency buttons, smoke detectors, a doorbell, and actuators, such as automated closing of windows and blinds. Home monitoring sensors and home actuators are used both for detecting emergencies as well as to enable comfort services like automated climate control. In case of the detection of an emergency situation, sensors can deliver information such as the location of the person and the place the emergency has happened. This information can be used by care givers and emergency services to handle the situation in a fast and reliable way.

In this section, we will first discuss how home sensors and actuators can be handled in a uniform way, using a context management framework, followed by the role a mobile robot can play in such a context. Thirdly, we will motivate why a mobile robot is preferable for activity monitoring from a user perspective.

A. The smart home model in Florence

One of the key practical difficulties with integrating the Florence robot with smart home systems is that there are so many sensor/actuator systems of various brands using different communication protocols and many different data standards. Instead of assuming or focusing on one home automation standard, a smart home model is considered that is generic enough to fit most future smart home networks. We assume that there are many sensors/actuators from many vendors inside the home. These sensors are either stand-alone (e.g. a room temperature sensor) or embedded in a device (e.g. a temperature sensor inside an oven). We assume that these sensors will communicate via their own proprietary protocol towards a sensor gateway. The role of the sensor gateway is to provide virtual IP connectivity to a number of sensors speaking the same proprietary protocol. The same holds for actuators as depicted in Fig. 6. For every protocol a separate sensor/actuator gateway is necessary.

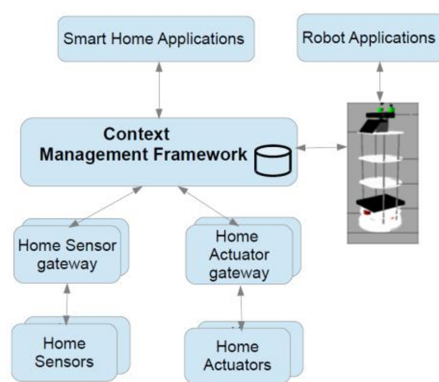

Fig. 6. Overview of the integration of the Florence robot with the smart home

In addition to providing IP connectivity to sensors and actuators, differences in data protocols have to be dealt with, before smart home applications can easily make use of them. To this end, a Context Management Framework is assumed in the Florence project that takes up a number of tasks:

- Providing a single access point and a unified API for home applications and home sensors and actuators.
- Providing sensor data fusion: sensor data from various sensors can be combined to provide richer and more abstract information towards applications.
- Providing storage facilities: for many applications it is beneficial to have the data history.
- Provides a rule engine: applications can provide rules on how to provide a response to a combination of sensors.

These rules can be executed independent of the application.

The Context Management Framework (CMF) is assumed to run on an "always-on" device, such as the home gateway, a NAT router, a NAS server, or a dedicated device.

In our project we use two types of sensor/actuator gateways, connecting FS20 [6][7] and HomeMatic [8] sensors/actuators for our prototyping work. FS20 and HomeMatic are both widely used proprietary protocols. In the Florence project, we use the following home sensors: pressure sensors in the couch and bed, power sensors to measure the power usage of oven, microwave, television, a fall detection button, presence sensors (PIR), a doorbell sensor, and door and window state sensors. We also use one type actuator than can remotely open and close windows. These smart home sensors are mainly used for user presence, user activity recognition, and emergency detection.

B. Integration of the Florence robot with the CMF

We make a distinction between "smart home applications" and "robot applications". Robot applications are applications that are specifically developed for the robot, such as robotic telepresence and an exercise coach. Smart home applications are applications that are specifically developed for a smart home, such as an automatic lighting control system or heating system. It would be beneficial if a smart home applications could make use of the robot's sensors and maybe also of the user interaction capabilities of the robot and vice versa.

When a robot enters a smart home that has a CMF as unified access point for sensor/actuator access, as explained in the previous section, it is relatively straightforward for the robot to make use of the smart home's sensors and actuators. The other way around is less straightforward. We argue that a robot has a number of characteristics that make it different from other typical in-home smart devices: firstly, it is mobile, secondly it is autonomous, and thirdly it has a high intelligence. One could argue that the robot is just a sensor/actuator and that all data could be transferred to the CMF, and rules could be applied and actuation commands are sent back to the robot to execute. However, this implementation does not scale very well with increasing complexity of robotic services. The sensor data of the robot is typically too high-bandwidth (laser scanner data, video data for computer vision and 3D depth data) to be continuously streamed back and forth over the network for processed by a rule engine. High-bandwidth data must be processed locally to extract relevant semantic data. Similarly, navigating a robot involves a high-frequent time critical control loop that is not easily handled remotely by a rule engine.

C. Robots – smart home integration from a user perspective

Adding a Florence-like robot to a smart home has two significant advantages from a user perspective: a robot enables the use of data rich cameras and microphones without invading privacy as much as the installation of cameras and microphones in a typical smart home setup would and a root also provides an intuitive interface to smart home technology and applications.

A mobile robot equipped with high-bandwidth sensors, like cameras and a 3D depth sensor, provide the unique possibility to monitor user activities and continuous vital signs of persons without obtrusion. Furthermore, such sensors can be used to detect patterns of daily life of the elderly like cooking, sleeping, eating, having diner, watching TV, reading, and having a visitor. Accurate detection of the user's activity is important for many AAL applications and more importantly enables the use of "soft timings", where applications can be scheduled relatively to a user activity. One specifically important example is medication reminders. These often need to be combined with the intake of a meal. Many smart home sensors are low-bandwidth sensors, and provide little information that is linked to privacy sensitive user data or user activity. In contrast, high-bandwidth sensors, precisely because of the large data they collect cause privacy concerns. Users do not like to obtrusion of constant monitoring. However, once placed on top of a robot, privacy concerns are alleviated in two important ways. First, a robot forms an easily understandable metaphor to the user when he/she is being observed and when not. Second, in case a user desires guaranteed privacy the robot can easily be sent to another room.

In the Ambient Intelligence (AmI) vision, where intelligent devices (the sensors, actuators and computing) become so small that the technology is hidden from the user except for the user interface, the Florence robot fits as a user interface towards ambient intelligence environments, such as a smart home. A mobile robot offers interaction via speech and gestures and provides an intuitive to understand metaphor (e.g. butler).

IV. Summary

In this paper, we have shown how smart home technologies and robot technologies can complement each other for providing AAL services to elderly. In the Florence view, robots can complement ambient intelligence environments (smart homes) in two ways. (1) The robot can be the interface to the smart home and (2) the robot can make use of high-bandwidth sensors possible without obtrusively breaching the user's privacy. Currently, a clear distinction is still visible between "smart home applications" and "robot applications". However, when the robot is sufficiently integrated with sensors through a CMF this distinction is expected to decrease.

