Luciano Baresi
Schahram Dustdar
Harald Gall
Maristella Matera (Eds.)

LNCS 3272

Ubiquitous Mobile Information and Collaboration Systems

Second CAiSE Workshop, UMICS 2004
Riga, Latvia, June 2004
Revised Selected Papers

Springer

Lecture Notes in Computer Science 3272

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Luciano Baresi Schahram Dustdar
Harald Gall Maristella Matera (Eds.)

Ubiquitous Mobile Information and Collaboration Systems

Second CAiSE Workshop, UMICS 2004
Riga, Latvia, June 7-8, 2004
Revised Selected Papers

Springer

Volume Editors

Luciano Baresi
Maristella Matera
Politecnico di Milano
Dipartimento di Elettronica e Informazione
Piazza L. da Vinci, 32, 20133 Milano, Italy
E-mail: {baresi,matera}@elet.polimi.it

Schahram Dustdar
TU Wien, E 184 Institut für Informationssysteme
Argentinierstr. 8/184-1, 1040 Wien, Austria
E-mail: schahram.dustdar@tuwien.ac.at

Harald Gall
University of Zürich, Deparment of Informatics
Winterthurerstr. 190, 8057 Zürich, Switzerland
E-mail: gall@ifi.unizh.ch

Library of Congress Control Number: 2004117073

CR Subject Classification (1998): H.4, C.2.4, D.2, H.3, H.5.3

ISSN 0302-9743
ISBN 3-540-24100-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11361930 06/3142 5 4 3 2 1 0

Preface

Over recent years most business processes have changed in various dimensions (e.g., flexibility, interconnectivity, coordination style, autonomy) due to market conditions, organizational models, and usage scenarios of information systems. Frequently, information is relocated within a geographically distributed system according to rules that are only seldom defined as a well-codified business process. This creates the need for a software infrastructure that enables ubiquitous mobile and collaboration systems (UMICS).

The anywhere/anytime/any means paradigm is becoming the major challenge in conceiving, designing, and releasing next-generation information systems. New technologies, like wi-fi networks and 3rd-generation mobile phones, are offering the infrastructure to conceive of information systems as ubiquitous information systems, that is, systems that are accessible from anywhere, at any time, and with any device. Ubiquity is not yet another buzzword pushed by emerging technologies, but is mainly a means to support new business models and encourage new ways of working. This new wave of UMICS will exploit the knowledge developed and deployed for conventional information systems, but will also need new concepts, models, methodologies, and supporting technologies to fully exploit the potentials of the enabling infrastructure and to be ready for the challenge.

Moreover, people need to move across organizational boundaries and collaborate with others within an organization as well as between organizations. The ability to query the company's distributed knowledge base and to cooperate with co-workers is still a requirement, but mobility brings new access scenarios and higher complexity. Therefore, some issues also arise about how to enable users to retain their ability to cooperate while displaced in different points of the enterprise, the role of context and location in determining cooperation, and the support for ad hoc cooperation in situations where the fixed network infrastructure is absent or cannot be used.

The approaches and technologies for supporting these new ways of working are still the subject of research. Nevertheless, they are likely to "borrow" concepts and technologies from a variety of fields, such as workflow systems, groupware and CSCW, event-based systems, software architectures, distributed database systems, mobile computing, ubiquitous information systems, and so on. A particularly interesting line of research is exploring a peer-to-peer paradigm enriched with sharing of abstractions, in which each network node is both a potential user and an information provider for the other members of the community.

June 2004

Luciano Baresi
Schahram Dustdar
Harald Gall
Maristella Matera

Organization

Program Chairs

Luciano Baresi	Politecnico di Milano, Italy
Schahram Dustdar	Vienna University of Technology, Austria
Harald Gall	University of Zurich, Switzerland
Maristella Matera	Politecnico di Milano, Italy

Program Committee

Wil M.P. van der Aalst	Eindhoven University of Technology, The Netherlands
Marios Angelides	Brunel University, UK
Farhad Arbab	CWI, The Netherlands
Boualem Benatallah	University of New South Wales, Australia
Christoph Bussler	DERI, National University of Ireland, Ireland
Stavros Christodoulakis	MUSIC/TUC, Crete, Greece
Fabio Casati	HP Labs, USA
Marlon Dumas	Queensland University of Technology, Australia
Josè Fiadeiro	University of Leicester, UK
Jim Gemmell	Microsoft Bay Area Research Center, USA
Dimitrios Georgakopoulos	Telcordia.com, Telcordia R&D, USA
Manfred Hauswirth	École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
Heiko Ludwig	IBM Research Labs, USA
Zakaria Maamar	Zayed University, UAE
Moira Norrie	ETH Zürich, Switzerland
Massimo Paolucci	CMU, USA
Wolfgang Prinz	Fraunhofer Institut FIT, Germany
Gustavo Rossi	LIFIA-Universidad Nacional de La Plata, Argentina
Thanh van Do	Telenor R&D, Norway
Florian Waas	Microsoft Corporation, Redmond, USA

Table of Contents

Process Modeling

Application Frameworks (II)

Paper on the Move

Moira C. Norrie

Institute for Information Systems,
ETH Zurich,
8092 Zurich, Switzerland
norrie@inf.ethz.ch

Abstract. We examine the properties and use of paper in everyday settings and discuss the motivation for retaining paper as an information medium. In particular, we consider the use of paper maps and guidebooks by tourists during city visits as an example of a mobile and collaborative environment. We then go on to present recent developments in technologies for digital paper and how they can be used to seamlessly integrate digital and printed information.

1 Introduction

Despite the emergence of digital technologies for capturing, managing, retrieving and processing information, paper persists as a fundamental resource for many human activities. It has therefore continued to be of great interest to the computer-supported cooperative work (CSCW) and human-computer interaction (HCI) communities. They are interested in studying both the properties of paper and how we use it, with a view to better understanding why it remains as a truly ubiquitous information medium. On the one hand, they want to use the results of their investigations to influence the design of digital technologies, with the ultimate aim of being able to replace paper. On the other hand, some researchers argue for the retention of paper and instead strive for means to interweave paper and digital media, enabling users to freely move back and forth between the printed and digital worlds.

In this article, we want to present the case for retaining paper and examine the move towards digital paper. While several projects in the past have sought to bridge the paper and digital divide, recent technological developments now make this a realistic option in some application domains and first business solutions are emerging. At the same time, current research opens up many exciting possibilities for the future in terms of whole new ways of interacting with paper.

Here we will focus on the potential use of digital paper in mobile environments. As a particular application domain, we will consider tourism which has been the subject of many projects in ubiquitous computing and mobile information systems. Typically, PDAs are used to provide tourists with, possibly personalised and context-dependent, digital maps and guides. Although many digital forms of tourist information are available nowadays, studies have shown that they are rarely used during visits, where paper continues to prevail in the form of maps and guidebooks. We will describe how technologies for digital paper could be used to support tourists and outline a demonstrator project for the Edinburgh festival that we are currently developing.

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 1–12, 2004.

We begin in Section 2 with a discussion of tourists on the move, examining typical activities and their use of paper. This is followed in Section 3 with a general examination of paper and the argument for its retention rather than replacement. In Section 4 we present various technologies for digital paper. Section 5 considers various issues of tools and infrastructures to support digital paper, in particular providing an overview of our work on developing a general framework and authoring tool. Section 6 then describes some applications that we have developed and also a tourist demonstrator currently under development. Concluding remarks are given in Section 7.

2 Tourists on the Move

Tourism is a domain with considerable potential for the use of mobile technologies and a number of research projects have developed PDA-based tourist guides, for example, Georgia Tech's Cyberguide [1], the Lancaster GUIDE system [7] and Xerox Parc's electronic guide [25]. However, commercially available digital guides have had little success and paper maps and guides continue to be considered the essential tourist accessories.

To understand why this is the case, it is necessary to carry out detailed studies of tourist activities during a visit to a city. In particular, it is important to know *how* maps and guidebooks are actually used by tourists. To date, most studies in tourism have been concerned with the effect of tourism rather than the tourist experience itself. Recently, Brown and Chalmers addressed this by carrying out an ethnographic study of tourists visiting cities [6]. Data collection in their study involved observing and taking video recordings of tourists as well as accompanying tourists on day excursions and generating video diaries of the day.

A tourist experience of a visit actually has three phases — pre-visit, visit and post-visit. In the pre-visit phase, the tourist may examine guidebooks, maps and web sites to gain background knowledge about the place that they will visit and plan activities. During the actual visit, the tourist will refer to maps and guidebooks to locate themselves, plan activities and find out more about a particular place of interest. The post-phase of the visit is about reminiscing and sharing the experience with family and friends. Photographs and videos play a central role in the post-visit phase, with occasional references to maps and guidebooks.

A major occupation of tourists is the planning of their activities, both before and during the visit. Unlike work plans, tourist plans are often not too detailed or specific. This not only enables tourists to adapt to any changes in circumstances such as the weather or special events, but also is due to the fact that a major part of the enjoyment of a holiday is discussing and planning what to do and when to do it. In fact, tourists generally travel in groups and there is a high degree of intra-group interaction and collaboration.

Figure 1 shows a typical scenario of two tourists collaborating around a map and a guidebook. They may be either simply finding out about their current environment or planning where to go. One tourist is holding a map and the other a guidebook. The map and guidebook are used in combination. Pointing is used to relate items in the map and guidebook to each other and also to the environment. In some cases, there may be a division of labour where one tourist uses the map to locate where they are, while the

Fig. 1. Tourists consulting Map and Guidebook

other tourist looks up something in the guidebook. Often tourists will work together with the map to try and establish their location, specific landmarks or the direction in which they should travel. To do this, frequently often the map is rotated to align it with the physical environment.

It is also important to appreciate that tourists do not simply use maps to plan specific routes. Often it is used as a general reference to the city in terms of areas within the city and their spatial relationship. Tourists may use it to help locate general areas that might be of interest — such as the old part of a city with many small streets or high areas which might provide good viewpoints. In conjunction with a guidebook, they may use it to identify areas with lots of restaurants and bars that might be a good place to wander in the evening. Sometimes tourists enjoy wandering freely, simply following streets that look to be of interest and only using a map or guidebook to find out more information about something they come across or to help them find their way back.

Annotations of maps and guidebooks may also play an important role. Routes or places of special interest may be marked on a map. During the pre-visit phase, selected entries in the guidebook may be highlighted or annotated with some textual comment. Markers may be placed in pages to indicate places not to be missed or information of special importance. Annotations such as simple highlighting, circling or underlining, may be used on both maps and guidebooks to keep a record of places that have been visited.

We have only been able to sketch some aspects of tourist activities involving maps and guidebooks that might influence the design of technologies for tourism. However, there are already a number of factors that we can highlight.

First, it is important to consider the effect of the physical form of technologies. Tourists spend a lot of time combining and comparing information. It is therefore important that they can easily switch back and forth between options and display related information such as map locations and descriptions simultaneously. In the case of the paper guidebook, several places can be bookmarked easily by placing fingers between pages and it is very simple and quick to jump between these pages. As we have seen above, tourists may display paper maps and guidebooks alongside each other simply by holding these next to each other and using pointing to link between the two.

The display option for electronic guidebooks is often a PDA. Comparison of information is rarely offered, mainly because of limited screen size. Further, the reduced size or coverage of maps can make it difficult for users to locate themselves relative to other areas of a city and to plan general directions of wandering. For this reason, some projects have opted for the use of graphical tablets to enable them to provide much richer information environments to the tourists. Clearly the problem here is that, with current technologies, graphics tablets are still relatively heavy if they are to be carried for a significant amount of time and they require the use of both hands. It is important to remember that tourists also want to have hands free, enabling them to carry shopping or hold the hand of an accompanying child or partner.

There are therefore requirements that any mobile devices be light and easy to place in pockets etc. Further, since collaboration is a key aspect of tourism, whether travelling in groups or just asking another person for directions or recommendations, another requirement is that more than one user can simultaneously view and interact with the display. Collaboration around current mobile digital devices is awkward because of both display size and required positioning for clear reading of the display.

Second, the applications developed for tourism need to take into account the nature of the experience. Tourists generally do not want to follow specified routes and fixed recommendations. Discovery is an important part of the experience, as is the social aspect of interacting with both companions and strangers in identifying landmarks and planning activities.

Further, tourists would like to be able to share information with each other. Often recommendations for places of interest, restaurants etc. are included in visitor maps, but these are based purely on advertising. There are many web sites for tourists that enable users to enter and access reviews and recommendations, but data tends to be entered by only a few dedicated tourists in the post-visit phase and access is mainly during the pre-visit phase. It would therefore be useful to have a system that enabled users to more easily enter and share reviews and recommendations during the actual visit.

3 Revisiting Paper

Paper consumption in all countries continues to grow despite earlier predictions of a rapid move towards a digital world, culminating in the paperless office. In fact, the average US office worker uses over 10'000 sheets of printing and copying paper each year [14].

A range of studies in the CSCW community have found that even with a wealth of new technologies and attempts to change work practices in accordance with these technologies, paper continues to be central to many activities in the work place, in particular those involving collaboration [21]. More recent studies of domestic environments, classrooms, museums and tourism have also reaffirmed the pervasiveness of paper. Generally, these studies have identified a range of uses and characteristics of paper that seem critical to a whole host of human activities including communication and collaboration.

Paper as a medium has many advantages over digital media in terms of how people can work with it, both individually and in groups. It is portable, cheap and robust. It can be folded, cut and torn. It is much more convenient to scan through a book by rapidly

flicking through pages than to browse a digital document. Paper also supports forms of collaboration and interaction that are difficult to mimic in current digital worlds. One of these mentioned in the previous section is the fact that current displays are best read when viewed straight on — and this makes it difficult for several users to simultaneously view an average or small screen. For this reason, a number of projects in the area of intelligent meeting rooms have focussed on the use of large wall-mounted displays. However, as we have seen with tourists, collaboration is also a factor in mobile environments.

In the world of paper, annotations are heavily used to enhance both reading and writing activities [13]. Annotation is some means of marking up a document so that it augments existing material. Annotations come in many forms and have a variety of uses. They may be private or public, permanent or transient, and formal or informal. Informal annotations often take the form of free text, but could also be sketches. For example, marginalia — the comments that we write in the margins when reading a paper or book — are informal annotations. The use of highlighting, underlining or circling to mark items of interest, as tourists often do with guidebooks or event programmes, is another form of commonly used informal annotation.

Formal annotations follow defined structures and conventions that enable them to be interpreted by other persons or by computer programs. Typographic markup for the editing of documents is one such example. Another example of formal annotation is that used in the semantic web community to markup digital data with metadata [5].

Both informal and formal annotations often provide a basis for communication and collaboration, whether it be part of a co-authoring activity or publishing annotations to aid the interpretation of data or comment on its quality.

There have been a number of proposals to provide annotation tools in the digital world either as part of general web infrastructure [11] or for specific domains or tasks. For example, SAM (Scientific Annotation Middleware) [20] is being developed to provide scientific researchers with a collaborative and cross-disciplinary working environment based around such annotations.

One study examined the task of writing document summaries and how users would annotate the document and take notes [18]. For this activity, readers not only wanted to highlight important items, but also to extract and re-order them according to the final structure of their summary. Annotations alongside the text heavily used references to structure outlines. The study not only examined how readers would use annotations, but also the differences between performing this task using only digital documents as opposed to using only paper. It was shown that there are many problems with digital annotation systems in terms of both inputting the actual annotations and also working with various documents alongside each other.

While there are many advantages to working with paper, clearly there are also advantages offered by digital media in terms of the storage, management, processing, retrieval and distribution of information. For example, although it might be easier to input free text annotations by writing them on paper, the ability to search for annotations and related information could be better supported if they are stored in digital form. For this reason, many researchers have now turned towards trying to integrate paper and digital media in an attempt to gain the best of both worlds. In the next section, we describe various projects and technologies that have been developed in this direction.

4 Move Towards Digital Paper

Over the last decade, there have been many efforts to bridge the digital and paper divide. In this section, we start with an overview of some past projects and then go on to discuss the state of the art technology, including both research and commercial solutions.

The Digital Desk was built at Xerox EuroPARC just over a decade ago as an attempt to enhance the physical desk and augment paper [16, 24]. A camera mounted above the desk was used to track a user's interaction with documents and additional digital information could be projected onto the documents. More recent projects that follow on from this idea of an augmented desk have extended the idea with other forms of tracking, such as real-time hand tracking and gesture recognition [12], and the support of specific forms of work activity such as architectural design [2]. Clearly, one of the main limitations of these approaches as far as mobile information environments are concerned is the fact that the documents are augmented only in the context of the physical desk and hence one of the main benefits of paper, namely mobility, is lost.

An alternative approach is to develop technologies based on paper itself and the tools that we use to work with it, particularly pens. DataGlyphs [10, 9] developed by Xerox are a means of encoding digital data on paper by printing images using patterns of forward and backward slashes representing zeros and ones. If the pattern is small enough it is not visible to the reader who sees only the image. However, a special scanning device can be used to read the encodings and activate digital services.