References

[1] Florence project, http://www.florence-project.eu
[2] M.S. Bargh, M. Isken, D. Lowet, N. Snoeck, B. Hebgen, and H. Eertink, "Sensing, Actuation Triggering and Decision Making for Service Robots Deployed in Smart Homes", in 2nd International Joint Conference, AmI 2011, Nov. 16-18, 2011, Amsterdam, LNCS 7040
[3] Robot Operating System (ROS), http://www.ros.org/wiki/
[4] Websockets, http://tools.ietf.org/html/rfc6455
[5] Elckerlyc, http://hmi.ewi.utwente.nl/showcase/Elckerlyc
[6] http://www.elv.de/fs20-funkschaltsystem.html
[7] http://fhz4linux.info/tiki-index.php?page=FS20+Protocol
[8] HomeMatic, http://www.homematic.com/

A Multi Agent System Providing Situation-Aware Services in a Smart Environment

Berardina Nadja De Carolis and Stefano Ferilli

Abstract—**This paper proposes an agent-based approach for proactively adapting the behavior of a Smart Environment that, based on the recognized situation and user goal, selects a suitable workflow for combining services of the environment. To this aim we have developed a multi agent infrastructure composed by different classes of agents specialized in reasoning and learning about the user and the context at different abstraction levels.**

Index Terms—**Smart Environment, Multi Agent System, User Modeling, Smart Services**

I. Introduction

USERS of a smart environment often have contextual needs depending on the situation they are in. In order to satisfy them, it is important to personalize service fruition according to the user situation, and to make the interaction with services easy and natural for the user. To this aim we propose an approach based on software agents able to provide smart (i.e., integrated, interoperable and personalized) services, accessible through several interfaces available on the various devices in the environment. The environment, then, has to be able to reason on the situation of the user so as to understand his/her needs and goals for composing the most appropriate services. In the *manual* setting the user has to indicate to the system how to compose services [9]; *automatic service composition* takes place without human intervention, using planning techniques borrowed from Artificial Intelligence [14-16]. In *semi-automatic composition* the system guides interactively the user in finding, filtering and composing services [10]. We work in this direction and propose a Multi Agent System (MAS) architecture aimed at meeting this goal. In particular, in this paper we choose a Smart Home Environment (SHE) scenario, where the problem is of particular interest since it is necessary to combine services of the physical environment with net-centric ones according to the recognized situation. The idea underlying our approach is the metaphor of the butler in grand houses, who can be seen as an household affairs manager with duties of a personal assistant, able to organize the housestaff in order to meet the expectations of the house inhabitants. Taking into account the results of a previous project [2], we have developed a MAS in which the butler agent has to recognize the situation of the user, based on interaction with Sensor Agents, in order to infer possible user's goals. The recognized goals are then used to select the most suitable workflow among a set of available candidates [17]. Such a selection is made by matching semantically the goals, the current situation features and the effects expected by the execution of the workflow. Once a workflow has been selected, its actions are executed by the Effector Agents. Since the user may change the execution of the selected workflow by substituting, deleting, undoing the effects of some services, as any good butler, it should be able to learn about situational user preferences but it should leave to its "owner" the last word on critical decisions [5]. To this aim, the butler agent must be able to interpret the user's feedback appropriately, using it to revise: (i) the knowledge about the user, with respect to his preferences and goals in a given situation, and (ii) the workflow or the services invoked in it [17]. One important feature of the proposed architecture is the presence of a class of agents designed to take care of the interaction with the users. These agents are responsible for implementing several kinds of interfaces according to contextual factors and to the user preferences. Indeed, since in a completely proactive approach users may feel a loss of control over the system actions, in our semi-automatic approach to composition of services, the SHE proactively proposes smart services but leaves the control over proposed service composition by allowing users to select alternative services, to provide more preference information in order to get a better personalization, to ask for explanations about the proposed services and so on.

Manuscript received September 4, 2012.

B. De Carolis is with the Department of Computer Science, University of Bari, Italy (e-mail: decarolis@ di.uniba.it).

S. Ferilli is with the Department of Computer Science, University of Bari, Italy (e-mail: ferilli@ di.uniba.it).

II. The proposed Multi Agent System

According to the butler metaphor, as main tasks, the butler must know the habits of the house inhabitants, perceive the situation of the house and coordinate the house staff. To this aim we have designed the following classes of agents:

Sensor Agents (SA): These provide information about context parameters and features (temperature, light level, humidity, etc.) at a higher abstraction level than sensor data.

Butler Agent (BA): Its behavior is based on a combination of intelligent reasoning, machine learning, service-oriented computing and semantic Web technologies for flexibly coordinating and adaptively providing smart services in dynamically changing contexts. This agent reasons on the user's goals and devises the workflow to satisfy them.

Effector Agents (EA): Each appliance and device is controlled by an EA that reasons on the opportunity of performing an action instead of another in the current context.

Interactor Agents (IA): These are in charge of handling interaction with the user in order to carry on communicative tasks, also choosing the best interaction metaphor according to the situation and to the user's needs and preferences;

Housekeeper Agent (HA): It acts as a facilitator since it knows all the agents that are active in the house and also the goal they can fulfill.

Cyclically, or as an answer to a user action, the BA runs its reasoning model about the user. According to the situation provided by the appropriate SA, the BA infers and ranks the possible user goals and needs. Then, it selects the workflow associated with a specific goal by matching semantically the goal with the Input, Output, Pre-Condition and Effect (IOPE) [11] descriptions of the workflows stored in a repository. Once the most appropriate workflow has been selected and activated, the services/actions to be invoked among those available in the environment must be selected through semantic matchmaking, as well. Hence, the workflow services are invoked dynamically, matching the user's needs in the most effective way [1]. As regards Web Services, both simple and complex Web Services will be implemented according to the standard Web Ontology Language for Services (OWL-S) [13], which is an ontology that enables automatic service discovery, invocation, composition and execution monitoring. In particular, the composition of complex services from atomic services is based on their pre-conditions and post-conditions.

A. The Agents' Architecture

All agents in our MAS architecture are endowed with two main behaviors, reasoning and learning. The former interprets the input (e.g., in the case of SAs, data collected through the sensors of the smart environment) and processes this input according to its specific role (e.g., SAs transform sensor data in high-level knowledge about the situation). Although for simpler activities mathematical and statistical processing techniques can be sufficient, the complexity of most real-world environments, and specifically of those aimed at pro-actively supporting human users, requires the additional exploitation of more powerful kinds of reasoning and knowledge management, such as [4]: i) deduction: to draw explicit information that is hidden in the data, ii) abduction: to be able to sensibly proceed even in situations in which part of the data are missing or otherwise unknown and iii) abstraction: to strip off details that are known but useless for the specific current tasks and objectives. Hence, we use a logic language to express the knowledge base of our agents. In particular, the need to handle relationships among entities and possible situations calls for the first-order logic setting. An advantage of this setting is that the knowledge handled and/or learned by the system can be understood and checked by humans. In particular, the input to an agent is processed by its reasoning layer, for:

• deciding which signals are to be ignored and which ones are to be sent to other entities that can understand and exploit them (e.g. agents or user or devices, depending on the kind of agent) and/or to its learning functionality.

• processing and combining input data to detect significant patterns and produce more complex information, using different kinds of inference techniques.

• deciding which part of this information is to be ignored and which part is to be forwarded to other entities (see above) and/or to the learning functionality.

The learning behavior, on the other hand, is used by an agent to refine and improve its future performance. For example, the BA may exploit user feedback to refine the user model accordingly. The branch of Machine Learning dealing with first-order logic languages is Inductive Logic Programming (ILP) [12]. It is less developed, but much more powerful, and potentially more useful, than traditional approaches such as artificial neural networks or Bayesian learning. In particular, for the specific needs of adaptation posed by the present application, an incremental approach to learning is mandatory, because the continuous availability of new data and the evolving environment cannot be effectively tackled by static models, but require continuous adaptation and refinement of the available knowledge. An ILP system that fits the above requirements (multistrategy, incrementality), is described in [4]; also abstraction and abduction theories can be learned automatically [8].

Regardless of the kind of agent, its behaviors strictly cooperate in the same way. Reasoning uses the agent's knowledge to perform inferences that determine how the agent achieves its objectives. Learning exploits possible feedback on the agent's decisions to improve that knowledge, making the agent adaptive to the specific user needs and to their evolution in time. The output of the learning behavior consists of new knowledge gained from experience, that extends/refines the model on which the reasoning behavior is based. The main inference strategy that characterizes the learning layer of our agents is induction, although a cooperation with other strategies, such as those exploited in the reasoning behavior, is strongly advised, for a better integration of the new knowledge with the reasoning engine. Although all agents share the same architecture, they differ in the following aspects:

• level of complexity, depending on the kind of agent;

• techniques that can be exploited by the reasoning functionality (deduction, abstraction, …), that are different according to the agent role;

• tools: determining how the techniques are implemented in the behavior, they may change even among agents in the same class;

• theories: they change for each single agent (even for those having same complexity, techniques and tools) and are strictly related to the agent's role.

Of course, different agents work on different portions of knowledge on the domain and may require different effort and pose different problems. More details about the MAS can be found in [1].

III. An Example

It's evening and Jim, a 73 y.o. man, he is at home alone, he has a cold and fever. Jim is sitting on the bench in his living room and he is a bit bored. The room is equipped with sensors, which can catch information about the current situation of the user, and with effectors, acting and controlling devices and also the execution of digital services. The room is equipped with several interaction devices implementing different interaction metaphors. Examples of such devices are the TV, touch screens on several appliances, a smartphone and a social robot that is able to move around in the home and to engage natural language dialogue with its inhabitants. [7]

Sensors, controlled by SAs, are placed in the environment for providing information about context parameters and features. SAs use abstraction to reason about the correspondent contextual parameter. In the example, the SA providing information about temperature will abstract the centigrade value into a higher level representation such as "warm", "cold", and so on using a rule of the form:

```
cold(X,Y) :- temperature(X,T), T<18, user(Y),
  present(X,Y), jim(Y).
```

This type of rules can be directly provided by an expert (or by the user himself), or can be learned (and possibly later refined) directly from observation of user interaction [6].

According to the recognized context situation, the BA infers user goals and composes a smart service corresponding to a workflow that suitably integrates elementary services. The reasoning of this agent mainly involves deduction, to draw explicit information that is hidden in the data, and abduction, to be able to sensibly proceed even in situations in which part of the data are missing or otherwise unknown. In some cases, it may also use abstraction, which is performed at a higher level than SAs. Each observation of a specific situation can be formalized using a conjunctive logic formula under the Closed World Assumption, described as a snapshot at a given time. A model consists of a set of Horn clauses whose heads describe the target concepts and whose bodies describe the pre-conditions for those targets to be detected. For instance:

```
improveHealth(X):- present(X,Y), user(Y),
  has_fever(Y).

improveHealth(X):- present(X,Y), user(Y),
  has_headache(Y), cold(X,Y).

improveHealth(X) :- present(X,Y), user(Y),
  has_flu(Y).

improveMind(X) :- present(X,Y), user(Y), sad(Y).

improveMind(X) :- present(X,Y), user(Y), bored(Y).
```

Although very simple for the sake of brevity, these rules clearly show how the knowledge in the model is expressed using high-level concepts of the domain that can be understood, evaluated and possibly produced/modified by humans. A sample observation might be:

```
morning(t0),closedWindow(t0),present(t0,j),jim(j),
user(j), temperature(t0,14), has_flu(j), bored(j).
```

(i.e., "in situation at time t0 it is morning, the window is closed and the temperature is 14°; user Jim is present and Jim has flu"). Reasoning infers that Jim is cold: cold(t0,j). Being all the preconditions of the first and fourth rules in the model satisfied by this situation for X = t0 and Y = jim, the user goals *improveHealth* and *improveMind* are recognized for Jim at time t0, which may cause activation of suitable workflows aimed at attaining those results. The BA reasons not only on goals but also on workflows. Indeed, once a goal is triggered, it selects the appropriate workflow by performing a semantic matchmaking between the semantic IOPE description of the user's high-level goal and the semantic profiles of all the workflows available in the knowledge base of the system [13, 14].

In this example the semantic matchmaking process leads to two different workflows associated to the two high-level goals previously recognized. In our case, the main workflow includes two different subflows corresponding, respectively, to each goal: improveHealth and improveMind (see Fig. 1). These subflows include both simple actions, that can be directly executed, and subflow that need to be satisfied. This hierarchical matchmaking process stops when the resulting workflow is composed of simple goals that can be directly satisfied by invoking a net-centric service or through simple actions performed on the effectors. In both cases, the BA asks to the HA which EAs can satisfy each planned action and send the specific request to the EA in charge for handling actions regarding changes of a particular parameter (i.e. temperature, light, etc.). In particular, when the goal satisfied by a workflow (or by part of it) regards a communicative action, its execution is delegated to the Interactor Agents (IAs). In this specific case, the HA returns to the BA the list of agent that are responsible for implementing the interaction with the user through different modalities (e.g. on a touch screen, on the smartphone or by using the social robot present in the smart environment).

Examples of communicative tasks that IAs may carry out are: i) information seeking, in which the IA exploits interaction with the user to get hints on how to attain a simple goal and, based on this, possibly learn new preferences of that user with respect to the given context and situation, in order to continuously and dynamically improve adaptation; ii) information provision, for example, referring to the previous scenario, the user may ask the robot to provide more information to justify the choice for a given medicine [3]; iii) remind, for example, if the object of the reminder is to take a medicine, it might be useful to provide a message on the smartphone, that the users usually brings with him.

IV. Conclusions and Future Work

This contribution briefly illustrates how a multi agent infrastructure can be employed for handling the situation-aware adaptation of a Smart Environment behavior. In this MAS different types of agents cooperate to the adaptation process. This process is performed at different levels, starting from the interpretation of sensor data from SAs, the planning services satisfying recognized user's goals from the BA and arriving to the decision on how to act on devices from EAs and how to communicate with the user from IAs. The main peculiarity of the proposed architecture lies in the fact that all kinds of agents in the MAS are a specialization of an abstract class endowed with both reasoning and learning behavior. Reasoning, in turn, can exploit any combination of abstraction, deduction and abduction according to the role of the agent in the MAS. The learning behavior uses a fully incremental technique based on a first-order logic representation and can exploit induction to build/update the theories used by the various inference strategies on which reasoning is based. Finally, pervasive interaction with the user is implemented through the IAs, which adapt the choice of the most appropriate interaction metaphor to the context and to the user preferences and needs. Still, open problems remain and will be the subject of our future work (e.g., how to reason on users' reactions to the proposed workflow in order to adopt the optimal behavior of the SHE). In fact, when the user undoes or gives a negative feedback to one or more actions of the selected workflow, it is necessary to understand if this is just an exception or if it must affect the reasoning models, e.g. because there is a change in the situation that has not been detected or taken into account, a mistake in controlling the effectors to achieve a simple goal or a mistake in interpreting the user's goals or in selecting or composing the workflow. Each of the latter cases determines which agent in the MAS has made a wrong decision, and is to be involved in theory refinement.