Many systems use some form of printed visible marks on papers as a means of creating links to digital material. For example, standard printed barcodes and a barcode reader have been used in a number of projects to link paper materials with electronic resources (for example [23]). The use of barcodes or other visible encodings has the advantage of clearly making links on paper visible to users. However, this turns out also to be a disadvantage if more than a few links are present and can be extremely disruptive to the reading activity.

Rather than relying on visible link encodings, a means of detecting pen position on paper can be used to both capture writing and activate links to digital content. For example, the mimio Xi [15] designed for whiteboards uses special pens and a high-resolution ultrasonic position detection device to track pen position and this enables handwritten information to be digitised automatically. Seiko has developed a smaller and more portable version called InkLink which can be used for documents up to A4 size. These technologies are essentially page-based rather than document-based and do not provide any mechanism that allows multiple pages to be associated with a document. Therefore while they are suited to the capture of writing, they are less suited to the creation and activation of links between paper documents and digital media.

Another way of detecting positions on paper has been developed by the Swedish company Anoto [4]. The technology is based on an almost invisible pattern of infrared-absorbing dots printed on paper and special digital pens which have a camera as well as a writing stylus. The pattern of dots encodes x-y positions in a vast virtual document address space. Camera images are recorded and processed in real time to give up to 100 x-y positions per second and enable a good representation of handwriting to be captured. Several pages of handwriting can be captured and stored within the pen before being transmitted to a PC. Several pens based on Anoto technology are now available

commercially and these include Nokia's Digital Pen, Logitech's io Personal Digital Pen and Sony Ericsson's Chatpen.

Currently, the focus of commercial products based on Anoto technologies has been on information capture. Hewlett-Packard has recently released a forms automation system based on this technology to reduce the costs of processing and correcting data input via forms in large enterprises such as insurance companies. Although the Anoto technology could be used to link paper with digital content interactively, the fact that current digital pens store data and transmit it only on demand, either by placing it in a cradle or activating special areas on paper, restricts this. However, for the purposes of our research we have been able to obtain a specially modified digital pen that sends data directly and which we can therefore use both for writing capture and general interaction.

From the above discussion, we can see that basically there are two different goals of technologies for digital paper. One is the capture of written information which leads to technologies to support *enhanced writing*. The other is the activation of links from paper to digital resources to enrich the reading activity and hence these technologies can be described as supporting *enhanced reading*. Although technologies such as Anoto have the potential to support both, currently the available pens support only the former. In contrast, within the European project Paper^{++} [19] the focus was on enhanced reading and the development of solutions that could be widely deployed in schools, homes etc. This meant that the cost of reading devices should be so low that they would almost be at the level of disposable technologies, i.e. only a few Euros. It was therefore necessary to devise a solution that avoids expensive optical components such as the cameras used in Anoto pens.

The Paper^{++} solution uses a grid of almost invisible barcodes printed with conductive ink to encode page number and x-y position within a page. A specially designed pen reads the information encoded in the barcodes by measuring the inductivity and this information is then decoded to obtain the position within a document. So far a few prototype pens have been developed and we have been able to use these in first user studies. However, there are still many open issues in terms of investigating alternative printing and reader technologies to obtain better performance and reliability, while keeping costs low.

Finally, we mention recent developments in display technologies that can be considered as a move from digital to paper in the sense of trying to develop digital technologies with some of the qualities of paper mentioned in the last section. There are considerable efforts to develop screens based on plastic substrates that are extremely lightweight and flexible.

The recently announced Sony LIBRIé e-Book Reader is a first commercial application of Philip's display based on E Ink's electronic ink technology. They claim that it provides readers with a truly paper-like reading experience and can be read in bright sunlight as well as dimly lit environments and viewed at any angle.

5 Software Infrastructure

We have seen that there are two aspects to the development of printing and pen technologies to support digital paper, namely supporting the writing activity and the reading

activity. Correspondingly, there are two aspects to providing a general software infrastructure for digital paper, namely authoring tools and a cross-media information server. Within the Paper^{++} project, we developed both and provide a brief overview of these in this section.

Our approach was to develop an open hypermedia system, called iServer, based on a generic link framework. Specific media types can be integrated through a plug-in mechanism as indicated in figure 2 which shows the main components of the iServer core and those of the plug-in developed for digital paper.

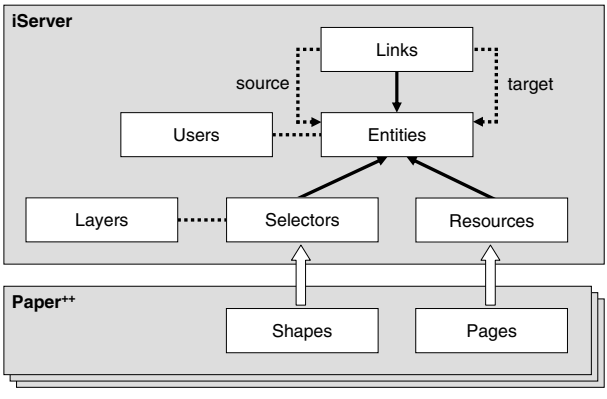


Fig. 2. Iserver

`Links` are first class objects which can have one or more sources and lead to one or more link targets. By modelling them as a subclass of `Entities`, we achieve full generality of allowing links over any type of entity objects, including links themselves. `Resources` represent entire information resources whereas `Selectors` provide a means of referring to elements within a resource. In the case of digital paper, `Pages` are resources and `Shapes` within a page are the selectors that may be used to define active areas linked to digital resources.

A resource may have any number of virtual `Layers` and each selector is associated with exactly one layer. This enables us to handle overlapping and nested link sources and targets. For example, in a printed document we may want to link an image as well as a piece of text that overlays part of the image, or individual words and phrases as well as an entire paragraph.

Layers are ordered and may be re-ordered dynamically by the application, which may also activate and de-activate individual layers at any time. For a given input, we calculate the set of selectors that could be activated. If there is more than one, then they must belong to different layers and, depending on application semantics, the appropriate layer and hence selector is activated. One could, for example, use this to provide a “zoom in" feature for images printed on paper.

Layer activation/deactivation and ordering may depend on various factors such as the history of activation, user task and context. For example, in an educational application, the links and information provided may depend on the age of the student.

While other hypermedia systems such as Hyper-G's Harmony browser [3] support overlapping link anchors, they do not provide any functionality to explicitly define the semantics of overlapping links. Further details of our layer concept together with the link model in general are given in [17].

So far we have developed plug-ins for XHTML, audio, video and still images as well as digital paper. It is therefore possible to link from paper to a section of a video or part of a web page as well as to entire digital resources. It is also possible to link from digital resources back to paper as links are bi-directional. At the moment, the targets of links to paper are shown by highlighted areas on PDF versions of the printed documents.

We have also developed an authoring tool to author links from paper to digital resources. The user first draws a shape on a page and then creates a link with the selected shape either as the anchor or the target of the link. The shape will be associated with a given layer and also the user who created it. By integrating a user management component into the core of the iServer we can control user access to both links and also their sources and targets.

There are a number of open issues associated with the authoring activity. One is how the user knows where the active areas on paper are. In the case of pre-authored links in a cross-media publishing framework, the areas could be highlighted in some way or we could make use of conventions adopted within the web community so that users have certain expectations about what information is linked. In some applications, such as an interactive map, we can ensure that some information is delivered for any position selected, possibly at different levels of granularity.

It is important to note that the software infrastructure was developed to be independent of particular position encoding and reader technologies. We are therefore able to use either the Paper++ technologies or the Anoto technologies and have demonstrator applications based on both. However, clearly, applications that require the authoring of annotations on paper as well as the authoring of links, need to be based on Anoto technologies since writing capture is not supported in Paper++.

6 Applications

Within the Paper^{++} project, several demonstrator applications were developed to illustrate the potential of integrating paper and digital media and carry out some preliminary user studies. We briefly describe some of them here to give an indication of the range of potential applications that could be supported by digital paper.

Based on a children's nature encyclopaedia published by Dorling Kindersley and the corresponding software application delivered on a CD-ROM, we designed a sample Paper^{++} version of part of the encyclopaedia linking areas of the book to digital information. Three different versions of the encyclopaedia (book only, book and CD-ROM, and the Paper^{++} version) were compared in a user study where groups of children were given specific tasks to solve using one of the encyclopaedias. Initial results of this evaluation are presented in [8].

Figure 3 shows two other demonstrator applications. On the left there is an interactive map of Zurich that uses an Anoto pen to detect positions within the map. Different layers are used to give city information at different levels of granularity, e.g. about

Fig. 3. Demonstrator Applications

the whole city, areas of the city, streets and specific landmarks. On the right, we show a prototype application developed for the Natural History Museum in London based on Paper $^{++}$ technologies. Children involved in the trials were issued with a paper worksheet on animal vision. During their visit to exhibits they had to fill in certain parts of the worksheet. After the visit, they would enter a designated investigate area within the museum where they could link from the worksheet to digital information. One layer augmented images with digital information in the form of detailed images and text, while another layer implemented a form of interactive game which tested the children's knowledge about the field of vision of various species.

We have also developed applications that use audio rather than visual displays. One was to support researchers reading publications where there are frequent citations to other publications. When the reader encounters a citation, often they will flick to the back of the paper to find out if the work is known to them. In our application, the reader could simply activate the citation with the pen and the title of the papers and authors would be read out to them. The speech was generated from a database of information about publications based on a general voice browser that we developed previously [22].

We are currently developing another demonstrator based around an interactive map and the use of Anoto technologies. The system is to support tourists visiting the Edinburgh festival, providing information about venues, events, restaurants, bars etc. in a given area of the city. The map will be part of a brochure that includes event information as well as note pages. Users can write comments and reviews on events in the note pages and these will be digitally captured and then can be shared with other users. Users can also enter information on restaurants etc. by creating a link from a location on the map to information about the restaurant authored on paper through a combination of selecting pre-printed option boxes and writing free text for comments.

As part of this project, we want to experiment with different forms of information display. In addition to technologies such as PDAs, we are particularly interested in experimenting with the use of audio, thereby relieving the user of having to carry any kind of display device. Instead the user will have an earpiece and receive information by speech and feedback by sound.

7 Concluding Remarks

We have presented the motivation for retaining rather than replacing paper as an information medium, especially in the case of mobile and collaborative environments. Research and developments in technologies for digital paper is currently a very active and rapidly evolving field. With a range of digital pens, and also business solutions based on these, now beginning to emerge in the marketplace, it raises lots of questions as to whether these technologies will be adopted and how they will be used.

Having developed the core software technologies required to support digital paper, we are now at the stage of beginning to be able to experiment with various applications and carry out user studies. Early applications tended to be relatively conservative in terms of the modes of linking and information delivery supported. As we gain experience and learn to observe our environment and how people work with paper, we are increasingly coming up with new ideas based on completely new forms of interaction.

Acknowledgements

Much of the work presented in this paper was brought to my attention as a result of participation in the European project Paper++, under the Disappearing Computer Programme (IST-2000-26130). I want to thank other members of the project for invaluable contributions and discussions. In particular, I want to thank my colleague Beat Signer at ETH Zurich who developed the general information infrastructure for integrating printed and digital information described in this paper and provided a great deal of source material, inclusive of photographs, used in this article.

References

1. G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A Mobile Context-Aware Tour Guide. *Wireless Networks*, 3:421–433, 1997.
2. D. Aliakseyeu and J.-B. Martens. Physical Paper as the User Interface for an Architectural Design Tool. In *Proceedings of INTERACT'2001, 8th Conference on Human-Computer Interaction*, Tokyo, Japan, July 2001.
3. K. Andrews. *Browsing, Building, and Beholding Cyberspace: New Approaches to the Navigation, Construction, and Visualisation of Hypermedia on the Internet*. PhD thesis, Graz University of Technology, 1996.
4. Anoto AB, `http://www.anoto.com`.
5. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.
6. B. Brown and M. Chalmers. Tourism and Mobile Technology. In *Proc. 8th European Conf. on Computer Supported Cooperative Work (ECSCW 2003)*, Helsinki, Finland, September 2003.
7. K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a Context-Aware Electronic Tourist Guide: Some Issues and Experiences. In *Proc. CHI 2000*, The Hague, September 2000.
8. D. Frohlich, E. Tallyn, N. Linketscher, B. Signer, and G. Adams. Reading Augmented Paper: Children's Experiences from a Simulation Study. Technical Report HPL-2001-308, HP Labs, 2001.

9. D. L. Hecht. Printed Embedded Data Graphical User Interfaces. *IEEE Computer*, 34(3):47–55, March 2001.
10. C. Kafka. DataGlyphs Bridge Paper and Digital Worlds. *Docu World*, 1998.
11. J. Kahan, M.-R. Koivunen, E. Prud'Hommeaux, and R. R. Swick. Annotea: An Open RDF Infrastructure for Shared Web Annotations. In *Proceedings of WWW10*, Hong Kong, May 2001.
12. H. Koike, Y. Sato, and Y. Kobayashi. Integrating Paper and Digital Information on EnhancedDesk: A Method for Realtime Finger Tracking on an Augmented Desk System. *ACM Transactions on Computer-Human Interaction*, December 2001.
13. C. C. Marshall. Annotation: From Paper Books to Digital Library. In *Proceedings of DL'97, 2nd ACM International Conference on Digital Libraries*, Philadelphia, USA, July 1997.
14. MDF Systems. Things to Know about Paper Consumption. `http://www.mdfsystems.com/artman/publish/article\_42.shtml`.
15. mimio Xi, `http://www.mimio.com/meet/xi/`.
16. W. Newman and P. Wellner. A Desk Supporting Computer-Based Interaction with Paper Documents. In *Proceedings of ACM CHI'92, Conference on Human Factors in Computing Systems*, May 1992.
17. M. C. Norrie and B. Signer. Switching over to Paper: A New Web Channel. In *Proceedings of WISE 2003*, Rome, Italy, December 2003.
18. K. O'Hara and A. Sellen. A Comparison of Reading Paper and On-Line Documents. In *Proceedings of ACM CHI'97, Conference on Human Factors in Computing Systems*, March 1997.
19. Paper^{++} Project, IST-2000-26130, Disappearing Computer Initiative, `http://www.paperplusplus.net`.
20. Scientific Annotation Middleware (SAM), `http://www.emsl.pnl.gov:2080/docs/collab/sam/`, 2001.
21. A. J. Sellen and R. Harper. *The Myth of the Paperless Office*. MIT Press, November 2001.
22. B. Signer, M. C. Norrie, P. Geissbuehler, and D. Heiniger. Aural Interfaces to Databases based on VoiceXML. In *Proceedings of VDB6, 6th IFIP Workshop on Visual Database Systems*, Brisbane, Australia, May 2002.
23. I. Siio, T. Masui, and K. Fukuchi. Real-world Interaction using the FieldMouse. In *Proceedings of UIST'99, 12th Annual ACM Symposium on User Interface Software and Technology*, November 1999.
24. P. Wellner. Interacting with Paper on the DigitalDesk. *Communications of the ACM*, 36(7), July 1993.
25. A. Woodruff, P. Aoki, A. Hurst, and M. Szymanski. Electronic Guidebooks and Visitor Attentio. In *Proc. 6th. Int. Cultural Heritage Informatics Meeting (ICHIM)*, Milan, Italy, 2001.

A Natural Language Model for Managing TV-Anytime Information in Mobile Environments

Anastasia Karanastasi, Fotis G. Kazasis, and Stavros Christodoulakis

Lab. Of Distributed Multimedia Information Systems / Technical University of Crete, (MUSIC/TUC),
University Campus, Kounoupidiana, Chania, Greece
{allegra, fotis, stavros}@ced.tuc.gr

Abstract. The TV-Anytime standard describes structures of categories of digital TV program metadata, as well as User Profile metadata for TV programs. We describe a natural language model for the users to interact with the TV-Anytime metadata and preview TV programs from their mobile devices. The language utilizes completely the TV-Anytime metadata specifications and it can accommodate future metadata extensions. The interaction model does not use clarification dialogues, but it uses the user profiles to rank the possible answers in case of ambiguities, as well as TV-Anytime Metadata information and ontologies with information concerning digital TV. We describe an implementation of the language that runs on a PDA and a mobile phone and manages the metadata on a remote TV-Anytime compatible TV set.

1 Introduction

The number of digital TV channels has increased dramatically the last few years, and several industrial sectors and content producing sectors are active in defining the environment in which the TVs of the future will operate.