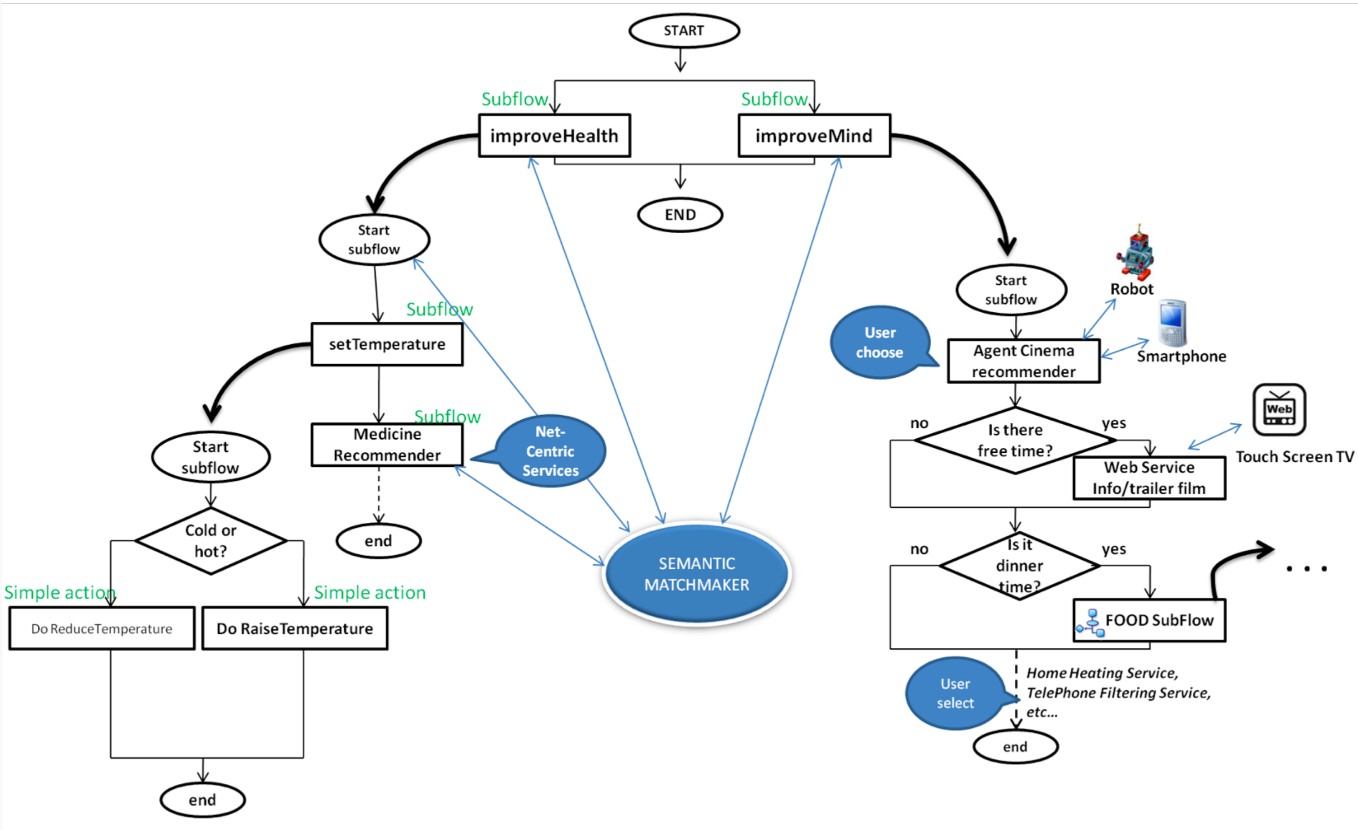

Fig. 1. An example of a Smart Service Workflow composed by the Butler Agent

References

[1] Cavone D., De Carolis B., Ferilli S., Novielli N. A Multiagent System Supporting Situation Aware Interaction with a Smart Environment. PECCS 2012: 67-72

[2] De Carolis, B., Cozzolongo, G. and Pizzutilo, S.: A Butler Agent for Personalized House Control. ISMIS 2006: 157-166

[3] De Carolis, B., I. Mazzotta, N. Novielli, NICA: Natural Interaction with a Caring Agent. In Ambient Intelligence, European Conference, AmI 2009, vol. 5859/2009, pages 159-163, Springer, 2009

[4] Esposito, F., Fanizzi, N., Ferilli, S., Basile, T.M.A. and Di Mauro, N. Multistrategy Operators for Relational Learning and Their Cooperation. Fundamenta Informaticae Journal, 69(4):389-409, IOS Press, 2006.

[5] Falcone, R. and Castelfranchi, C. Tuning the Collaboration Level with Autonomous Agents: A Principled Theory. AI*IA 2001: 212-224.

[6] Ferilli S., Basile T.M.A., Di Mauro N., Esposito F. On the LearnAbility of Abstraction Theories from Observations for Relational Learning. In: J. Gama, R. Camacho, G. Gerig, P. Brazdil, A. Jorge, L. Torgo. Machine Learning: ECML 2005. p. 120-132, Springer, 2005

[7] Ferilli, S., Cavone, D., De Carolis, B. and Novielli, N. A Layered Architecture for Situation Aware Home Environments, to appear in Proceedings of 6th Workshop on "Artificial Intelligence Techniques for Ambient Intelligence" (AITAmI'11)

[8] Ferilli S., Esposito F., Basile T.M.A., Di Mauro N. Automatic Induction of Abduction and Abstraction Theories from Observations. In: S. Kramer, B. Pfahringer. Inductive Logic Programming. 103-120, Springer, 2005.

[9] Humble, J., Crabtree, A., Hemmings, T., Åkesson, K.-P., Koleva, B., Rodden, T., and Hansson, P. Playing with the Bits - User-Configuration of Ubiquitous Domestic Environments, vol. 2864, Lecture Notes in Computer Science, pp. 256-263, 2003.
[10] Kim J., et al., An Intelligent Assistant for Interactive Workflow Composition,. 9th international conference on Intelligent User Interfaces, 2004, pp:125-131, ACM Press.
[11] Meyer, H. On the Semantics of Service Compositions. Lecture Notes in Computer Science, 2007, Volume 4524/2007, 31-42
[12] Muggleton, S.H. Inductive Logic Programming. New Generation Computing, 8(4):295-318, 1991.
[13] OWL-S, Semantic markup for web services; W3C member submission, 2004. 10
[14] Paolucci, M., Kawamura, T., Payne, T. and Sycara, K. Semantic Matching of Web Services Capabilities. In The Proceedings of The First International Semantic Web Conference (ISWC), Sardinia (Italy), June 2002. 11
[15] Paik, I. and Maruyama, D. "Automatic Web Services Composition Using Combining HTN and CSP," Conference on Computer and Information Technology (CIT 2007), Aizu- Wakamatsu City, Fukushima, Japan, 2007
[16] Srividya, K. et al., "Automatic Composition of Semantic Web Services," IEEE 2007 International Conference on Web Services (ICWS), Salt Lake City, USA, 2007.
[17] Yau, S.S. and Liu, J. Incorporating Situation Awareness in Service Specifications, isorc, pp.287-294, Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06), 2006.

Infrastructure and Architectural Principles for Plastic User Interfaces

Joëlle Coutaz, Gaëlle Calvary, Alexandre Demeure, Lionel Balme, Stéphane Lavirotte, Gaëtan Rey, and Jean-Yves Tigli

***Abstract*—This position paper discusses the problem of UI adaptation to the context of use. To address this problem, we propose to mix declarative languages as promoted in Model Driven Engineering (MDE) with a "code-centric" approach where pieces of code are encapsulated as service-oriented components (SOA), all of this within a unified software framework that blurs the distinction between the development stage and the runtime phase.**

***Index Terms*—User interface development, user interface plasticity, infrastructure for plastic user interfaces.**

I. Introduction

With the move to ubiquitous computing, it is increasingly important that user interfaces (UI) be adaptive or adaptable to the context of use (user, platform, physical and social environment) while preserving human-centered values [6]. We call this "UI plasticity". From the software perspective, UI plasticity goes far beyond UI portability and UI translation.

As discussed in [6], the problem space of plastic UI is complex: it covers UI re-molding, which consists in reshaping all (or parts) of a particular UI to fit the constraints imposed by the context of use. It also includes UI re-distribution (i.e. migration) of all (or parts) of a UI across the resources that are currently available in the interactive space. UI plasticity may affect all of the levels of abstraction of an interactive system, from the cosmetic surface level re-arrangements to deep re-organizations at the functional core and task levels. When appropriate, UI re-molding may be concerned by all aspects of the CARE properties [10], from synergistic-complementary multimodality (as in "put-that there") and post-WIMP UI's, to mono-modal GUI.