The TV-Anytime Forum is an association of organizations which seeks to develop specifications to enable audio-visual and other services based on mass-market high volume digital storage in consumer platforms - simply referred to as local storage [1]. These specifications target interoperable and integrated systems, from content creators/providers, through service providers, to the consumers and aim to enable applications to exploit the storage capabilities in consumer platforms. The basic architectural unit is an expanded TV set (known as a Personal Digital Recorder – PDR) capable of capturing digital satellite broadcasts according to user interests as they are described in his profile and storing them into large storage devices. The current TV-Anytime standard specifications define the structures for the metadata that can be used to describe TV programs and broadcasts, as well as for the metadata that can be used to describe the user profile. Expanded versions of the TV-Anytime architecture foresee also last mile TV-Anytime servers, Internet connection of the TV set and mobility aspects. Mobile devices (mobile phones, PDAs, etc.) in the TV-Anytime architecture can be used by a user to communicate with the home TV set not only for viewing TV programs, but also for managing the contents of the TV set (like previewing its contents, searching for content, deleting content that has been recorded for him by the TV set, etc.) and for managing his profile preferences [3].

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 13–27, 2004.

There is a strong need for new interface paradigms that allow the interaction of naïve users with the future TV sets in order to better satisfy their dynamic preferences. The usual pc-based interfaces are not appropriate to interact with mobile devices (like mobile phones or PDAs) or with TV sets. Natural language interfaces are more appropriate interface styles for naïve users, and they can also support voice-based interactions for mobile devices.

In this paper we present a model for natural language interactions with a TV set in an environment that follows the TV Anytime specifications, both for the TV program metadata as well as for the user profile metadata. The natural language interactions are used to preview programs or summaries of programs as well as to completely manage the metadata and the programs that the TV set keeps for the user. In addition we describe an implementation of this TV-Anytime compatible natural language interaction model that works on a PDA and a mobile phone, which communicates with the TV-Anytime TV set for managing its programs and metadata and also allowing the previewing of TV programs from the mobile device.

Much research has been published in the area of natural language interfaces to interactive TV based information systems [4], [5], [6]. A well-known problem with the natural language interfaces is that user interactions may be ambiguous. Ambiguity in the natural language interfaces is a serious problem and most systems proposed in the literature lead to lengthy clarification dialogues with the user to resolve ambiguities [12]. Our own model also includes the possibility for ambiguities in the user interaction. However, our environment is more concrete since the structure imposed by the TV-Anytime specifications for the metadata limit the possibilities for ambiguities. Unlike the previous systems we do not resolve the ambiguities with clarification. Instead we can take advantage of the TV-Anytime user profile specifications in order to rank the possible interpretations and present to the user at the top position the one with the highest ranking.

The best-known dialogue systems that have been developed for digital TV and mobile environments are related to the MIINA project [8] and the Program Guide Information System of NOKIA [9]. In the context of MIINA project, a system has been developed for information retrieval from the set-top-box Mediaterminal of NOKIA. The user is allowed to insert queries for TV programs, channels, program categories and broadcast time, using a natural language. However, the natural language interaction in this model is rather simple since it is only related to the information provided by a traditional TV-Guide. The Program Guide Information System is an electronic call-in demo application offering information about television programs over the phone by allowing the user to converse with the system in natural language sentences. This system is not based on TV-Anytime metadata structures for describing the programs or the user profiles. The scope of the interaction does not include any management of the stored content or the user profiles.

The main differences between those systems and the one described in this paper is that the present system uses the TV-Anytime content and consumer metadata specifications for a complete management of TV programs and user profiles, and that the system uses additional information that exists in the TV-Anytime User Profile in order to avoid length clarification dialogues and help the user to get the most relevant answers at the top of the result list.

In section 2 of this paper the natural language model for digital TV environment is presented, along with the functionality provided and the representation of the information that the system collects from the user's input. In section 3 we present the algorithm for resolving the ambiguities instead of using clarification dialogues. In section 4 there is the analysis of the system architecture and of the modules that constitute it. Section 5 presents the implementation environment of the system and of the applications from the client side. In section 6 we present an example of a user's utterance and the actions taken by the system in order to satisfy the user's request. Finally section 7 presents the results of the system's evaluation based on user experiments and section 8 concludes by summarizing the content of this paper.

2 The Natural Language Model for the Digital TV Environment

The design of the proposed model fulfills a certain number of requirements that also determine its final functionality. Figure 1 presents the satisfied functional requirements in the form of use cases.

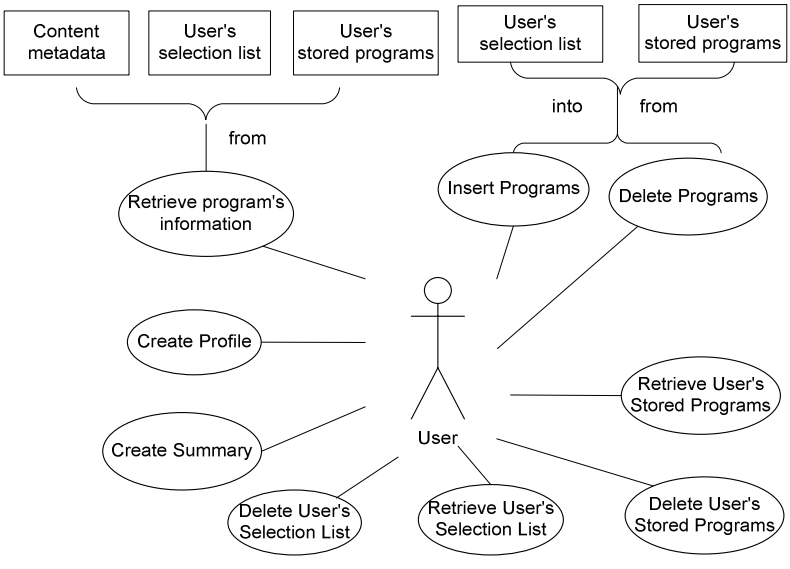


Fig. 1. The use cases that the natural language system was designed to satisfy

A collection of sub-phrases was implemented and combined to constitute the user's utterance. The categories of these sub-phrases are Introduction phrases, Search phrases, Target phrases, Temporal phrases and Summary phrases.

This clustering was used to model the functionality provided by the TV-Anytime specifications. The minimal utterance consists of either a combination of an introduction phrase and a search phrase, or a combination of an introduction phrase and a target phrase. The summary phrases are individual and must be combined only with temporal phrases.

The design of the system was made taking into account two main goals. The first goal was to determine how the system should behave according to the information gathered by the user's input utterance. Based on this behavior, the second one was, to design and define the roles of each module of the system.

We created a structure for the representation of the information gathered by the user's utterance. This structure consists of three parts namely Element, Element Type and Element Value. The first structure part (Element) is used to differentiate the TV-Anytime metadata information (modeled as *TVA-properties*) from the information that directs the system to the correct management of the user's input (modeled as *flags*). The TV-Anytime information about date and time is modeled as *temporal* element. The second structure part (Element Type) is used in order to further specialize the aforementioned information and to obtain its corresponding value (the third structure part), from the user's utterance. When a user inserts an utterance into the system, it generates a feature structure [7] that follows the structure of the model.

The 'flags' element takes its value from the introduction phrases and the target phrases. The 'TVA-properties' element takes its value from the search phrases and the summary phrases and the 'temporal' element from the temporal phrases.

The feature structure can contain one, two or three elements. These three types of feature structure are:

Type 1: Markers

- E.g. I want to see what is in my selection list

This utterance consists of an **introduction phrase** (I want to see what is) and a **target phrase** (in my list). The type action of the element markers takes the value 'retrieval' (information that comes from the introduction phrase) and the element type target takes the value 'list' (information that comes from the target phrase).

Type 2: Markers – TVA-Properties

- E.g. I would like you to show me movies starring Mel Gibson

This utterance consists of an **introduction phrase** (I would like you to show me) and a **search phrase** (movies starring Mel Gibson). The type action of the element markers obtains the value 'retrieval' (information that comes from the introduction phrase), the type genre of the element TVA-properties obtains the value 'movies', the type creator takes the value 'actor' and the type name takes the value 'Mel Gibson' (information that comes from the search phrase).

Type 3: Markers – TVA-Properties – Temporal

- E.g. Insert English spoken mystery movies broadcasted at midnight into my selection list.

This utterance consists of an **introduction phrase** (Insert), a search phrase (English spoken mystery movies broadcasted), a **temporal phrase** (at midnight) and a **target phrase** (into my selection list). In this case, the type action of the element markers takes the value 'insert' (information that comes from the introduction phrase), the type target takes the value 'list', from the target phrase, the type genre of the element TVA-properties takes the values 'mystery' and 'movies', the type language takes the value 'English' and in the feature structure there is also the type dissemination value, but without value. This information comes from the search phrase. Also, in the element

temporal, the type time takes the value '24' and the type time indicator takes the value 'am'. This information also comes from the search phrase.

The TV-Anytime metadata model integrates specifications for content metadata used to describe digital TV Programs in terms of various features and specifications for user preferences used to filter program metadata. The "Filtering and Search Preferences" are used to store the preferences of the users in terms of content features so that a digital TV system can filter content and provide personalization services. The FilteringAndSearchPreferences Descriptor Scheme (DS) specifies a user's filtering and/or searching preferences for audio-visual content. These preferences can be specified in terms of creation-, classification- and source-related properties of the content. The FilteringAndSearchPreferences DS is a container of CreationPreferences (i.e. Title, Creator), ClassificationPreferences (i.e. Country, Language) and SourcePreferences (i.e. DisseminationSource, DisseminationLocation). The BrowsingPreferences DS is used to specify a user's preferences for navigating and accessing multimedia content and is a container of SummaryPreferences (i.e. SummaryType, SymmaryTheme, SummaryDuration) and PreferenceCondition (i.e. Time, Place).

For the first three system functions (referring to the system functionality presented in fig. 1 above), namely the retrieval of the personalized content metadata, the management of the personalized content and the creation of the user's profile, the utterance contains in its body one or more search phrases. The system will create a TV-Anytime XML document, compatible with the UserIdentifier and the FilteringAndSearchPreferences Descriptor Schemes of the TV-Anytime metadata specification. For the forth system function, the definition of the user's preferences for the characteristics of an audio-visual content summary, the system constructs a TV-Anytime XML document, compatible with the UserIdentifier and the BrowsingPreferences Descriptor Schemes, with values in the fields of the SummaryPreferences and the PreferenceCondition (for handling the time of the summary delivery).

A user's selection list is a list of information about program's metadata that the system recommends to the user based on his preferences expressed either at his TV-Anytime profile or directly by him. Every program in this list has a status. The four possible values of this status are: undefined, toBeRecorded, recorded, toBeDeleted.

If the user wants to manage the contents of his selection list or the list of his stored contents the actions that take place are represented in figure 2:

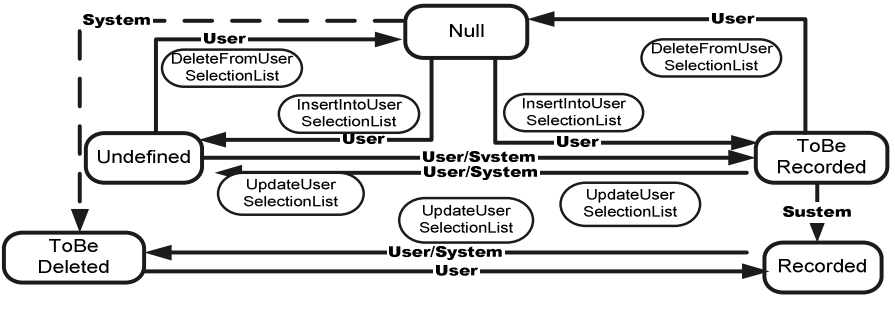


Fig. 2. The state machine for managing the status of a program in the user's selection list

The user may know neither the contents nor their status of his selection list, so the system undertakes to execute the proper function on behalf of the user. The system supports the functions that are described in figure 3:

Command	Target	Action
Insert	List	InsertIntoUserSelectionList with status Undefined
Insert	PDR	If the programs were already in the selection list, then UpdateUserSelectionList with status ToBeRecorded If they were not, then InsertIntoUserSelectionList with status ToBeRecorded
Insert	-	InsertIntoUserSelectionList with status Undefined
Delete	List	DeleteFromUserSelectionList
Delete	PDR	UpdateUserSelectionList with status ToBeDeleted
Delete	-	If they were into the selection list with status Recorded then UpdateUserSelectionList with status ToBeDeleted If they were into the selection list with any status but Recorded then DeleteFromUserSelectionList
StorelRecord	-	InsertIntoUserSelectionList with status ToBeRecorded

Fig. 3. Functions supported by the system for managing user's selection list

3 Resolving Ambiguities – The Algorithm

If the user does not specify in his utterance the TV-Anytime category he is referring to, the system tries to resolve any ambiguities by following specific steps. First, it collects word by word the sub-phrase with the ambiguities and creates a table of these words (refer to Vector of keywords in figure 4 below). Then -by using a stop list containing words with no semantic values, such as prepositions, pronouns, conjunctions, particles, articles, determiners and so on- it eliminates the words that match any word in this list. However, the system retains the existence of an 'and' or an 'or' for the optimum cross-correlation of the results. Then the system gathers the remaining words and by filtering them through its database or other existing ontologies it returns a TV-Anytime XML document that is compatible with the FilteringAndSearchPreferences DS. This descriptor scheme is the one that is used for the semantic resolving of the words. Finally, the system checks for matches in any existing TV-Anytime user's profile. This flow is represented in figure 4. The classification of the results is important in order to prioritize the user's preferences.

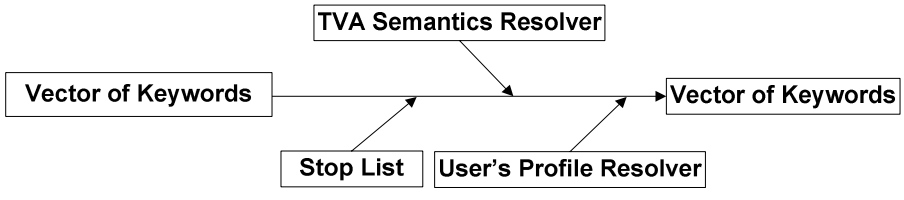


Fig. 4. The system tries to give TV-Anytime semantics to the words with the ambiguities

Based on the form of the sub-phrase with the ambiguities the algorithm acts as follows:

```
Subroutine: semantic resolver
      for every word with ambiguity check the stop list
      if there is no match.
            check for TVA semantics
            if there are TVA semantics add a specific weight value
              if there is a user profile
                        check from semantics from profile
                    if there is a match
                            add a weight value based on the preference value from profile to the TVA  semantics
      if there is a match
            cut the word
      return

check the words with ambiguities for an 'and' or an 'or'
if there is no match
    call semantic resolver
      check for same strings that contain words with the same semantic
      cut the rest
      search for results
     rank them based on the weight value
else
      call semantic resolver
      search for results
      group the results based on the same TVA semantic
         rank the results based on the weight value
```

An example for the case where there is no match between the stop list and the words with the ambiguities would be the sub-phrase **'… with Nicole Kidman'**. Suppose that the TV-Anytime Semantics Resolver gives the semantics shown in table 1.

Table 1. Semantics from the TV-Anytime Semantics Resolver for the words 'Nicole', 'Kidman'

Creator	Role
Nicole Rosselle	Actor
Mary Kidman	Director
Nicole Kidman	Producer
Nicole Kidman	Actor

Suppose now that the User's Profile Resolver gives the semantics with the corresponding reference value as the one declared in the TV-Anytime User's Profile, shown in table 2.

Table 2. Semantics based on the TV-Anytime User's Profile for the words 'Nicole', 'Kidman'

Creator	Role	Pval
Nicole Rosselle	actor	80
Nicole Kidman	producer	50
Nicole Kidman	actor	70

The results in this case, based on the above, are shown in table 3.

Table 3. Results after resolving the ambiguity of the sub-phrase '… with Nicole Kidman'

	Creator	Role
1	Nicole Kidman	Actor
2	Nicole Kidman	Producer
3	Nicole Rosselle	Actor
4	Mary Kidman	Director

An example for the case where there is a match between the stop list and the words with the ambiguities and there is an 'and' or an 'or' between the words with the ambiguities would be the sub-phrase **'… with Frank and English'**. Suppose that the TV-Anytime Semantics Resolver gives the semantics shown in table 4.

Table 4. Semantics from the TV-Anytime Semantics Resolver for the words 'Frank', 'English'

Creator	Role	Language
Frank English	actor	english
Holger Franke	director	
Frank… 66 results	various	

Suppose now that the User's Profile Resolver gives the semantics with the corresponding reference value as is the one declared in the TV-Anytime User's Profile, shown in table 5.

Table 5. Semantics based on the TV-Anytime User's Profile for the words 'Frank', 'English'

Language	PVal
english	80

The results in this case, based on the above, are shown in table 6.

Table 6. Results after resolving the ambiguity of the sub-phrase '…with Frank and English'

	Creator	Role	Language
1	Frank / English	actor/actor	-
2	Franke / English	director/actor	-
3	Frank	actor	English

An example for the case where there is a match between the stop list and the words with the ambiguities but there is not 'and' or an 'or' between the words with the ambiguities would be the sub-phrase **'… with Frank in English'**. Suppose that the TV-Anytime Semantics Resolver gives the semantics in table 7.