Re-molding and re-distribution should be able to operate at any level of granularity from the interactor level to the whole UI while guaranteeing state recovery at the user's action level. Because we are living in a highly heterogeneous world, we need to support multiple technological spaces[3] simultaneously such that a particular UI may be a mix of, say, Tcl/Tk, Swing, and XUL. And all of this, should be deployed dynamically under the appropriate human control by the way of a meta-UI.

A meta-UI is a special kind of end-user development environment whose set of functions is necessary and sufficient to control and evaluate the state of an interactive ambient space [6]. This set is *meta*- because it serves as an umbrella *beyond* the domain-dependent services that support human activities in this space. It is *UI*-oriented because its role is to allow users to control and evaluate the state of the ambient interactive space. By analogy, a meta-UI is to ambient computing what desktops and shells are to conventional workstations.

In the following, we will use Photo-Browser as a running example to illustrate the problem of UI plasticity and show how we have addressed this problem in terms of software principles, architecture and infrastructure.

II. An Exemplar: Photo-Browser

Photo-Browser supports photo browsing in a centralized or distributing way depending on the availability of a dynamic set of heterogeneous devices. These include a Diamond Touch interactive table, a wall, and a smart phone running Windows, MacOS X, and Android, respectively.

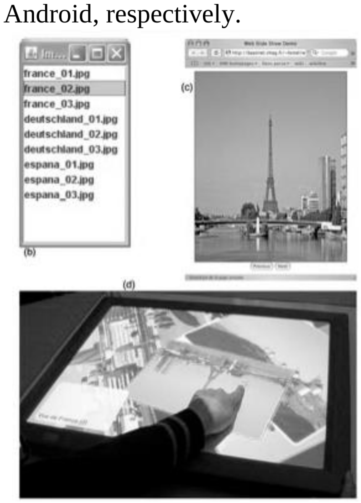

Fig. 1. Photo-browser: a dynamic composition of executable, distributed, and transformable UI components running on a dynamic heterogeneous platform (Windows, MacOS X, and Android).

Manuscript received September 4, 2012.

J. Coutaz, G. Calvary and A. Demeure are with University of Grenoble, France (e-mail: joelle.coutaz@imag.fr)

L. Balme, is with Immotronic, France.

S. Lavirotte, G. Rey and J.Y. Tigli, are with University of Nice, France.

[3] "A technological space is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities." [5]

The user interface of Photo-Browser is dynamically composed of: (1) a Tcl-Tk component running on the multi-point interactive surface (Fig. 1-d); (2) a Java component that shows a list of the image names (Fig. 1-b); (3) an HTML-based browser to navigate through the images set (Fig. 1-c); (4) a Java component running on the gPhone to navigate sequentially through the photos using Next and Previous buttons (Fig. 2).

Fig. 2. (Left) Connecting a Gphone to the interactive space by laying it down on the interactive table. (Right) Using the Gphone as a remote-controller to browse photos displayed by the HTML UI component of fig. 1-c and video-projected on the wall.

The gPhone is dynamically connected to the interactive space by laying it down on the interactive table (Fig. 2, left). As part of the platform, the gPhone can be used as a remote-controller to browse photos displayed by the HTML UI component of Fig. 1-c and video-projected on the wall.

Within the problem space of UI plasticity, the context of use covered by Photo-Browser is a dynamic heterogeneous platform, adaptation is multi-Technological Spaces based on redistribution at the interactor level (i.e. photos) with no remolding. The meta-UI includes the recognition of three gestures: a "wipe" gesture that allows the user to command the migration of the current selected photo from the table to the wall, the "wipe" gesture that commands the system to shut down the table and the contact of the gPhone with the Diamond Touch.

Having characterized Photo-Browser in the problem space of UI plasticity, we now report our observations about current practices before going into the principles and mechanisms we have developed to support UI plasticity.

III. Observations about current practices

Our approach to the problem of UI plasticity is based on the following observations:

(1) The software engineering community of HCI has developed a refinement process that now serves as a reference model for many tools and methods [6]: from a task model, an abstract UI (AUI) is derived, and from there, the Concrete UI (CUI) and the Final UI (FUI) are produced for a particular targeted context of use. The process is sound but cannot cope with ambient computing where task arrangement may be highly opportunistic and unpredictable.

(2) Software tools and mechanisms tend to make a dichotomy between the development stage and the runtime phase making it difficult to articulate run-time adaptation based on semantically rich design-time descriptions. In particular, the links between the FUI and its original task model are lost. As a result, it is very hard to re-mold a particular UI beyond the cosmetic surface.

(3) Pure automatic UI generation from high level models is appropriate for simple (not to say simplistic, "fast-food") UI's. The nuances imposed by high-quality multi-modal UI's and post-WIMP UI's (as the UI of Photo-Browser), call for powerful specification whose complexity might be as high as programming the FUI directly with the appropriate toolkit. In addition, conventional UI generation tools are based on a single target toolkit. As a result, they are unable to cross multiple technological spaces.

(4) Software adaptation has been addressed using many approaches over the years, including Machine Learning, Model-Driven Engineering (MDE), and service-oriented components. These paradigms have been developed in isolation and without paying attention to UI-specific requirements. Typically, a "sloppy" dynamic reconfiguration at the middleware level is good enough if it preserves system autonomy. It is not "observable" to the end-user whereas UI re-molding and UI re-distribution are! Thus, UI plasticity adds extra constraints such as making explicit the transition between the source and the target UI's so that the end-user can evaluate the new state.

Based on these observations, we propose to reconcile Model Driven Engineering (MDE) with code-centric approaches in the following way.

IV. Principles and Architecture for UI plasticity

We propose to conciliate MDE and "code-centricity" with the following three principles: (1) Cooperation between closed-adaptiveness and open-adaptiveness; (2) Runtime availability of high-level models; (3) Balance between the importance of Principles #1 and #2.

A. Principles #1: Cooperation between closed-adaptiveness and open-adaptiveness

"A system is open-adaptive if new application behaviors and adaptation plans can be introduced during runtime. A system is closed-adaptive if it is self-contained and not able to support the addition of new behaviors" [17]. By design, an interactive system has an "innate domain of plasticity": it is closed-adaptive for the set of contexts of use for which this system/component can adapt on its own. For unplanned contexts of use, the system is forced to go beyond its domain of plasticity. It must be open-adaptive so that a tier infrastructure (i.e. a middleware) can take over the adaptation process. We have proposed the CAMELEON Runtime conceptual architecture (CAMELEON-RT) as a way to support closed-adaptiveness and open-adaptiveness harmoniously [1] (See Fig. 3).

At the bottom of Fig. 3, "Hardware" denotes a wide variety of physical entities: computing and communication facilities, interaction resources such as displays, mice, and stylus, as well as sensors and actuators. "Operating Systems" includes legacy OS such as Linux, MacOS and Android, virtual machines such as the JVM, and modality interpreters such as speech and gesture recognition. Together, “Hardware” and “Operating Systems” constitute the ground-basis of the platform.

The top of Fig. 3 shows the interactive systems (e.g., Photo-Browser) that users are currently running in the interactive space. The Meta-UI is one of them. A flower-like shape, , denotes open-adaptive components of these interactive systems. Components are open-adaptative if they provide the world with management mechanisms. Management mechanisms include self-descriptive meta-data (such as the current state and the services it supports and requires), and the methods to control its behavior such as start/stop and get/set-state. Software reflexivity coupled with a component model is a good approach to achieve open-adaptiveness. The miniature adaptation-manager shape, , denotes facilities embedded in the interactive system to support closed-adaptiveness to observe the world, to detect situations that require adaptation, to compute a reaction that satisfies the new situation, and to perform adaptation. This functional decomposition is similar to that of the tier infrastructure shown in the center of Fig.3.