Suppose now that the User's Profile Resolver gives the semantics with the corresponding reference value as this is declared in the TV-Anytime User's Profile, shown in table 8.

Table 7. Semantics from the TV-Anytime Semantics Resolver for the words 'Frank', 'English'

Creator	Role	Language
Frank English	actor	english
Holger Franke	director	
Frank… 66 results	various	

Table 8. Semantics based on the TV-Anytime User's Profile for the words 'Frank', 'English'

Language	PVal
English	80

The results in this case, based on the above, are shown in table 9.

Table 9. Results after resolving the ambiguity of the sub-phrase '…with Frank in English'

	Creator	Role	Language
1	Frank	actor / language	English
2	Frank / English	actor/actor	-
3	Franke / English	director/actor	-

4 System Architecture

The complete architecture of the system is presented in figure 5. This architecture follows a multi-tier approach and consists of three tiers. The lowest tier handles the metadata management. The middleware tier includes all the logic for interfacing the system with the outside world. The application tier enables the exchange of information between the server and heterogeneous clients through different communication links.

The user uses a wireless device (mobile phone, PDA) and inserts an utterance that is forwarded to the **ChartParser module**. The ChartParser module consists of the JavaChart parser [10] that creates a feature structure with the information from the user's input. There exist two lexicons, the stem lexicon, which contains the stems of the words used in this language, and the words that help to give the right values to the TV-Anytime categories, and the mini lexicon, which contains the endings of the words. Finally, there is the grammar that follows a unified-based formalism.

The **Dialogue Manager** acts as the core module of the system and is responsible for communicating with all the other system modules. It takes as input a list of feature structures from the chart parser and the user's data from the application that he/she is using. It checks for specific properties in order to eliminate the list of the feature structures and, in the case there are any ambiguities, it passes to the Ambiguities Resolver module the list of the words with the ambiguities. Finally, it creates a structure with information about the action, the target and the TV-Anytime XML document from the user's input.

The **Relational Database** of the system contains the TV-Anytime Metadata information, as well as a number of **ontologies** (alternative classification schemes to represent genre hierarchies and creator roles as well as more advanced semantic descriptions following the Semantic Part of the MPEG-7 MDS), with information concerning digital TV.

The **Relational DBMS** manages the transactions, utilizes a **Java API** (implemented for the extraction of the functionality for filtering, retrieval and summariza

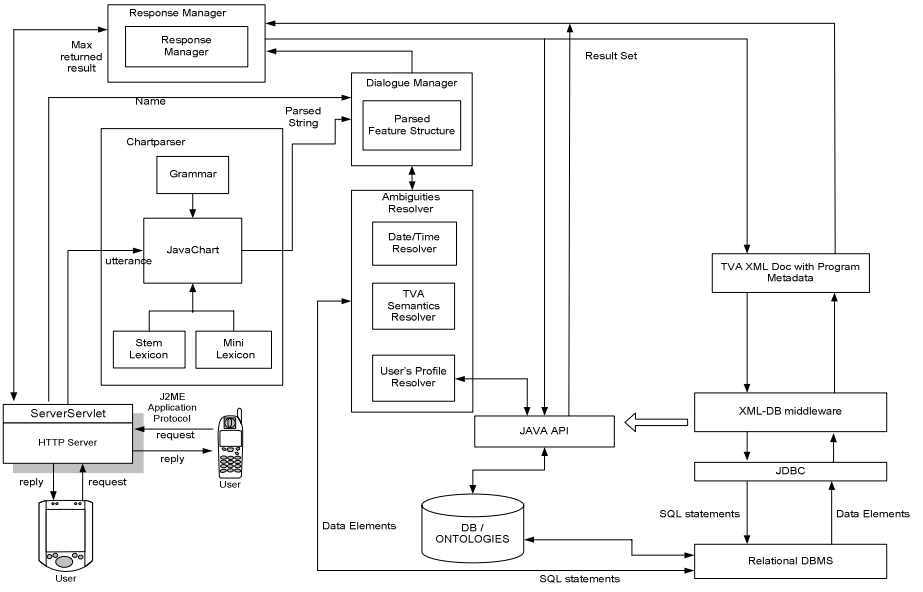


Fig. 5. The Natural Language System Architecture

tion) and cooperates with the **XML-DB middleware**. The **XML-DB middleware** is a set of software components responsible for the management of the TV-Anytime XML documents and the correspondence of the TV-Anytime Metadata XML schema with the underlying relational schema.

The **Ambiguities Resolver** module consists of three modules that are responsible for the resolution of different kinds of ambiguities. The Date/Time Resolver is the component that converts the temporal phrases in a TV-Anytime compliant form. The TVA Semantics Resolver communicates with the relational DBMS and is responsible to attach TV-Anytime semantics to the words with the ambiguities. The User's Profile Resolver filters the list of the words from any existing user's profile and returns a FilteringAndSearchPreferences XML document with values from the corresponding TV-Anytime categories. Finally, it passes this document to the Response Manager module.

The **Response Manager** module interacts with the system's database, by providing it the structured information, executes the appropriate functions, retrieves the results and classifies them accordingly. Then, it creates a message and adds it to any existing result list.

5 Implementation Environment

For the system's server side, the implementation platform consists of the MySQL Platform [13]. MySQL is an open source relational database management system, which is reliable and easy to use. The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces. The implementation of the server was based

on Java 2 and the class files were compiled using JDK1.4.1. [14]. The parsing of the user's utterance was made by the use of the JavaChart parser [10], a chart parser written in Java. For the implementation of the communication between the server and the client, we have exploited the JAVA Serlvet technology in the server side by developing a servlet that acts as the interface between the user client and the database server or the PDR interface. This servlet was locally deployed for testing on an Apache Tomcat v4.0.1 server and the class files were compiled using JDK1.4.1.

Two cases are considered related to the wireless device used on the system's client side. The first one is the application that runs on any Java-enabled mobile phone device and the second is the application that runs on a PDA device.

For the client (mobile device) the implementation platform consists of the Java 2 Platform, Micro Edition (J2ME). The platform addresses the large, rapidly growing consumer space, from pagers till TV set-top boxes. The Connected Limited Device Configuration (CLDC) has been used for the limited capabilities of the mobile phone. The Mobile Information Device profile (MIDP) is an architecture and a set of Java libraries that create an open, third party application development environment for small, resource-constrained, devices MIDlets. By sub classing the MIDlet class, an interface between the application and the application management software on the device is defined.

For the second implementation of the client side we used JEODE Java Runtime Environment [15] for the PDA device client. The JEODE runtime environment supports the CDC/Foundation Profile and the Personal Profile J2ME specifications that support implementations in PersonalJava and EmbeddedJava. The PersonalJava platform is a Java platform optimized for the requirements and constraints of wireless devices. The PersonalJava application-programming interface (API) is a collection of packages, classes and methods defined by a high-level specification.

For both cases of the remote access application, the client establishes an http connection with the server.

6 System Flow

In this section we describe an example of a user's utterance and the way that this input flows inside the modules of the system.

We consider the user's utterance

- 'Are there any movies with Frankie Muniz after 5 pm'

The ChartParser module obtains the utterance, as a string, from the wireless device. The output from the parser is the feature structure presented below:

```
[[ properties: [ keyword:  [ 1: Muniz]
                           [ 0: Frankie]]
               [ genre1: movies]]
 [ temporal:   [ timeind:  pm]
               [ after: [ time : 5 ]]
 [ markers:    [ act: ret ]]]
```

The feature structure has a type keyword with values 'Frankie' and 'Muniz'. For that the Dialogue Manager module first creates the vector of the keywords that have ambiguities (Frankie, Muniz) and then parses these words from the stop list and checks if there is an 'and' or an 'or' among them. The stop list does not remove any of the words

and the Dialogue Manager passes the vector with the words to the Ambiguities Resolver module. The TV-Anytime Semantics Resolver (after communicating with the metadata management system) returns a vector from vectors with the semantics for every word:

[[ACTOR, Frankie Muniz]]

The Dialogue Manager module collects the information provided by the Ambiguities Resolver module, plus the personal data (name, password) from the user's input and creates a structure of the following form:

String action = retrieval	**String target =** null	XML doc

The XML document produced for the utterance applied is presented in figure 6.

In the previous example there is no information provided by the User's Profile Resolver. The Response Manager module first checks the fields 'action' and 'target' of the structure. Since the action is a retrieval with no target, the search should be done in the metadata of the TV-Anytime server. The XML-DB Middleware inserts the TVA XML doc into the relational database with a temporary value in the field Name of the UserIdentifier. The Response Manager module uses a procedure to match the information with the metadata and returns the matching programs with a weight value that represents the relevance between the information and the result. The results, in the form of programs' titles, are presented to the user with a proper message.

```
<tva:UserDescription>
        <tva:UserPreferences>
                <mpeg7:UserIdentifier>
                        <mpeg7:Name>__temp_faspmatching_</mpeg7:Name>
                </mpeg7:UserIdentifier>
                <mpeg7:FilteringAndSearchPreferences preferenceValue="100">
                        <mpeg7:CreationPreferences preferenceValue="100">
                                <mpeg7:Creator preferenceValue="100">
                                <mpeg7:Role href="urn:mpeg:MPEG7RoleCS:ACTOR" />
                                <mpeg7:Agent  xsi:type="mpeg7:PersonType">
                                <mpeg7:Name>
                                <mpeg7:GivenName>FrankieMuniz</mpeg7:GivenName>
                                </mpeg7:Name>
                                </mpeg7:Agent>
                                </mpeg7:Creator>
                                <mpeg7:DatePeriod preferenceValue="100">
                                        <mpeg7:TimePoint>2003-08-
21T17:00:00</mpeg7:TimePoint>
                                        <mpeg7:Duration></mpeg7:Duration>
                                </mpeg7:DatePeriod>
                        </mpeg7:CreationPreferences>
                        <mpeg7:ClassificationPreferences preferenceValue="100">
                                <mpeg7:Genre href="urn:mpeg:TVAnytime_v0.1ContentCS:6"
preferenceValue="100">
                                        <mpeg7:Name>Movies</mpeg7:Name>
                                        <mpeg7:Definition>Film, Cinema</mpeg7:Definition>
                                </mpeg7:Genre>
                        </mpeg7:ClassificationPreferences>
                        <mpeg7:SourcePreferences preferenceValue="100" />
                </mpeg7:FilteringAndSearchPreferences>
        </tva:UserPreferences>
</tva:UserDescription>
```

Fig. 6. The XML doc created by the Dialogue Manager module with the user's input information

The user can set his preferences for the information about the programs metadata that he would like to obtain as a result while interacting with the system. This information can be also enriched with the program's broadcast date and time, broadcast channel and synopsis; something that mainly depends on the resource capabilities of the device that the user uses to communicate with the natural language system.

7 Evaluation

This section presents some preliminary results of the system's evaluation. The evaluation has been based on a user experiment that was performed by ten laboratory users. All the users had previous experience using computers and graphical interfaces. Nevertheless none of them had ever used a natural language system before. There were three main tasks used in the evaluation. The first one was for the users to use the system to define their user profile, the second one to interact with the system by using a set of utterances with no ambiguities and the third one to interact with the system by using a set of 10 utterances with ambiguities.

The functionality provided by the natural language system was also provided by the use of alternative menu-driven user interfaces for wireless devices (mobile phone, PDA). Based on the preliminary results it becomes clear that the end users found the system easier to use with the natural language interface than with the traditional user interfaces. The natural language interface was shown to provide an easy way to specify TV-Anytime structures without complex navigation between screens. For utterances that showed no ambiguities the system proved to fully exploit the structured TV-Anytime model capabilities for filtering in order to retrieve the qualified ranked results.

In the case of the utterances with ambiguities, we have considered 100 interactions in total. All the utterances contained a sub-phrase with no declaration of any TV-Anytime categories. For example, two of the utterances considered were:

- I want comedies with Tom English
- Record movie from Russia with love

In order to evaluate our approach for the use of the existing user profiles in order to rank the possible answers in case of ambiguities we have considered different types of user profiles so that the similarity between the user's preferences in these profiles and the specific input utterances to be near 0,5.

Diagram 1 shows the number of interactions per rank position for the cases that the system has either used the user profile in order to better rank the results or not. When the system uses the user profile to rank the results, we get about 90% of the exact results in the first 20 positions of the ranked resulting lists. This percentage varies according to the result list. In the first 5 positions we get the 65% of the exact results. When the system does not use the user profile to rank the results, we get about 80% of the exact results in the first 20 positions of the ranked resulting lists. In the first 5 positions we get the 40% of the exact results.

Diagram 2 shows the number of interactions in the top percentage of the total number of results. When the system uses the user profile to rank the results we get about 90% of the exact results in the 40% of the total number of results. In the case that the system does not use the user profile we get about 90% of the exact results in the 70% of the total number of results.

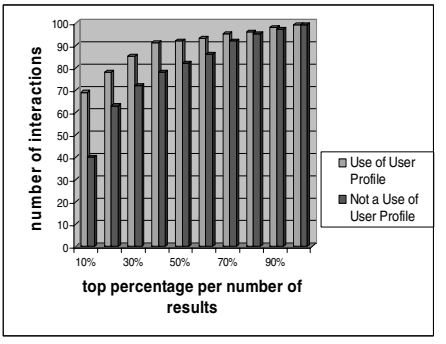


Diagram 1. Number of interactions per rank position

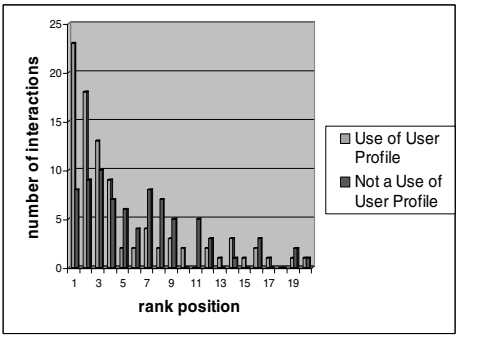


Diagram 2. Number of interactions per top percent age of the total number of results

8 Summary

In this paper we described the design and the implementation of a natural language model in digital TV and mobile environments that allow the user, by using a variety of devices to communicate with the Personal Digital Recorder of his TV set, in order to perform a variety of functions.

The model was developed to be compatible with the TV-Anytime specifications and manages the information of the content metadata categories for TV programs and the user's profile metadata. It is expandable to future additions in the TV-Anytime metadata specifications.

In order to satisfy the proposed functionality a dialogue model was developed that contains: Introduction phrases, to define the functionality, Search phrases, to define the TV-Anytime information, Target phrases, to define where each of the functions is targeting, Temporal phrases, to define phrases about date and time and Summary phrases, to define summaries with audio/visual content.

In the proposed natural language model, an utterance can contain one or more entities. These entities distinguish the information, which describes TV-Anytime content metadata information from the information that concerns the functions for its management.

In order to resolve possible ambiguities contained in the user's utterance, the system firstly checks for specific words in the utterance, then searches existing ontologies and attaches semantics to every word that appears with ambiguity and finally checks the TV-Anytime user's profile to attach a weight value to the search results. The algorithm checks for the best cross-correlations and unifies the results by assigning the regular weight values to the results.

The implementation of the natural language model runs on a mobile phone and a PDA, and preliminary evaluation studies have shown it to be a good tool for these environments, better than traditional PC interfaces. Our current research aims to show that natural language (and speech) interfaces are appropriate interface styles for accessing audiovisual content, stored in home information servers, from mobile devices.