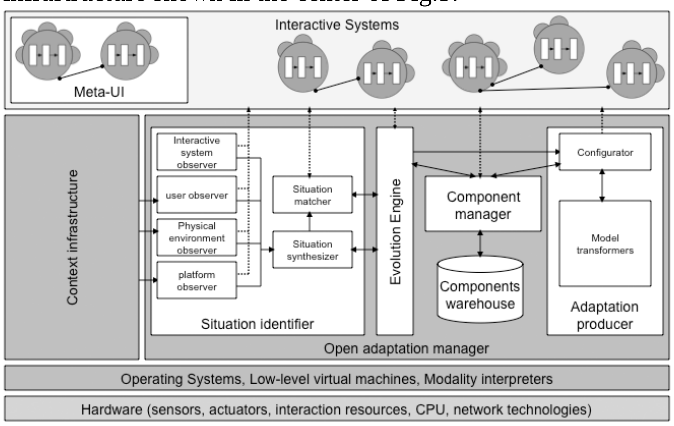

Fig. 3. CAMELEON RT: a functional decomposition for supporting a mix of closed-adaptiveness and open-adaptiveness at runtime.

The tier infrastructure that supports open-adaptiveness is structured in the following way:

- The context infrastructure builds and maintains a model of the context of use [11,14]. In turn, this infrastructure can be refined into multiple levels of abstraction, typically: raw data acquisition as numeric observables, transformation of raw data at the appropriate level of abstraction (e.g., as symbolic observables) which then feeds into situation management.
- The situation synthesizer computes the situation and possibly informs the evolution engine of the occurrence of a new situation. (This layer is in general considered as part of the context infrastructure.)
- The evolution engine elaborates a reaction in response to the new situation.
- The adaptation producer implements the adaptation plan produced by the evolution engine. This is where the following dimensions of the problem space of UI plasticity come in play: granularity of UI remolding and/or redistribution, granularity of state recovery, coverage of technological spaces, and presence of a meta-UI.

Such a functional decomposition is commonly used for the development of autonomic systems. To adapt this decomposition to plastic UI’s, we propose the following improvements:

- The end-user is kept in the loop: the reaction to a new situation may be a mix of specifications provided by developers or learnt by the evolution engine based on observations of, and reasoning on human and environmental behavior. In addition, the evolution engine as well as the adaptation producer may call upon end-users’ advice by the way of the meta-UI.
- The components referred to in the action plan do not necessarily exist as executable code. This is where Principle #2 comes into play.

B. Principle #2: Runtime availability of high-level of abstraction models

At runtime, we propose that an interactive system is a set of graphs of models that express different aspects of the system at multiple levels of abstraction. These models are related by mappings and transformations. As a result, an interactive system is not limited to a set of linked pieces of code. Models developed at design-time, which convey high-level design decision, are still available at runtime for performing rational adaptation beyond cosmetic changes. When a component retrieved by the component manager is a high-level description such as a task model, the configurator relies on reificators to produce executable code as in Digymes [9] and iCrafter [19]. A retrieved component may be executable, but may not fit the requirements. Ideally, it can be reversed-engineered through abstractors, then transformed by translators and reified again into executable code [4].

C. Principle #3: Balance between Principles #1 and #2

By analogy with the Slinky meta-model of the well-known Arch architecture model used in HCI [2], the software developer can play with principles #1 and #2. At one extreme, the interactive system may exist as one single task model linked to one single AUI graph, linked to a single CUI graph, etc. This application of Principle #1 does not indeed leave much flexibility to cope with unpredictable situations unless it relies completely on the tier middleware infrastructure that can modify any of these models on the fly, then triggers the appropriate transformations to update the Final UI. This approach works well for interactive systems for which conventional WIMP user interfaces are “good enough”.

At the other extreme, the various perspectives of the system (task models, AUI, FUI, context model, etc.) as well as the adaptation mechanisms of the tier infrastructure are distributed across distinct UI service-oriented components, each one covering a small task grain that can be run in different contexts of use. This approach has been applied in the Comet toolkit [13].

These principles and architecture have been applied to the implementation of Photo-Browser using two distinct tier middleware infrastructures: Ethylene developed with HCI-centered concerns, then with WCOMP, a general purpose service-oriented middleware. Ethylene is a distributed system composed of Ethylene factories each one running on possibly different IP devices. An Ethylene factory manages the life cycle of a set of components that reside on the same IP device as this factory, and that have been registered to this factory. When residing on storage space, a component is meta-described using EthylenXML, an extension of the W3C standard WSDL (Web Service Definition Language). This meta-description includes the human task that the component supports, the resources it requires, and whether it is executable code or transformable code. In the latter case, it may be a task model, an AUI, a CUI, or even a graph of these models. For example, the HTML-based component (Fig. 1-c) is a CUI expressed in a variation of HTML. It must be transformed on the fly to be interpreted by an HTML renderer. The Tcl-Tk multi-point UI and the Java list are executable code. Their EthyleneXML meta-description specifies that they support image browsing and image selection tasks, that they need such and such interaction resources (e.g., a Tcl-Tk interpreter and a Diamond Touch interactive table) for proper execution, and that they require such and such communication protocol to be interconnected with other components. The Gphone UI component is an executable Gphone app that supports the next-previous browsing tasks (Fig. 2). Interconnection between the components is initiated by the factories.

In contrast to Ethylene, WCOMP [20] conveniently includes an Architecture Description Language (ADL) that the evolution engine uses to express adaptation plans. An adaptation plan, which is interpreted by the adaptation producer (here, an ADL interpreter), specifies which components to stop or instantiate and how to reconfigure the connections between them.

In short, the same ambient system (i.e. Photo-Browser) has been implemented from the same domain-dependent components (that of Photo-Browser) with two distinct tier infrastructures developed by very different research teams but applying the architectural principles described above.

V. CONCLUSION

As shown by the example above as well as by other work [3, 8, 15, 21], the engineering community of HCI has focused its attention on runtime adaptation of the UI portion of interactive systems, not on the dynamic adaptation of the interactive system as a whole. The software engineering community is developing several approaches to enable dynamic bindings for service-oriented architectures. For example, Canfora et al. propose the dynamic composition of web services based on BPEL4People (that expresses a task-like model) as well as an extension of WSDL to meta-describe the services and using these two descriptions to generate the corresponding user interface [7]. Although bindings can be performed at runtime, users are confined within the workflow designed by the software developers. In addition, the generated UI's are limited to conventional WIMP user interfaces.

One promising approach to support flexibility at runtime, is to consider the functional core components as well as UI components as services. In Ethylene, UI components adhere to this philosophy. They can be implemented in very different technologies, they can be discovered and recruited on the fly based on their meta-description, and they can be transformed on the fly. On the other hand, the business logic side of interactive systems is left opened. CRUISe [17] aims at supporting both sides in a uniform way, but applies only to the dynamic composition of web services and UI composition for the web [22].

ACKNOWLEDGMENT

This work is the result of research supported by the CAMELEON European project (IST-2000-28323) and the ANR CONTINUUM project (ANR-08-VERS-005).

REFERENCES

[1] Balme, L., Demeure, A., Barralon, N., Coutaz, J., & Calvary, G. "CAMELEON-RT: A Software Architecture Reference Model for Distributed, Migratable, and Plastic User Interfaces", In Proc. *Ambient intelligence, EUSAI 2004*, Springer-Verlag, 2004, pp. 291–302.

[2] Bass, L., Faneuf, R., Little, R., Mayer, N., Pellegrino, B., Reed, S., Seacord, R., Sheppard, S., & Szczur, M. "Arch, a meta-model for the runtime architecture of an interactive system". The UIMS Developers Workshop. *SIGCHI Bulletin*, 24(1), ACM Publ., pp. 32–37.