References

1. The site of the TV-Anytime Forum, http://www.tv-anytime.org
2. ETSI TS 102 822-3-1 V1.1.1 (2003-10) Technical Specification. *Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime Phase 1"); Part 3: Metadata;*
3. Kazasis, F.G., Moumoutzis, N., Pappas, N., Karanastasi, A., Christodoulakis, S. (2003). *Designing Ubiquitous Personalized TV-Anytime Services.* In the International Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS), Klagenfurt/Velden, Austria.
4. Johansson, P., Degerstedt, L., Jönsson, A. (2002). *Iterative Development of an Information-Providing Dialogue System.* In Proceedings of the 7th Workshop on User Interfaces for All. Chantilly, France.
5. Ibrahim, A., Johansson, P. (2002). *Multimodal Dialogue Systems: A Case Study for Interactive TV.* In Proceedings of the 7th Workshop on User Interfaces for All. Chantilly, France.
6. Ibrahim, A., Johansson, P. (2002). *Multimodal Dialogue Systems for Interactive TV Applications.* In Proceedings of 4th IEEE International Conference on Multimodal Interfaces, Pittsburgh, USA. pp. 117-222.
7. Jurafsky, D., Martin, J.H. (2000). *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* New Jersey: Prentice Hall.
8. Multimodal Interaction for Information Appliances (MIINA Project), `http://www.ida.liu.se/~nlplab/miina/`
9. NOKIA – Talkative TV-Guide, Program Guide Information System, `http://www.nokia.com/nokia/0,,27727,00.html`
10. Degerstedt, L. (2002). *JavaChart User Manual*, `http://nlpfarm.sourceforge.net/javachart/`
11. IST Project UP-TV: "biquitous Personalized Interactive Multimedia TV Systems and Services", `http://www.up-tv.de`
12. Maier E. (1997). *Clarification dialogues in VERBMOBIL.* In Proceedings of EuroSpeech-97, pp. 1891-1894, Rhodes, 1997
13. MySQL: The World's Most Popular Open Source Database, `http://www.mysql.com/`
14. Java Technology, `http://java.sun.com/`
15. The Java 2 Platform, Micro Edition (J2ME™), `http://java.sun.com/j2me/`

Updated Data Dissemination in Ad Hoc Networks

Hideki Hayashi, Takahiro Hara, and Shojiro Nishio

Dept. of Multimedia Eng., Grad. Sch. of Information Science and Tech., Osaka Univ.
2-1 Yamadaoka, Suita, Osaka 565-0871, Japan
{hideki, hara, nishio}@ist.osaka-u.ac.jp

Abstract. In this paper, we propose two updated data dissemination methods to update old replicas efficiently in ad hoc networks where each data item is updated at irregular intervals. The first method disseminates the updated data item to all connected mobile hosts when a mobile host holding an original data item updates the data item. In the other method, when two mobile hosts are connected, they disseminate updated data items. Our proposed methods reduce the number of accesses to old replicas and improve data accessibility.

1 Introduction

As one of the research fields in mobile computing environments, there has been increasing interest in *ad hoc networks* that are constructed of only mobile hosts that play the role of a router. Disconnections occur frequently in ad hoc networks, since mobile hosts move freely, and this causes frequent network division. If network division occurs due to the migration of mobile hosts, mobile hosts in one of the two divided networks cannot access data items held by mobile hosts in the other network. In Fig. 1, if the radio link between two mobile hosts is disconnected at the central part, the mobile hosts on the left-hand side and those on the right-hand side cannot access data items D_1 and D_2, respectively. A key solution to this problem is to replicate data items on mobile hosts that are not the owners of the original data item.

In ad hoc networks, there are many applications in which mobile hosts access data held by other mobile hosts; a good example is when a research project team constructs an ad hoc network and the team members refer to data obtained by other members in order to streamline work. Therefore, it is becoming increasingly important to improve data accessibility in ad hoc networks.

In [4, 6], we assumed that each mobile host has limited memory space for creating replicas in ad hoc networks where a data item is not updated, and proposed three replica allocation methods for improving data accessibility. These methods heuristically determine replica allocation based on the access frequency from each mobile host to each data item and the network topology at that moment. In [5], we extended the three methods proposed in [4] to adapt to an environment where each data item is periodically updated. These extended methods replicate data items on mobile hosts based on the access frequency, the time remaining until each item is next updated, and the network topology.

In a real environment, it is more likely that data items are updated at irregular intervals. In such a case, mobile hosts may access invalid replicas that have been updated. If a

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 28–42, 2004.

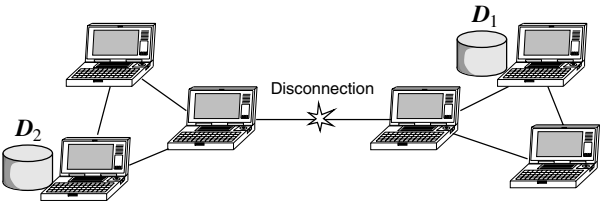


Fig. 1. Network division

mobile host accesses an invalid replica, a rollback occurs as needed when it connects with the mobile host holding the original. Such invalid accesses consume the power of mobile hosts and this is a serious problem for mobile hosts that usually have poor resources. In [7], we proposed two replica invalidation methods in ad hoc networks wherein each data item is updated at irregular intervals. These methods can reduce the number of accesses to old replicas that have been updated, although they cannot improve data accessibility. In this paper, we propose two updated data dissemination methods to update old replicas efficiently in order to reduce the number of accesses to old replicas and improving data accessibility.

The remainder of the paper is organized as follows. In Sect. 2, we show some conventional works related to our work. In Sect. 3, we describe our assumed environment, then in Sect. 4, propose two updated data dissemination methods. In Sect. 5, we show the results of simulation experiments, and finally in Sect. 6, we summarize this paper.

2 Related Works

Several strategies that invalidate and update old replicas in mobile computing environments have been proposed. In [2, 8], it is assumed that mobile hosts cache data items held by the database server in the fixed network and each data item is updated only by the server. The authors of [2, 8] proposed cache invalidation strategies in which the server periodically broadcasts invalidation reports to mobile hosts and invalidates old data items held by the mobile hosts efficiently. In [3], cache consistency management methods were classified into several categories and detailed simulation experiments were performed in both cases in which the server broadcasts invalidation reports and *update information* that includes new values of the updated data items. These approaches are similar to our approach because they invalidate and update cached data items held by mobile hosts. However, they assume that the database server in the fixed network broadcasts invalidation reports and disseminates updated data items. On the other hand, in our research, mobile hosts broadcast invalidation reports and disseminate updated data items considering the changes to network topology in ad hoc networks where each data item is updated at irregular intervals.

A few studies in the research field of ad hoc networks have been conducted to improve data accessibility [9, 10, 11]. In [9], the authors proposed methods by which replicas are allocated to a fixed number of mobile hosts that act as servers and keep the consistency among the replicas. There, consistency is maintained by employing a strategy based on

the quorum system that has been proposed for distributed database systems. In [10], the authors defined two new consistency levels among replicas and proposed methods that disseminate updated data to maintain consistency in an environment where data items are updated by different mobile hosts. In [11], the authors proposed a method for predicting the time when a network division occurs in ad hoc networks and creates replicas at each mobile host before the network division occurs. These are all considered similar to our methods because replicas are allocated to mobile hosts in an ad hoc network. However, these methods assume that replicas are allocated to mobile hosts holding unlimited memory space, whereas our methods effectively improve data accessibility in an environment where there are mobile hosts with limited memory space.

3 Assumptions and Approach

The system environment is assumed to be an ad hoc network where mobile hosts access data items held by other mobile hosts. In this paper, mobile hosts connected to each other by one-hop/multihop links are simply called *connected mobile hosts*. In addition, we make the following assumptions:

- We assign a unique *host identifier* to each mobile host in the system. The set of all mobile hosts in the system is denoted by $M = \{M_1, M_2, \cdots, M_m\}$, where m is the total number of mobile hosts and M_j $(1 \leq j \leq m)$ is a host identifier. Each mobile host moves freely.
- Data are handled as a single data item, which is a collection of data. We assign a unique *data identifier* to each data item located in the system. The set of all data items is denoted by $D = \{D_1, D_2, \cdots, D_n\}$, where n is the total number of data items and D_j $(1 \leq j \leq n)$ is a data identifier. All data items are of the same size, and the original of each data item is held by a particular mobile host.
- Each mobile host has memory space of C data items for creating replicas, excluding the space for the original data item held by the host. For this purpose, we do not place any restrictions on replica allocation methods used in the network, since our proposed updated data dissemination methods behave independently of the used replica allocation method.
- Each data item is updated by the mobile host holding the original at irregular intervals. After a data item is updated, the replicas become invalid; that is, each data item is not partially updated and the update information is not represented by the difference from the previous version.
- Each mobile host holds a table in which the information on the latest update times (time stamps) of all data items in the entire network is recorded. This information table is called a *time stamp table*. This table incorporates the data identifier and the time stamp as the attributes.
- We assume a simple transaction model in which a rollback occurring at a mobile host does not affect data processing in transactions issued by other mobile hosts. Therefore, cascade aborts do not occur.

In this system environment, a request for a data item is successful only when the request-issuing host accesses the original target data item or its replica with the same

time stamp (version) as the original. The request succeeds immediately if the request-issuing host holds the original target data item or connects to the mobile host holding the original. Otherwise, if the request-issuing host or at least one connected mobile host holds the replica of the target data item, the request-issuing host tentatively accesses the replica. Following that tentative access, when the request-issuing host connects to the mobile host holding the original target data item, the tentative access is determined as having either succeeded or failed. If the tentative access fails, the rollback occurs as needed so that the request-issuing mobile host to the state it was in before the replica was accessed. If the request issuing host and the connected mobile hosts do not hold the original/replicas of the target data item, the request fails immediately.

4 Updated Data Dissemination Methods

In this section, we propose two updated data dissemination methods to reduce the number of accesses to old replicas and improve data accessibility. In the following, we describe the details of the two methods. Table 1 shows message packets used in the proposed methods. In Table 1, "host ID" denotes the host identifier of the mobile host sending the message, and "TS" denotes the time stamp of the data item specified by the data identifier ("data ID").

Table 1. Messages for updated data dissemination methods

Packet name	Elements
Invalidation report	data ID, TS
Updated data request	host ID, list of data IDs
Updated data reply	host ID, list of data IDs

4.1 DU (Dissemination on Update) Method

In the DU method, when a mobile host holding an original data item updates the item, it floods all connected mobile hosts with an invalidation report, just like the methods proposed in [7]. When a mobile host receives the invalidation report, it refers to its own time stamp table and checks whether the replica held by the host is invalid. More specifically, the mobile host compares the time stamp in the received invalidation report with that of the corresponding data item in its own time stamp table. If the former is larger, the host updates the time stamp in its own time stamp table to that in the received invalidation report. At the same time, the mobile host broadcasts the received invalidation report to its neighboring mobile hosts. Furthermore, if the mobile host holds a replica of the corresponding data item, it discards the replica from its own cache and requests the updated data item to the mobile host holding the original. When the mobile host holding the original receives this request, it transmits the updated data item to the request issuing host. If a radio link on the route for transmitting the updated data item temporarily disconnects during the transmission, e.g., due to electromagnetic

interference, the mobile host attempts to re-transmit the updated data item via another route founded by the used routing protocol. If no route is available, the mobile host does not re-transmit the updated data item. This operation for temporal disconnection is also performed in the DC method described in the next subsection.

If the time stamp in the received invalidation report is same as that in the time stamp table, i.e., the mobile host receives the same invalidation report once again, it does not broadcast the received invalidation report to its neighboring mobile hosts and discards the report.

Figure 2 shows a behavior of mobile host M_1, which updates its original data item D_1 and transmits the updated data item to mobile host M_3. In this figure, a rectangle with heavy line denotes an original, other rectangles denote replicas and an arrowhead denotes the flow of the updated data item.

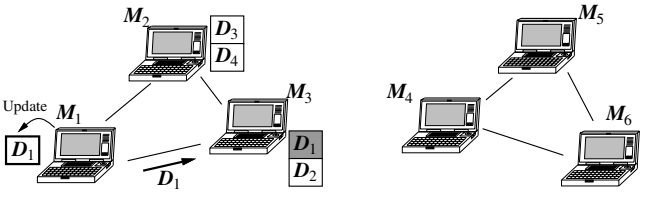


Fig. 2. Updated data dissemination in DU method

In this method, the traffic is light since the flooding with an invalidation report and the dissemination of an updated data item are performed only when a mobile host updates the original data item. Mobile hosts connecting to the mobile host holding the original data item can allocate the replicas with the same time stamp as the original because they can receive the updated data item each time update occurs. In contrast, mobile hosts that do not connect to the mobile host holding the original data item cannot receive the invalidation report nor the updated data item. Therefore, in an environment where link connections and disconnections frequently occur due to the movement of mobile hosts, connected mobile hosts may hold different time stamps and replicas with different versions of the same data item.

4.2 DC (Dissemination on Connection) Method

In the DC method, similar to the DU method, a mobile host holding an original data item floods other hosts with the invalidation report and disseminates the updated data item every time it updates the data item. In addition, every time two mobile hosts are newly connected with each other, they update their time stamp table, flood invalidation reports, and disseminate updated data items. Similar to the method proposed in [7], when two mobile hosts are newly connected with each other, the flooding of invalidation reports is performed as follows:

1. When two mobile hosts M_i and M_j ($i < j$) are newly connected with each other, the mobile host with the larger suffix (j) of the host identifier (M_j) transmits its own time stamp table to the other one.

2. Mobile host M_i compares the entry for each data item in its own time stamp table with that in the time stamp table received from mobile host M_j and updates its own time stamp table. The following processes are then executed:
   - M_i floods invalidation reports for data items whose time stamps held by M_i are smaller than that held by M_j to mobile hosts originally connected to M_i. Here, mobile hosts originally connected to M_i denote mobile hosts that have been connected to M_i by one-hop/multihop links before M_i and M_j are connected, and are also connected after that. For example, in Fig. 3, mobile hosts originally connected to M_3 are M_1 and M_2.
   - M_i sends information on the updated time stamps for data items whose time stamps held by M_j are smaller than those held by M_i to M_j. Then, M_j floods the mobile hosts originally connected to M_j with the invalidation reports for these items.

   Mobile hosts receiving invalidation reports update their time stamp tables and discard old replicas in similar fashion to the DU method.

In the DC method, connected mobile hosts hold the same time stamp table because invalidation reports are rebroadcast whenever two mobile hosts are newly connected.

After the mobile host floods others with invalidation reports, it disseminates updated data items. The traffic produced by the DC method is larger than that in the DU method since each mobile host floods others with invalidation reports and disseminates updated data items more frequently. In particular, the disseminations of updated data items drastically increase the network traffic. Therefore, we propose two variations of the DC method that differ from each other in terms of the range in which they disseminate updated data items.

DC/OO (DC/One-to-One) Method. In the DC/OO method, two mobile hosts newly connected disseminate updated data items with each other after they flood each other with invalidation reports. The procedure that M_j (M_i) disseminates updated data items to M_i (M_j) is as follows:

1. M_i (M_j) sends the *updated data request packet* that includes the data identifiers of replicas invalidated by the invalidation reports to M_j (M_i).
2. M_j (M_i), which received the updated data request packet, transmits the updated data items whose identifiers are included in the packet to M_i (M_j).

Figure 3 shows the DC/OO method where M_3 and M_4 are newly connected. This focuses on the behavior of M_3, which disseminates the updated data item (D_3) to M_4, where M_3 holds the data item requested by M_4.

In this method, data accessibility is expected to be superior to that in the DU method since two newly connected mobile hosts disseminate updated data items with each other and refresh old replicas in their cache space. Here, it should be noted that mobile hosts that are connected originally to the two hosts cannot refresh the old replicas.

DC/GG (DC/Group-to-Group) Method. In the DC/GG method, two groups of mobile hosts that were originally connected to the two newly connected mobile hosts disseminate updated data items after they transmit floods of invalidation reports. The procedure by

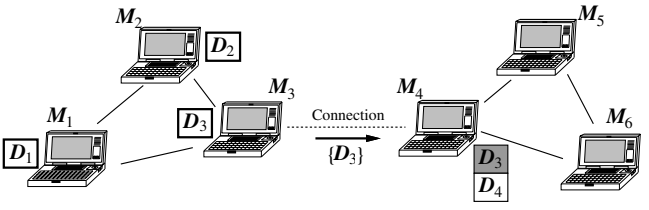


Fig. 3. Updated data dissemination in DC/OO method

which each mobile host originally connected to M_j (M_i) disseminates updated data items to each mobile host originally connected to M_i (M_j) is as follows:

1. Each mobile host originally connected to M_i (M_j) sends the updated data request packet, which includes the data identifiers of replicas invalidated by the invalidation reports, to M_i (M_j).
2. M_i (M_j) merges the received updated data request packets into its own updated data request packet and sends the packet to M_j (M_i).
3. M_j (M_i) floods the mobile hosts to which it was originally connected with the received updated data request packet.
4. If each mobile host originally connected to M_j (M_i) holds data items or replicas whose data identifiers are included in the updated data request packet, it sends an updated data reply packet that includes the data identifiers to M_j (M_i).
5. M_j (M_i) recognizes mobile hosts requesting the updated data items from the received updated data reply packets. If two or more mobile hosts hold a requested updated data item (replica), M_j (M_i) requests the item from one of them, where the mobile host connects to M_j (M_i) by the link with the smallest number of hops. After receiving the updated data items, M_j (M_i) transmits them to M_i (M_j).
6. M_i (M_j) disseminates the updated data items to the mobile hosts to which it was originally connected, which have sent the updated data request packets to M_i (M_j) in step 1.

Figure 4 shows an example of the updated data dissemination in the DC/OO method where M_3 and M_4 are newly connected. This example focuses on the behavior of mobile hosts on the left-hand side $\{M_1, M_2, M_3\}$ that disseminate the updated data item $\{D_1, D_2, D_3\}$ to those on the right-hand side $\{M_4, M_5, M_6\}$.

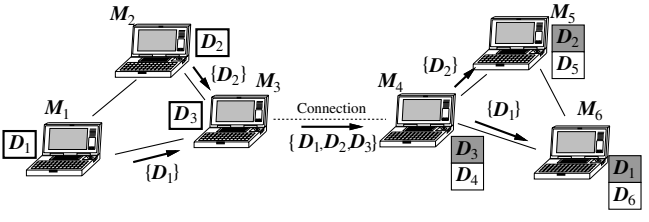


Fig. 4. Updated data dissemination in DC/GG method

In this method, the data accessibility is expected to be higher than that in the DC/OO method since mobile hosts connected to the two newly connected mobile hosts refresh old replicas in their cache space. On the other hand, the traffic is expected to be higher than that in the DC/OO method since the updated data items are widely disseminated.

5 Simulation Experiments

In this section, we present simulation results to evaluate the performance of the proposed methods.

5.1 Simulation Model

The number of mobile hosts in the entire network is 40 ($M = M_1, \cdots, M_{40}$), and they exist in a size L [m] $\times$ L [m] flatland. Each host randomly determines a destination in the flatland and moves toward the destination at a velocity randomly determined from 0 to v [m/s] (v: maximum movement speed). When the host arrives at the destination, it determines a next destination and moves toward that destination without pausing. The radio communication range of each mobile host is a circle with the radius R [m].

There are 40 types of data items in the entire network, and M_i holds D_i ($i = 1, \cdots, 40$) as the original. The access frequency of mobile host M_i to D_j is $p_{ij} = 0.005 \times (1 + 0.0001j)$. We assume the unit of time as 10 [s]. We define the size of each packet used in our proposed methods as the number of attributes in the packet. The size of each data item D_j is 250,000. Table 2 shows the size of each packet and a data item. Updates of data item D_i occur at intervals based on an exponential distribution with mean U [s].

Each mobile host creates up to 10 replicas in its memory space with the DCG (Dynamic Connectivity based Grouping) method that we have proposed in [4]. The DCG method shows the highest data accessibility among the three methods proposed in [4], ant it creates stable groups of mobile hosts at every relocation period T and shares replicas in the groups. More specifically, this method creates groups of mobile hosts as *biconnected components* [1] in the network and subsequently allocates replicas of data items on mobile hosts in each group in descending order of access frequencies within the group. By grouping mobile hosts as a biconnected component, the group is not divided even if any one of the mobile hosts in the group disappears from the network.

Table 2. Packet size

Packet name	size
Invalidation report	2
Updated data request	1+(number of elements in the list of data IDs)
Updated data reply	1+(number of elements in the list of data IDs)
Data query	2
Data query reply (DU method)	4
Data query reply (DC method)	3
Data item	250,000

Table 3. Parameter configuration

Parameter	value		
L^2	250,000	(10,000∼1,000,000)	[m^2]
R	60		[m]
U	1,000	(100∼3,000)	[s]
v	1		[m/s]
T	1,000		[s]

Table 3 shows parameters and their values used in the simulation experiments. The parameters are basically fixed to constant values, but some parameters are changed, indicated by values in parentheses shown in Table 3. In this simulation environment, mobile hosts move at speeds ranging from walking to running. Each mobile host applies a wireless radio communication protocol that offers a broad communication range such as IEEE 802.11b, and issues an access request for a data item almost every minute. We assume the size of an attribute in each packet to be 4 [B]. As a result, the size of a data item becomes 1 [MB] and the size of memory space for creating replicas becomes 10 [MB].

In the simulation experiments, we take into account the transfer delay of each data item assuming that the radio communication bandwidth is 11 [Mbps] and the hop delay is 0.1 [s]. If a radio link on the route for transmitting an updated data item or a message disconnects during the transmission due to the movement of mobile hosts on the route, we assume that the destination mobile host cannot receive the item or message.

In the simulation experiments, we randomly determine the initial position of each mobile host and evaluate the following four criteria of each of our proposed methods during 1,000,000 units of time.

- *Data accessibility*:
  The ratio of the number of successful access requests to the number of all access requests issued during the simulation period.
- *Rate of accessing invalid replicas*:
  The rate of the number of tentative data accesses that resulted in failure to the number of all access requests issued during the simulation period.
- *Traffic for disseminating updated data*:
  The total traffic caused by disseminating updated data items and transmitting control packets (invalidation reports, updated data request packets, and updated data reply packets) during the simulation period. Here, the traffic of disseminating updated data items and transmitting control packets is defined as the product of the total hop count for disseminating and transmitting them and their sizes.
- *Rate of traffic for disseminating updated data*:
  The rate of traffic for disseminating updated data to the total traffic in the system. We define the total traffic in the system as the sum of that for allocating replicas to mobile hosts, that for accessing data items, and that for disseminating updated data items.

5.2 Effects of Average Update Period

First, we examine the effects of the average update period U on each of the proposed methods. Figures 5, 6, 7, and 8 show the simulation results. In these graphs, the horizontal axis indicates the average update period U. The vertical axes indicate the data accessibility, the rate of accessing invalid replicas, the traffic for disseminating updated data, and the rate of traffic for disseminating updated data, respectively. In all graphs, for the purpose of comparison, the performances when the flooding of invalidation reports and the dissemination of updated data items are not performed are also shown as "NO."

Figure 5 shows that the data accessibility in the DC/OO and DC/GG methods is higher than that in the DU method. This is because in the DC/OO and DC/GG methods, each mobile host holds more new replicas, since updated data items are disseminated every time two mobile hosts are newly connected to each other. The DC/GG method provides the highest data accessibility, since updated data items are disseminated to all mobile hosts that were originally connected to the two newly connected mobile hosts. As

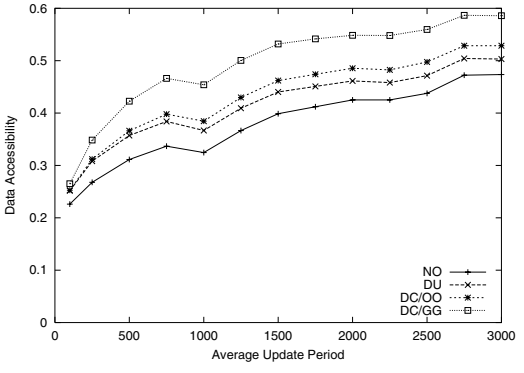


Fig. 5. Average update period U and data accessibility

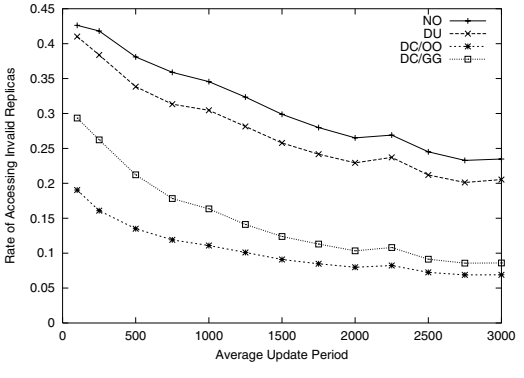


Fig. 6. Average update period U and rate of accessing invalid replicas

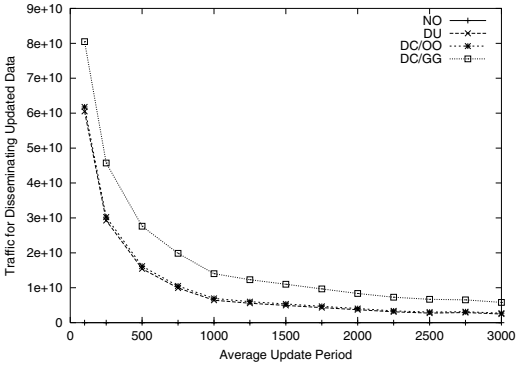


Fig. 7. Average update period U and traffic for disseminating updated data

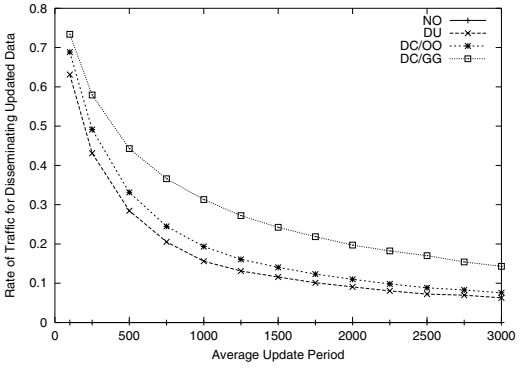


Fig. 8. Average update period U and rate of traffic for disseminating updated data

the average update period increases, in all of the proposed methods, the data accessibility improves because the replicas held by each mobile host are valid for a longer time.

Figure 6 shows that the rates of accessing invalid replicas in the DC/OO and DC/GG methods are lower than that in the DU method. This is because the DC/OO and DC/GG methods can invalidate many old replicas because invalidation reports are flooded across the entire network every time two mobile hosts are newly connected. The DC/OO method gives lower rate of accessing invalid replicas than the DC/GG method. This is because the DC/GG method not only allocates more valid replicas to mobile hosts, but also allocates more invalid replicas than the DC/OO method.

Figure 7 shows that the DC/GG method produces the highest traffic for disseminating updated data and the DC/OO method produces the next. This result is obvious because the DC method disseminates updated data items more frequently than the DU method, and the DC/GG method disseminates them to wider ranges than the DC/OO method. As the average update period increases, the traffic for disseminating updated data decreases in all of the proposed methods because the frequency of disseminating updated data items and transmitting control packets falls as the frequency at which each data item

is updated drops. In addition, since the differences among time stamps held by mobile hosts become smaller, the number of updated data items disseminated by mobile hosts reduces when two mobile hosts are newly connected.

Figure 8 shows that the DC/GG method gives the highest rate of traffic for disseminating updated data and the DC/OO gives the next. This is due to the reason similar to that given for Fig. 7: As the average update period decreases, the traffic for disseminating updated data falls in comparison with the total traffic in all of the proposed methods.

5.3 Effects of Area Size

In this subsection, we examine the effects of the simulation area L^2 on each of the proposed methods. Figures 9, 10, 11, and 12 show the simulation results. In these graphs, the horizontal axis indicates the simulation area L^2. The vertical axes indicate the data accessibility, the rate of accessing invalid replicas, the traffic for disseminating updated data, and the rate of traffic for disseminating updated data, respectively.

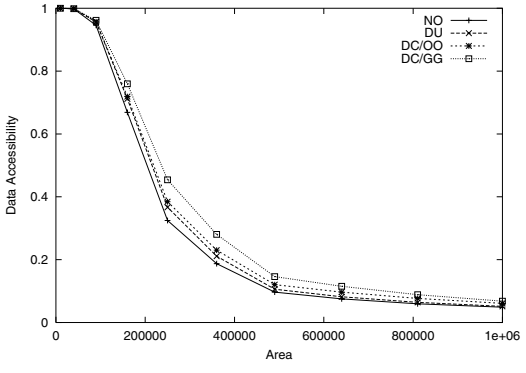


Fig. 9. Area L^2 and data accessibility

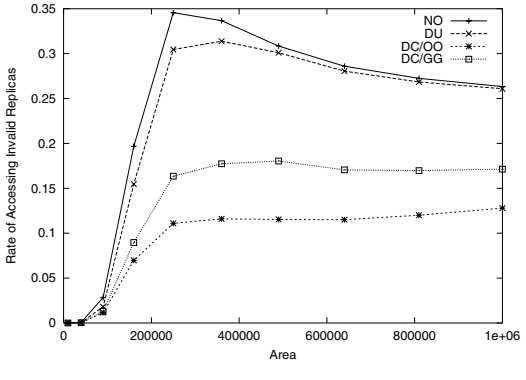


Fig. 10. Area L^2 and rate of accessing invalid replicas

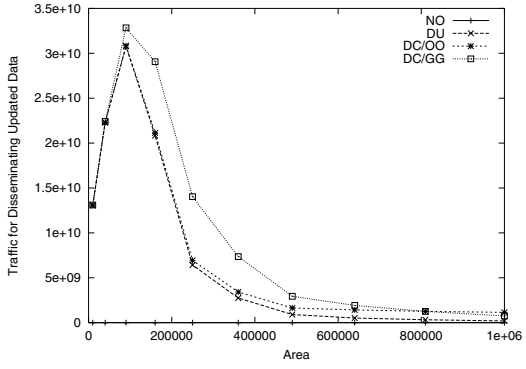


Fig. 11. Area L^2 and traffic for disseminating updated data

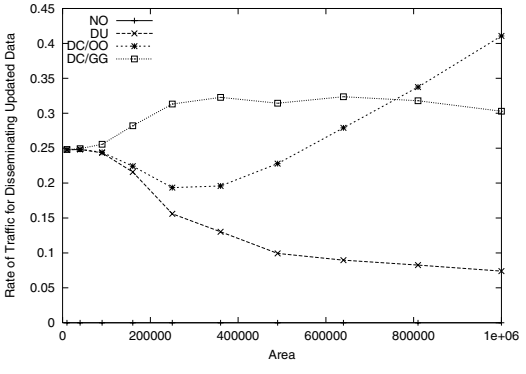


Fig. 12. Area L^2 and rate of traffic for disseminating updated data

Figure 9 shows that the DC/GG method gives the highest data accessibility, then the DC/OO method, for a similar reason to that given for Fig. 5. As the area drastically shrinks, the data accessibility gets closer to 1 in all of the proposed methods. This is because each mobile host nearly always connects to mobile hosts holding the original data items. On the other hand, as the area increases, the data accessibility decreases in all of the proposed methods because the number of connected mobile hosts decreases, thus the number of replica types that the mobile hosts can access falls.

Figure 10 shows that the DC/OO method gives the lowest rate of accessing invalid replicas, followed by the DC/GG method, for a similar reason to that given for Fig. 6. In the DU method, as the area expands, the rate of accessing invalid replicas increases drastically, although it begins to decrease from a certain point. This is because the probability that each mobile host connects to mobile hosts holding the original data items falls, thus fewer hosts receive invalidation reports and updated data items. However, as the area becomes extremely large, each mobile host rarely accesses replicas of the data items and thus the rate of accessing invalid replicas decreases. In the DC/OO and DC/GG methods, as the area grows drastically, the rate of accessing invalid replicas remains fairy

steady. This is because frequent disseminations of updated data items in the DC/OO and DC/GG methods increase the chance of accessing replicas.

Figure 11 shows that the DC/GG method produces the highest traffic for disseminating updated data, followed by the DC/OO method. In all of the proposed methods, as the area expands, the traffic for disseminating updated data first increases, but then falls from a certain point. When the area is very small, nearly all mobile hosts connect with each other by one-hop links and thus the traffic for disseminating updated data is low. When the area is very large, the number of connected mobile hosts is low and thus the traffic is low.

Figure 12 shows that the DC method gives a higher rate of traffic for disseminating updated data than the DU method. This reason is similar to that given for Fig. 11. On comparing the DC/OO and DC/GG methods, the DC/GG method gives a higher rate in most cases. However, when the area is very large, the DC/OO method gives a higher rate because the traffic for data access in the DC/OO method is much lower than that in the DC/GG method since it is more difficult to disseminate updated data items. As a result, in the DC/OO method, the rate of traffic for disseminating updated data increases.

6 Conclusions

In this paper, we proposed two updated data dissemination methods in ad hoc networks to reduce the number of accesses to old replicas and to improve data accessibility. In the DU method, when a mobile host holding an original data item updates the data item, it disseminates the updated data item to its connected mobile hosts. In the DC method, when two mobile hosts are newly connected, they disseminate updated data items. The DC method features two versions: the DC/OO and DC/GG methods. In the DC/OO method, two newly connected mobile hosts send updated data items between each other, while in the DC/GG method, two groups of mobile hosts that were originally connected to the two newly connected mobile hosts disseminate updated data items. The results of simulation experiments showed that the DC method reduces the rate of accessing invalid replicas, but produces higher traffic than the DU method. The results also showed that the DC/GG method gives superior data accessibility, but it also has a higher rate of accessing invalid replicas and traffic than the DC/OO method. In a real environment, the most appropriate method should be chosen among our proposed methods according to the update frequencies of data items and system characteristics.

We plan to consider a method in which mobile hosts that updated their originals disseminate not the entire data items but their differences from their previous versions in order to reduce the traffic.

Acknowledgments. This research was partially supported by The 21st Century Center of Excellence Program “New Information Technologies for Building a Networked Symbiotic Environment” and Grant-in-Aid for Young Scientists (A)(1668005) of the Ministry of Education, Culture, Sports, Science and Technology, Japan, and by Tateishi Science and Technology Foundation.