[3] Blumendorf, M., Leehmann, G & Albayrak, S. "Bridging models and systems at runtime to build adaptive user interfaces". In Proc. *EICS 2010*, ACM Publ., 2010, pp. 9-18.

[4] Bouillon, L., & Vanderdonckt, J. "Retargeting web pages to other computing platforms". Proc. of *IEEE WCRE'2002*, 2002, pp. 339–348.

[5] Bourguin, G., Lewandowski , A. &Tarby, J.-C. "Defining Task Oriented Component". In Proc. *TAMODIA*, Springer, 2007, pp. 170-183.

[6] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Souchon, N., Bouillon, L., & Vanderdonckt, J. "A Unifying Reference Framework for Multi-target User Interfaces". *Interacting with Computers*, Elsevier Science B.V., 15(3), 2003, pp. 289-308.

[7] Canfora, G., Di Penta, M., Lombardi, P. & Villani, M.L. "Dynamic Composition of Web Applications in Human centered Processes". IEEE *PESOS'09*, 2009.

[8] Clerckx, T., Vandervelpen, C. & Coninx, K. "Task-based design and runtime support for multimodal user interface distribution". In *EICS*'07.

[9] Coninx, K., Luyten, K., Vandervelpen, C., Van den Bergh, J., & Creemers, B. "Dygimes: Dynamically generating interfaces for mobile computing devices and embedded Systems". In Proc. *Mobile HCI*, 2003, pp. 256–270.

[10] Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J. & Young, R. "Four Easy Pieces for Assessing the Usability of Multimodal Interaction: The CARE properties", I*NTERACT'95*, Chapman&Hall Publ., 1995, pp. 115-120.

[11] Coutaz, J., Crowley, J., Dobson, S., & Garlan, D. "Context is key". *Communications of the ACM*, ACM Publ., 48(3), 2005, pp. 49–53.

[12] Coutaz, J., Calvary, G. "HCI and Software Engineering for user Interface Plasticity", in *The Human Computer Handbook – Fundamentals, Evolving Technologies, and Emerging Applications*, 3rd edition, J.A. Jacko Eds., Taylor&Francis, 2012.

[13] Demeure, A., Calvary, G., Koninx, K. (2008). "A Software Architecture Style and an Interactors Toolkit for Plastic User Interfaces". In Proc. *DSV-IS 2008*, LNCS, Springer Berlin, 2008, pp. 225-237.

[14] Dey, A. K. "Understanding and using Context". *Journal of Personal and Ubiquitous Computing*, Springer London, Vol 5, 2001, pp. 4-7.

[15] Duarte, C. & Carriço, L. "A conceptual framework for developing adaptive multimodal applications". In Proc. of the 11th int. conference on Intelligent User Interfaces, IUI'06, ACM, New York, 2006, pp. 132-139.

[16] Kurtev, I., Bézivin, J., & Aksit, M. "Technological spaces: An initial appraisal". *CoopIS, DOA'2002 Federated Conferences*, 2002.

[17] Oreizy, P., Gorlick, M., Taylor, R., Heimbigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D. & Wolf, A. "An Architecture-Based Approach to Self-Adaptive Software", IEEE Intelligent Systems, May/June, 1999, pp. 54-62.

[18] Pietschmann, S., Voigt, M. & MeiBner, K. "Dynamic Composition of Service-Oriented Web User Interfaces". Proc. I*EEE ICIW 2009*, 2009.

[19] Ponnekanti, S., Lee, B., Fox, A., Hanrahan, P., & Winograd, T. "Icrafter: A service framework for ubiquitous computing environments". In Proc. *Ubicomp 2001*, Springer Publ., LNCS 2201, pp. 57–75.
[20] Tigli, J.Y., Lavirotte, S., Rey, G., Hourdin, V., Riveill, M. "Lightweight Service Oriented Architecture for Pervasive Computing." *International Journal of Computer Science* issues 4(1), 2011, pp. 1-9
[21] Savidis, A. & Stephanidis, C. "Software refactoring process for adaptive user interface composition". In Proc. of the 2010 ACM SIGCHI EICS 2010, ACM Publ., 2010, pp. 19-28.
[22] Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F. M. & Matera, M. "A Framework for rapid Integration of Presentation Components". In Proc. *WWW'07*, 2007, pp. 923-932.

Science Reports **Department of Mathematics and Computer Science**
Technische Universiteit Eindhoven

If you want to receive reports, send an email to: wsinsan@tue.nl (we cannot guarantee the availability of the requested reports).

In this series appeared (from 2009):

09/01	Wil M.P. van der Aalst, Kees M. van Hee, Peter Massuthe, Natalia Sidorova and Jan Martijn van der Werf	Compositional Service Trees
09/02	P.J.l. Cuijpers, F.A.J. Koenders, M.G.P. Pustjens, B.A.G. Senders, P.J.A. van Tilburg, P. Verduin	Queue merge: a Binary Operator for Modeling Queueing Behavior
09/03	Maarten G. Meulen, Frank P.M. Stappers and Tim A.C. Willemse	Breadth-Bounded Model Checking
09/04	Muhammad Atif and MohammadReza Mousavi	Formal Specification and Analysis of Accelerated Heartbeat Protocols
09/05	Michael Franssen	Placeholder Calculus for First-Order logic
09/06	Daniel Trivellato, Fred Spiessens, Nicola Zannone and Sandro Etalle	POLIPO: Policies & OntoLogies for the Interoperability, Portability, and autOnomy
09/07	Marco Zapletal, Wil M.P. van der Aalst, Nick Russell, Philipp Liegl and Hannes Werthner	Pattern-based Analysis of Windows Workflow
09/08	Mike Holenderski, Reinder J. Bril and Johan J. Lukkien	Swift mode changes in memory constrained real-time systems
09/09	Dragan Bošnački, Aad Mathijssen and Yaroslav S. Usenko	Behavioural analysis of an I²C Linux Driver
09/10	Ugur Keskin	In-Vehicle Communication Networks: A Literature Survey
09/11	Bas Ploeger	Analysis of ACS using mCRL2
09/12	Wolfgang Boehmer, Christoph Brandt and Jan Friso Groote	Evaluation of a Business Continuity Plan using Process Algebra and Modal Logic
09/13	Luca Aceto, Anna Ingolfsdottir, MohammadReza Mousavi and Michel A. Reniers	A Rule Format for Unit Elements
09/14	Maja Pešić, Dragan Bošnački and Wil M.P. van der Aalst	Enacting Declarative Languages using LTL: Avoiding Errors and Improving Performance
09/15	MohammadReza Mousavi and Emil Sekerinski, Editors	Proceedings of Formal Methods 2009 Doctoral Symposium
09/16	Muhammad Atif	Formal Analysis of Consensus Protocols in Asynchronous Distributed Systems
09/17	Jeroen Keiren and Tim A.C. Willemse	Bisimulation Minimisations for Boolean Equation Systems
09/18	Kees van Hee, Jan Hidders, Geert-Jan Houben, Jan Paredaens, Philippe Thiran	On-the-fly Auditing of Business Processes
10/01	Ammar Osaiweran, Marcel Boosten, MohammadReza Mousavi	Analytical Software Design: Introduction and Industrial Experience Report
10/02	F.E.J. Kruseman Aretz	Design and correctness proof of an emulation of the floating-point operations of the Electrologica X8. A case study