References

1. A.V. Aho, J.E. Hopcroft, and J.D. Ullman, "The design and analysis of computer algorithms," *Addison-Wesley*, 1974.
2. D. Barbara, and T. Imielinski, "Sleepers and workholics: caching strategies in mobile environments," *Proc. ACM SIGMOD'94*, pp. 1–12, 1994.
3. J. Cay, K.L. Tan, and B.C. Ooi, "On incremental cache coherency schemes in mobile computing environments," *Proc. IEEE ICDE'97*, pp. 114–123, 1997.
4. T. Hara, "Effective replica allocation in ad hoc networks for improving data accessibility," *Proc. IEEE Infocom'01*, pp. 1568–1576, 2001.
5. T. Hara, "Replica allocation methods in ad hoc networks with data update," *ACM-Kluwer Journal on Mobile Networks and Applications (MONET)*, vol. 8, no. 4, pp. 343–354, 2003.
6. T. Hara, N. Murakami, and S. Nishio, "Replica Allocation for Correlated Data Items in Ad-Hoc Sensor Networks," *ACM SIGMOD Record*, Vol. 33, no. 1, pp. 38–43, 2004.
7. H. Hayashi, T. Hara, and S. Nishio, "Cache invalidation for updated data in ad hoc networks," *Proc. Int'l Conf. on Cooperative Information Systems (CoopIS'03)*, pp. 516-535, 2003.
8. J. Jing, A. Elmagarmid, A. Helal, and R. Alonso, "Bit-sequences: an adaptive cache invalidation method in mobile client/server environments," *ACM-Kluwer Journal on Mobile Networks and Applications (MONET)*, vol. 2, no. 2, pp. 115–127, 1997.
9. G. Karumanchi, S. Muralidharan, and R. Prakash, "Information dissemination in partitionable mobile ad hoc networks," *Proc. Symposium on Reliable Distributed Systems (SRDS'99)*, pp. 4–13, 1999.
10. K. Rothermel, C. Becker, and J. Hahner, "Consistent update diffusion in mobile ad hoc networks," *Technical Report 2002/04, Computer Science Department, University of Stuttgart*, 2002.
11. K. Wang, and B. Li, "Efficient and guaranteed service coverage in partitionable mobile ad-hoc networks," *Proc. IEEE Infocom'02*, vol. 2, pp. 1089–1098, 2002.

Modelling Context for Information Environments

Rudi Belotti, Corsin Decurtins, Michael Grossniklaus,
Moira C. Norrie, and Alexios Palinginis

Institute for Information Systems,
ETH Zurich,
8092 Zurich, Switzerland
rudi.belotti@switzerland.org
{decurtins, grossniklaus, norrie, palingins}@inf.ethz.ch

Abstract. Context-awareness has become an important consideration in the development of mobile and ubiquitous systems. While efforts have been made to develop general context models and application frameworks, it remains the case that often the notion of context supported is very restrictive and/or the representation of context is based on simple key-value pairs. As a result, most existing systems lack well-defined semantics and typing for context that would facilitate the general implementation, maintenance and adaption of context-aware systems. In this paper, we present a general context model that consolidates the models underlying many other approaches, while moving to a higher-level of abstraction in terms of semantic context description.

1 Introduction

As humans, every time we act and speak, we do it in a given context. Adapting to context in our daily lives is very easy for us and most of the time it is a completely subconscious action. We implicitly change our behaviour depending on the situation and in relation to the environment. Context is the driving force behind our ability to discriminate what is important and what is not. It allows us to enrich our knowledge about a certain situation by drawing on related memories and experiences. Hence, only through context are we able to manage the tremendous amount of information that surrounds us at any given point in time. For instance, the implicit use of context during a dialogue between humans has been shown to increase the conversational bandwidth [1].

The nature of the information that surrounds us is not unlike the data that can be found in ubiquitous and mobile applications. As in real life, vast amounts of information are readily available from an almost unlimited number of sources. To make such information environments work, the need to distinguish relevant from irrelevant information arises, as well as the necessity to augment the information by associating related knowledge. If such applications are ever to be successful, it will entirely depend on their ability to deliver the right information to the right user at the right time. We believe that context is a powerful instrument to achieve the dissemination of relevant information that those applications demand.

Many of us dream of better applications that have at least an inkling of what we are doing or where we are working and adapt themselves accordingly. This adaptation

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 43–56, 2004.

can come in various forms. For instance, a very simple adaptation would be having the user interface adapt according to the situation by presenting the toolbars which offer the actions that are needed to perform the next task we have in mind. A more complex example would comprise the delivery of useful and relevant information via mobile devices such as mobile phones or personal digital assistants (PDAs) to support expedient collaborative work. A PDA, for instance, could be used to display background information on the knowledge and know-how of a co-worker whenever this person enters the room. Mobile phones on the other hand could simply adapt their ringing profile automatically depending on the situation. Context in this example would also allow calls that are irrelevant to the current setting, such as an important meeting, to be filtered out.

Recently, a lot of research has been dedicated to context and context-aware applications. Nowadays, the first results on context-aware computing are already appearing in various application domains. Work on context has mainly been concentrated in the areas of philosophical implications of context [11], low-level implementation of context-aware systems [30, 4] and the development of more general frameworks [27, 28] that provide a basis for the engineering of such applications. We feel, however, that all of the models used in these approaches lack precise and expressive semantics for context as well as typing. In this paper, we present our comprehensive model for context that aims at consolidating existing approaches. As it is based on a well-defined object model and therefore features well-defined semantics and typing, it propels the reusability and interchangeability of context in a given application.

In Sect. 2 existing work on context and context-aware applications is presented and related to our work. Section 3 introduces our own model for context and gives an overview of the various parts of the model. Sections 4 to 6 then examine certain aspects of the model in greater detail. Final remarks and conclusions are given in Sect. 7.

2 Related Work

Context has become a well researched topic over the past few years with contributions from a wide variety of domains, including philosophy [11], linguistics and social sciences as well as many different fields of computer science. In the following we present related work from some of these areas and state how these results relate to our own approach. The aim is not to give a complete overview of all related work, but rather to provide an overview of the various approaches that people have adopted to tackle the problem of context. A more complete survey on related work, especially in the field of mobile computing, is available in [9].

The problem of context has a long tradition in different areas of cognitive science. Several formalisations [17, 10] of context have been proposed and, in the field of Computational Linguistics [18], context is a central notion for understanding and working with text or speech. Cognitive system engineers and specifically people in the area of Human Computer Interactions (HCI) have recognised that cognition does not exist without a context and have started applying the theory to practical applications [4].

The first implementations of context-aware systems originate in the area of application development for ubiquitous and pervasive computing. The solutions tend to be application-driven, i.e. context is not necessarily regarded as the main research topic, but rather just used to implement a certain functionality for a given application. In [2], for instance, although a context layer is described in the architecture, its design is heavily affected by the fact that the system is built with the focus on multi-channel acquisition and delivery and therefore not considered as a general purpose context model. In other approaches context providers — physical sensors and software components — are added to the applications to allow them to sense their environment and gather information about users, location and possibly other properties of the physical environment. Context is an integral part of these systems. The integration of sensors and the reactions and adaptations of the system to the context are customised and tailor-made for each application. Most of the systems do not have an explicit model of context. Context is just additional data that the application can use to provide the desired functionality. An interesting example of such an application is described in [30] where active badges are used to sense the location of people and forward calls for a given person to a telephone in the same room.

The notion of context has also been integrated into various application frameworks, e.g. for web engineering. The output generated by a web application usually depends on various explicit parameters. These parameters include, for instance, the exact URL that was requested or the input of a form. In addition to these explicit parameters, the output can also depend on *implicit* parameters, i.e. parameters which are not specified explicitly by the user, but rather form the context of the current session. These implicit parameters might include the preferred language of the user, the browser that is used to access the web application (e.g. an HTML browser or a WAP phone) or the current time. Also, the history of the current session, i.e. the documents that the user has accessed before the current request, can be part of this context. A discussion on implicit and explicit application parameters can be found in [15].

The web modelling language WebML and the corresponding CASE tool WebRatio have also incorporated a notion of context [3]. This context is used to provide users with a personalised and customised view of the web information system. Pages of the application can be defined to be context-aware, which means that the context elements are available to the page in the form of data units. The page can then be customised according to these context data units.

We have developed our own context-aware web publication frameworks OMS Web Elements (OMSwe) [23, 22] and the eXtensible Content Management (XCM) [7, 8] as an extension of an object-oriented database system. The database contains, not only objects of the application domain, but also metadata that controls the assembly of documents along with templates that are used to render the pages into a specific output format. For each of these objects, multiple variations can coexist. Each variation is annotated with special attributes known as *characteristics*. For example, for an object of the application domain, this might be the language of the content, whereas, for a template object, it might be the type of the output format. In the process of responding to a specific user request, the correct variants of the objects are selected according to the context of the request.

In the realm of ubiquitous and pervasive computing, many researchers have worked on application-specific context models [29]. The focus of such applications is the physical context instances such as location, temperature etc. and the relationship between them. The application specification handles the context acquisition and manages both the context information and the interoperability with the application in a hard-wired manner.

While the approaches that we have described so far are quite feasible for standalone context-aware applications, they are certainly not practical for the integrated suites of context-aware applications that have recently been developed in, for example, the AwareHome [13] and Gaia [26] projects. To achieve a better separation of the sensors for the context aggregation and the context-aware applications, various people started to develop *context frameworks*. These frameworks allow the integration of various sensors through corresponding drivers and present a sensor-independent view of the context information to the applications. Instead of having sensors and context as an integral part of the application, the systems use a context component that provides context information for multiple applications. In some frameworks, the applications are also able to modify and insert additional context information and thus are able to act as context deliverers for other applications. A well-known representative of this approach is the Context Toolkit [27], which was used for a variety of research projects, including the previously mentioned AwareHome. Another example is the context component of the Gaia project.

Proposed frameworks define further system architectures [5, 28, 31] that offer general mechanisms and services to the underlining applications. In [12], this kind of service infrastructure is even shifted towards network-accessible middleware infrastructures.

Whereas context frameworks provide a good solution for the problems of aggregation and decoupled provision of context information, they do not really solve the problem of the lack of semantics. In most frameworks, context values are nothing but key-value pairs or higher-order tuples, with little explicit semantics. Some systems provide simple taxonomies for the values, but these conventions are neither checked nor enforced by the system. The interpretation of the context information is left completely to the application. Thus a real separation of context producers and context consumers is not achieved by context frameworks alone. With our background in information systems, we believe that the lack of semantics and expressiveness of context in the previously described approaches is actually the biggest drawback and makes them inadequate for use in a general context model and framework for information environments.

Various people have come up with definitions for context. A precise definition of context is very important for users of context frameworks and developers of context-aware applications in general. Schilit [28], for example, identified the *computing environment* (processors, devices, displays etc.), the *user environment* (location, social situation, nearby people etc.) and the *physical environment* (e.g. lighting and noise level) as the key components of context. Dey and Abowd [1] describe context as:

> [...] any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

These definitions have also been incorporated into some context frameworks. The Context Toolkit, for instance, is based on the definition of Dey and Abowd. However, the integration is very weak and the system provides little means of checking and enforcing the definitions. In most implementations, the efforts of defining context are only used for naming conventions. For this reason, other researchers have formalised definitions of context in the form of ontologies. With corresponding changes to the context frameworks, these definitions can be exploited by the systems and thus more powerful semantics can be built right into the frameworks and applications.

The previously mentioned definitions of context are mostly influenced by the ubiquitous and pervasive computing community. Another community which is very interested in context is the HCI community. Researchers from this field have brought aspects of ergonomics, psychology, sociology and even philosophy into the discussion. Dourish [6], for example, says that context is not information per se, but rather a property of existing information. Every data object can be "contextually relevant" to another object for a particular activity of a user. In other words, there is no separation between context objects and normal application objects.

The notion of a subject for context values, i.e. an entity that the context value is attributed to, has been implemented for example by the context framework of the Gaia project. They use quadruples for the representation of context: *Context(ContextType, Subject, Relater, Object)*. The context *Object*s have a *ContextType* and a *Subject* that they attribute. The nature of this attribution is described by the *Relater* property.

As for another requirement of Dourish, namely the integration of context and data model, relatively little research has been done in this area. Of course, it is always possible to use object identifiers or primary keys for context keys, values or subjects to work around this problem. However, we believe that a real integration of context models and data models is necessary to bring context-aware computing to the next level. The information modelling community certainly has great expertise in such approaches and this is actually where we see our main contribution. In the following sections, we will describe our approach for the modelling of context and explain some of the special features in more detail.

3 A Conceptual Data Model for Context

The purpose of this work is neither to develop just another context-aware application nor to create a framework and an infrastructure for building context-aware applications. Instead, by consolidating requirements and experiences from earlier context-aware applications and frameworks, including our own, we seek to develop a conceptual data model around the notion of context.

Based on the approaches presented in the previous section, we agree that frameworks, toolkits and service infrastructure in general, increase code reusability in a system. However, due to their complexity and the service-oriented description, many existing frameworks are difficult to comprehend due to the lack of a clearly defined conceptual information model. The specification of a framework reflects the infrastructure underneath which, inevitably, is influenced by the system on which it is implemented. Service-oriented specification is not sufficient in heterogeneous environments, where in-

formation exchange becomes a crucial issue due to the lack of a comprehensive general information model. The solution to such problems is the clear definition of a specific context model and infrastructure metamodel. Although this could limit the flexibility of seeing everything as an abstract service, it enables us to better comprehend and control such environments.

In the subsequent sections, we describe an abstract conceptual model to represent context. Based on this model, frameworks and infrastructures can be built by ensuring comprehension of the context domain and interoperability between the different solutions through the common model.

The semantics used to develop our model are those of the Object Model (OM) [20], an object-oriented data model influenced by ER and conceptual modelling. It is based on the simple and fundamental notions of objects participating in collections that represent real world concepts. Objects can be associated to each other using bidirectional associations. Concepts are represented by shadowed rectangles and associations by shadowed ovals, respectively. The organisation of collections and associations into classification structures together with basic constraints such as membership containment and cardinality constraints over associations increase the expressiveness of the model. Each data model is presented in a two-layer representation, one for the abstract high-level concepts, their roles and relationships, and another for the object type definitions and structures used by an underlying implementation. The simple high-level conceptual abstractions used make the model appropriate, not only for application modelling, but also as a metamodelling tool. In the past, we have used such metamodelling approaches to describe the OM model itself [19] and for the design of a modular database environment [21].

Additionally, OMS Pro, an object-oriented database management system [24] based on the model, enables to test the approach in a rapid prototyping environment. The rich functionality offered by the system ranges from typical database services such as persistence, user transactions, declarative query and definition languages, to system extensions such as XML support, web and GUI interfaces. Such a platform allows us to seamlessly integrate the modelling efforts with a preliminary proof-of-concept infrastructure and prototype application.

By consolidating the field of context-aware applications and frameworks, we highlight three basic concepts illustrated by the modules model of Fig. 1:

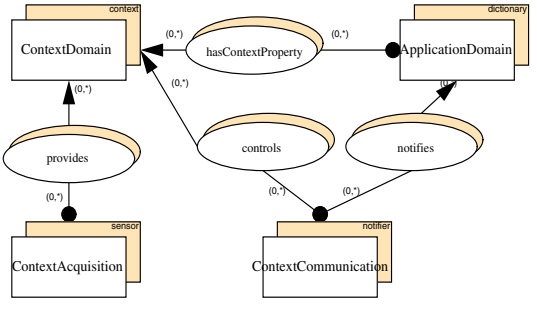


Fig. 1. Modules of the Metamodel

- The abstract notion of *context* as information that can be used to characterise the situation of an application *entity* [1]. Depending on the level of detail when considering the situation, context information can be anything from a single temperature measurement to an arbitrary complex structure, including entities of the application domain itself, representing multiple aspects of the entity's environment.
- Context information is gathered by *providers*. They supply a wide range of context information, from real world environment properties such as temperature sensing to complicated social behaviour aspects of the entity in the current situation. This is the reason for using a general notion of providers rather than the one of sensors typically used in supplying characteristics of a physical environment.
- *Notifiers* bridge the world between the environment properties and the entities interested in them. Crucial to the applicability of a context-aware application is the interaction between the context model and the application model. The mechanism allows both the automatic notification on context change or the explicit context acquisition upon request.

Instances of these modules and the interaction between them should be a subject of well defined definitions. Thus all model components have to be based on a *type* system.

In the following sections, core domain concepts of our model, such as the acquisition of context, types and operations on context, event triggering and the binding of context to an application are presented in more detail.

4 Context, Types and Application Entities

The core concept of the model is the *context* element, which represents the context abstraction. In most of the related discussions and definitions, context information characterises an *application entity* which is illustrated in the model of Fig. 2 by the association `hasContextProperty` in the lower right corner. The cardinality constraint of `(1,1)` on the source of the association reveals that any context element is associated with exactly one application entity. Reusability and semantic relevance of context elements of the same class, such as that of the temperature of a building and of the temperature of a human is introduced in our model through well defined context types. Thus, based on a context type, we instantiate for multiple application entities distinct context. We should remark here that, at this level of abstraction, it does not make sense to distinguish sensed and calculated context as in the work of Dey. As an example, the temperature reading of a sensor or the social situation are both context elements that characterise the entity "room_A45" of a building.