10/03	Luca Aceto, Matteo Cimini, Anna Ingolfsdottir, MohammadReza Mousavi and Michel A. Reniers	On Rule Formats for Zero and Unit Elements
10/04	Hamid Reza Asaadi, Ramtin Khosravi, MohammadReza Mousavi, Neda Noroozi	Towards Model-Based Testing of Electronic Funds Transfer Systems
10/05	Reinder J. Bril, Uğur Keskin, Moris Behnam, Thomas Nolte	Schedulability analysis of synchronization protocols based on overrun without payback for hierarchical scheduling frameworks revisited
10/06	Zvezdan Protić	Locally unique labeling of model elements for state-based model differences
10/07	C.G.U. Okwudire and R.J. Bril	Converting existing analysis to the EDP resource model
10/08	Muhammed Atif, Sjoerd Cranen, MohammadReza Mousavi	Reconstruction and verification of group membership protocols
10/09	Sjoerd Cranen, Jan Friso Groote, Michel Reniers	A linear translation from LTL to the first-order modal μ-calculus
10/10	Mike Holenderski, Wim Cools Reinder J. Bril, Johan J. Lukkien	Extending an Open-source Real-time Operating System with Hierarchical Scheduling
10/11	Eric van Wyk and Steffen Zschaler	1st Doctoral Symposium of the International Conference on Software Language Engineering (SLE)
10/12	Pre-Proceedings	3rd International Software Language Engineering Conference
10/13	Faisal Kamiran, Toon Calders and Mykola Pechenizkiy	Discrimination Aware Decision Tree Learning
10/14	J.F. Groote, T.W.D.M. Kouters and A.A.H. Osaiweran	Specification Guidelines to avoid the State Space Explosion Problem
10/15	Daniel Trivellato, Nicola Zannone and Sandro Etalle	GEM: a Distributed Goal Evaluation Algorithm for Trust Management
10/16	L. Aceto, M. Cimini, A.Ingolfsdottir, M.R. Mousavi and M. A. Reniers	Rule Formats for Distributivity
10/17	L. Aceto, A. Birgisson, A. Ingolfsdottir, and M.R. Mousavi	Decompositional Reasoning about the History of Parallel Processes
10/18	P.D. Mosses, M.R. Mousavi and M.A. Reniers	Robustness os Behavioral Equivalence on Open Terms
10/19	Harsh Beohar and Pieter Cuijpers	Desynchronisability of (partial) closed loop systems
11/01	Kees M. van Hee, Natalia Sidorova and Jan Martijn van der Werf	Refinement of Synchronizable Places with Multi-workflow Nets - Weak termination preserved!
11/02	M.F. van Amstel, M.G.J. van den Brand and L.J.P. Engelen	Using a DSL and Fine-grained Model Transformations to Explore the boundaries of Model Verification
11/03	H.R. Mahrooghi and M.R. Mousavi	Reconciling Operational and Epistemic Approaches to the Formal Analysis of Crypto-Based Security Protocols
11/04	J.F. Groote, A.A.H. Osaiweran and J.H. Wesselius	Benefits of Applying Formal Methods to Industrial Control Software
11/05	Jan Friso Groote and Jan Lanik	Semantics, bisimulation and congruence results for a general stochastic process operator
11/06	P.J.L. Cuijpers	Moore-Smith theory for Uniform Spaces through Asymptotic Equivalence
11/07	F.P.M. Stappers, M.A. Reniers and S. Weber	Transforming SOS Specifications to Linear Processes
11/08	Debjyoti Bera, Kees M. van Hee, Michiel van Osch and Jan Martijn van der Werf	A Component Framework where Port Compatibility Implies Weak Termination
11/09	Tseesuren Batsuuri, Reinder J. Bril and Johan Lukkien	Model, analysis, and improvements for inter-vehicle communication using one-hop periodic broadcasting based on the 802.11p protocol

11/10	Neda Noroozi, Ramtin Khosravi, MohammadReza Mousavi and Tim A.C. Willemse	Synchronizing Asynchronous Conformance Testing
11/11	Jeroen J.A. Keiren and Michel A. Reniers	Type checking mCRL2
11/12	Muhammad Atif, MohammadReza Mousavi and Ammar Osaiweran	Formal Verification of Unreliable Failure Detectors in Partially Synchronous Systems
11/13	J.F. Groote, A.A.H. Osaiweran and J.H. Wesselius	Experience report on developing the Front-end Client unit under the control of formal methods
11/14	J.F. Groote, A.A.H. Osaiweran and J.H. Wesselius	Ananlyzing a Controller of a Power Distribution Unit Using Formal Methods
11/15	John Businge, Alexander Serebrenik and Mark van den Brand	Eclipse API Usage: The Good and The Bad
11/16	J.F. Groote, A.A.H. Osaiweran, M.T.W. Schuts and J.H. Wesselius	Investigating the Effects of Designing Control Software using Push and Poll Strategies
11/17	M.F. van Amstel, A. Serebrenik And M.G.J. van den Brand	Visualizing Traceability in Model Transformation Compositions
11/18	F.P.M. Stappers, M.A. Reniers, J.F. Groote and S. Weber	Dogfooding the Structural Operational Semantics of mCRL2
12/01	S. Cranen	Model checking the FlexRay startup phase
12/02	U. Khadim and P.J.L. Cuijpers	Appendix C / G of the paper: Repairing Time-Determinism in the Process Algebra for Hybrid Systems ACP
12/03	M.M.H.P. van den Heuvel, P.J.L. Cuijpers, J.J. Lukkien and N.W. Fisher	Revised budget allocations for fixed-priority-scheduled periodic resources
12/04	Ammar Osaiweran, Tom Fransen, Jan Friso Groote and Bart van Rijnsoever	Experience Report on Designing and Developing Control Components using Formal Methods
12/05	Sjoerd Cranen, Jeroen J.A. Keiren and Tim A.C. Willemse	A cure for stuttering parity games
12/06	A.P. van der Meer	CIF MSOS type system
12/07	Dirk Fahland and Robert Prüfer	Data and Abstraction for Scenario-Based Modeling with Petri Nets
12/08	Luc Engelen and Anton Wijs	Checking Property Preservation of Refining Transformations for Model-Driven Development
12/09	M.M.H.P. van den Heuvel, M. Behnam, R.J. Bril, J.J. Lukkien and T. Nolte	Opaque analysis for resource-sharing components in hierarchical real-time systems - extended version –
12/10	Milosh Stolikj, Pieter J. L. Cuijpers and Johan J. Lukkien	Efficient reprogramming of sensor networks using incremental updates and data compression
12/11	John Businge, Alexander Serebrenik and Mark van den Brand	Survival of Eclipse Third-party Plug-ins
12/12	Jeroen J.A. Keiren and Martijn D. Klabbers	Modelling and verifying IEEE Std 11073-20601 session setup using mCRL2
12/13	Ammar Osaiweran, Jan Friso Groote, Mathijs Schuts, Jozef Hooman and Bart van Rijnsoever	Evaluating the Effect of Formal Techniques in Industry
12/14	Ammar Osaiweran, Mathijs Schuts, and Jozef Hooman	Incorporating Formal Techniques into Industrial Practice
13/01	S. Cranen, M.W. Gazda, J.W. Wesselink and T.A.C. Willemse	Abstraction in Parameterised Boolean Equation Systems
13/02	Neda Noroozi, Mohammad Reza Mousavi and Tim A.C. Willemse	Decomposability in Formal Conformance Testing

13/03	D. Bera, K.M. van Hee and N. Sidorova	Discrete Timed Petri nets
13/04	A. Kota Gopalakrishna, T. Ozcelebi, A. Liotta and J.J. Lukkien	Relevance as a Metric for Evaluating Machine Learning Algorithms
13/05	T. Ozcelebi, A. Weffers-Albu and J.J. Lukkien	Proceedings of the 2012 Workshop on Ambient Intelligence Infrastructures (WAmIi)