From the semantic point of view, this simple context/entity model is enough to express definitions and formalisms found in [1, 17, 25]. Nevertheless, the difference between the approaches is the kind of information that composes the context and the flexibility to use references to application entities. As discussed in [9], different approaches use diverse data structures to express and exchange context information. The lack of a general, yet flexible, model makes the exchange of context information a tedious task.

We propose a solution to the problem by introducing a type system along with our context model concepts. The type system provides types for three major domains:

- Types defining application domain entities, *ApplTypes*. These are used to control the interoperability with the application through the associated entities. The detailed attribute definitions of the application types are out of the scope of a context model and therefore it suffices here to consider these type objects only as unique references to the actual application type. Imagine, for instance, types such as `user` and `location` of an application from the ubiquitous computing domain. Although the detailed type definition is not of interest, inheritance relationships involving them may be used to reveal entity compatibility.
- *Base Types* that define basic value types such as `integer`, `real` etc. The model offers further bulk types to express lists of values of a given type. Thus, we could, for example, specify a context element as a `set` of integer values or a set of application references to `user` entities. `BaseTypes` can also build hierarchies of restrictions. This is a very useful construct that allows syntactical and some semantic control over base values. A similar design is found in the simple data type declarations of XML Schema. In the scope of context, imagine the base types `celsius` and `fahrenheit` modelled as a restriction of type `real`. Although both can be regarded as real numbers, we can clearly define operators between them, namely, a Celsius to Fahrenheit transformer and vice versa.
- Types defining *ContextTypes*. Each context is well defined by its context type declaration which is composed of a set of attribute definitions. Attributes have a name and can be of any type, `BaseTypes`, `BulkTypes`, `ApplTypes` or further `ContextTypes`. This is an elegant way to model aggregated context as discussed in [1] without the need of introducing separate context concepts such as *Widgets* and *Servers*

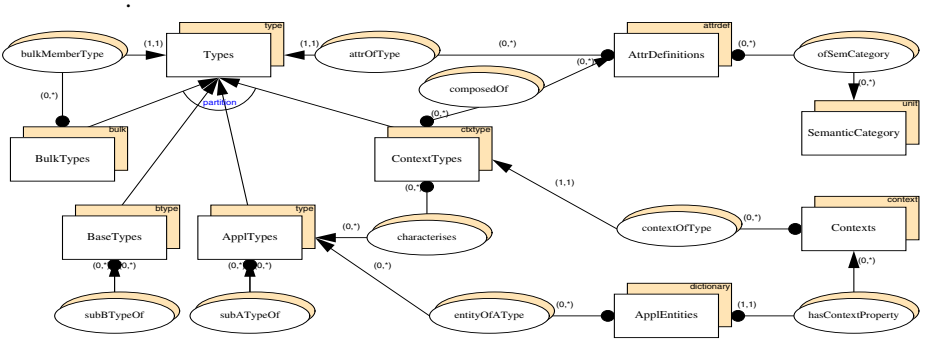


Fig. 2. Context Type Model

Attribute definitions can be classified through the `ofSemanticCategory` association to provide semantic categorisation of attributes. Imagine, for instance, a context-aware agenda application with a composite context element with two attributes, `capture_time` and `appointment_time`. Although both are of type `time`, the `capture_time` attribute designates the time that the context was captured and the `appointment_time` designates the actual time of the appointment. The context modeler can create a semantic category `capture_time` that will be used to classify context attributes for a particular purpose.

Attribute values specifying the quality or confidence of the context values could also be modelled by using the concept of semantic categories. As an example, imagine the confidence value of a face recognition sensor. Due to changes in the lighting conditions, the sensor might work better in some situations than in others. A framework based on our model can classify an otherwise convenient attribute definition, as a quality value. In other approaches, confidence might qualify a complete sensor such as a camera. Thus declaring the confidence of the measurement on the sensor level. By modelling the quality as a class of an attribute, we can define quality values for each measurement and not just a single unique value for all measurements of a sensor. Further, through the context composition, we can define the overall confidence of the sensor by composing the complex context information. Imagine, for instance, face recognition context based on a camera with a given quality value based on the camera's resolution. By using further sensors, such as lighting sensors or badge detection, we could create a composite face recognition context with a quality value that takes into account all gathered information.

The proposed type system is general enough to describe both simple value properties and complex composite context elements. By incorporating references to application types, context-aware applications can control the interaction with context information. Furthermore, context information not only consists of simple basic values, but can also use any application specific entity. In Fig. 2, this is apparent from the `attrOfType` association bound to `Types` with `ApplTypes` being a specialisation of it.

Nevertheless a type system can only ensure data type integrity and consistency, disregarding semantic constraints. By introducing restrictions on base types, subtypes of application types and semantic classification of attributes, the application can ensure a wide range of semantic constraints. An ontology as presented in [14] could be beneficial and its hierarchy, which is formed based on the constituents of each context type, can be modelled appropriately by composed contexts. For example, the `Environment:Sound:Intensity` type presented in the above mentioned work could be modelled with the following context types in our model:

```
contextType environmentContext characterising environment {
        sound: soundContext;
};

contextType soundContext {
        intensity: intensity; // restriction of string , enumeration
        frequency: hertz; // restriction of integer
};
```

The example defines two context types — the *environmentContext* and the *soundContext*. The first is constrained to be bound only to application entities of type *environment* or subtypes of it. Thus we can ensure context information especially declaring the environment of application entities of a given application type. Each of the contexts has property values, those of *sound, intensity* and *frequency*, respectively. The attributes of the *soundContext* are restrictions of `BaseType`, while the *environmentContext* context is a composite context that uses the *soundContext*.

5 Context Acquisition

Now that we have discussed the abstract notion of context and its type model, we will present the lower-level interface of context provisioning. We use the notion of a sensor as an abstract concept to represent a provider of a particular context element, for example, the location-ambience context element that is composed from the subcontext elements of temperature and lighting. The provider can be a hardware sensor, as in case of the light and temperature, or a software sensor, responsible for combining temperature and lighting to provide ambience values. Thus a sensor could either be bound to a simple context or a composite context.

To allow reusability and type safety, each sensor is an instance of some well defined *sensor type*. The sensor type defines an operation that is responsible for providing the information for the context. This operation can optionally have some parameters of any type defined in the model, with the purpose of controlling the execution of the operation execution (see `sensorParameterTypes` in Fig. 3). Further, a sensor type defines the type of context for which the sensor should provide values. With this information, the system can check that a sensor will always provide information of the correct context type.

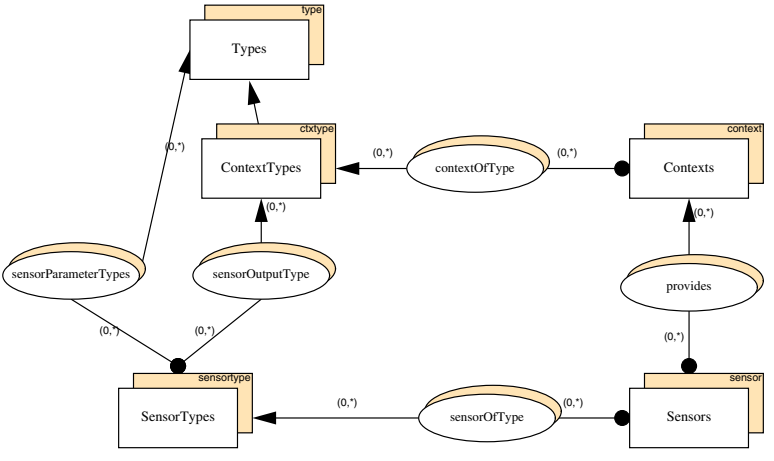


Fig. 3. Context Acquisition

The system provides a powerful algebraic query language based on set algebra that can be used for querying the system. The following constraint ensures that each sensor is only associated with a context of a type compatible to the sensor's output type:

```
( ("sensorOfType" compose "sensorOutputType")
    compose (inverse "contextOfType")
) minus "provides" = ∅
```

The left hand side of the constraint expression is a query that searches for sensors with output context type different than the type of the context bound to the sensor. The constraint checks that the result set is empty, meaning that there are no sensors bound to an incorrect output type.

Examples of sensors could range from as simple as a thermometer to complex pieces of software. To install a new sensor, the developer simply needs to initialise a sensor instance and bind the sensor to a given context. The sensor can either run individually and update the status of its context or it can be of a reactive nature, responding to an update request from its associated context.

6 Event Triggering and Application Binding

We discussed previously the fact that each context characterises an application entity. This is the only link between the application and the context system. Based on the well defined context model and typing, an application can easily access the context of interest. On the other hand, in reactive applications, it is necessary to adapt the application behaviour upon some context alternation.

We therefore need a mechanism for notifying parties interested in a given context change. Analogous to the concept of producers, we incorporate into our model the notion of notifiers. They notify subscribed application entities upon some occurring context event. As seen in the event model of Fig. 4, a notifier is subscribed to a context. Each time the context changes, the notifier is invoked. Based on some logic, the notifier evaluates the context change and, if desired, the appropriate application entity is notified. The specific notifier logic can be implemented in any language that has access and an interface to the context model and the application domain. In ongoing work, we are investigating a language based on boolean expressions and application callbacks to assist the notifier logic.

Again, our model allows the declaration of abstract notifier classes, based on which, multiple notifier instances can be instantiated. Imagine, for instance, a notify class that is subscribed to a room context. Whenever the lighting conditions change, the notifier is invoked. The notifier checks if the value is above or below a certain threshold and, if so, notifies the associated application entity by passing the corresponding context.

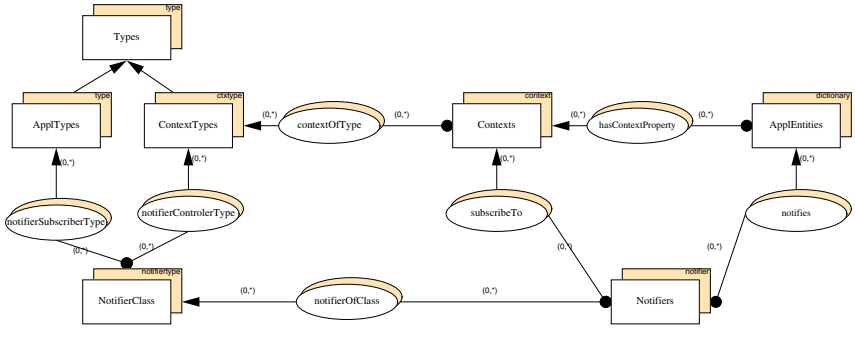


Fig. 4. Event triggering and application binding

It is not the purpose of our model to describe how to physically react to the incoming changes of context. These changes are propagated to the interested entities in the application layer, which are responsible for taking the appropriate actions. This feature makes our solution compatible with the active context awareness model presented in [16].

7 Conclusions

We have presented a general and abstract conceptual model for context and consolidated it with related work from various domains. The model supports a flexible and semantically expressive context representation that can be used for various applications scenarios. A sophisticated type system enables consistency checks at the system level and enables the definition and exchange of context information, something missing in the diverse data structure approaches used in earlier works. With the proposed model, it is possible to add new context elements at any time, and define new behavioural patterns for them, resulting in a very flexible and adaptable solution. Additional flexibility gives us the possibility to use object references as context values rather than just simple basic values. We believe that the proposed model contributes in the understanding and representation of context and could offer a common information model, based on which, application specific context models and ontologies could be defined.

Future work will include the implementation of the context model and its integration in a platform for ubiquitous and mobile information environments. We are currently developing a context-aware news application with a specific application context model based on the current proposal.

References

1. Anind K. Dey and Gregory D. Abowd. Towards a Better Understanding of Context and Context-Awareness. In *Proceedings of the CHI 2000 Workshop on The What, Who, Where, When, and How of Context-Awareness*, The Hague, Netherlands, apr 2000.
2. Luciano Baresi, Devis Bianchini, Valeria De Antonellis, Maria Grazia Fugini, Barbara Pernici, and Pierluigi Plebani. Context-aware composition of e-services. In *Proceedings of 4th International Workshop on Technologies for E-Services, 4th International Workshop (TES)*, pages 28–41, Berlin, Germany, September 2003.
3. Stefano Ceri, Florian Daniel, and Maristella Matera. Extending WebML for Modeling Multi-Channel Context-Aware Web Applications. In *Proceedings of MMIS'2003, International Workshop on Multichannel and Mobile Information Systems*, December 2003.
4. Anind K. Dey, Daniel Salber, Gregory D. Abowd, and Masayasu Futakawa. The conference assistant: Combining context-awareness with wearable computing. In *ISWC*, pages 21–28, 1999.
5. Anind K. Dey, Daniel Salber, Masayasu Futakawa, and Gregory D. Abowd. An architecture to support context-aware applications. Technical report, Georgia Institute of Technology, June 1999.
6. Paul Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1), 2004.

7. Michael Grossniklaus and Moira C. Norrie. Information Concepts for Content Management. In *Proc. Intl. Workshop on Data Semantics in Web Information Systems (DASWIS 2002)*, Singapore, December 2002.
8. Michael Grossniklaus, Moira C. Norrie, and Patrick Büchler. Metatemplate Driven Multi-Channel Presentation. In *Workshop on Multi-Channel and Mobile Information Systems, WISE 2003*, Roma, Italy, December 2003.
9. Guanling Chen and David Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.
10. Ramanathan V. Guha. *Contexts: A formalization and some applications*. PhD thesis, Stanford University, December 1991.
11. Martin Heidegger. *Sein und Zeit*. Neomarius Verlag, Tuebingen, 1927.
12. Jason I. Hong and James A. Landay. An Infrastructure Approach to Context-Aware Computing. *Human-Computer Interaction*, 16, 2001.
13. Cory D. Kidd, Robert Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth D. Mynatt, Thad Starner, and Wendy Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Cooperative Buildings*, pages 191–198, 1999.
14. Panu Korpipää and Jani Mäntyjärvi. An Ontology for Mobile Device Sensor-Based Context Awareness. In *CONTEXT 2003, LNAI*, volume 2608, pages 451–458, Berlin, 2003. Springer-Verlag.
15. Henry Lieberman and Ted Selker. Out of context: Computer systems that adapt to and learn from context. *IBM Systems Journal*, 39(3):617–631, 2000.
16. Louise Barkhuus and Anind K. Dey. Is Context-Aware Computing Taking Control Away from the User? Three Levels of Interactivity Examined. In *UbiComp 03 Conference*, Seattle, oct 2003. To Appear.
17. John McCarthy. Notes on formalizing contexts. In Ruzena Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 555–560, San Mateo, California, 1993. Morgan Kaufmann.
18. Surav Mehmet and Akman Varol. Modeling context with situations. In *Proceedings IJCAI-95 Workshop on Modelling Context in Knowledge Representation and Reasoning*, pages 145–156, Montreal, Canada, 1995.
19. Moira C. Norrie. A specification of an object-oriented data model with relations. In *Proceedings of International Workshop on specifications of Database Systems*, pages 211–227, Glasgow, July 1991.
20. Moira C. Norrie. An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. In *Proceedings of ER'93, 12th International Conference on the Entity-Relationship Approach*, December 1993.
21. Moira C. Norrie and Alexios Palinginis. A Modelling Approach to the Realisation of Modular Information Spaces. In *14th Conference on Advanced Information Systems Engineering (CAiSE'02)*, May 2002.
22. Moira C. Norrie and Alexios Palinginis. Empowering Databases for Context-Dependent Information Delivery. In *Ubiquitous Mobile Information and Collaboration Systems (UMICS), CAiSE Workshop Proceedings*, June 2003.
23. Moira C. Norrie and Alexios Palinginis. Versions for context dependent information services. In *Conference on Cooperative Information Systems (COOPIS 2003)*, Catania-Sicily, Italy, November 2003.
24. Moira C. Norrie, Alain Würgler, Alexios Palinginis, Kaspar von Gunten, and Michael Grossniklaus. *OMS Pro 2.0 Introductory Tutorial*. Institute for Information Systems, ETH Zürich, March 2003.

25. Jason Pascoe. Adding Generic Contextual Capabilities to Wearable Computers. In *ISWC*, pages 92–99, 1998.
26. Manuel Román, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt. Gaia: A middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, pages 74–83, Oct-Dec 2002.
27. Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The context toolkit: Aiding the development of context-enabled applications. In *Proceedings of the 1999 Conference on Human Factors in Computing Systems (CHI '99)*, pages 434–441, Pittsburgh, PA, May 1999.
28. Bill N. Schilit. *A System Architecture for Context-Aware Mobile Computing*. PhD thesis, Columbia University, 1995.
29. Bill N. Schilit, Norman Adams, and Roy Want. Context-Aware Computing Applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, 1994.
30. Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.
31. Terry Winograd. Architectures for Context. *Human-Computer Interaction*, 16, 2001.

Distributed Task Processing Within the Mobile Memory Aid System MEMOS

Andrei Voinikonis, Klaus Irmscher, and Hendrik Schulze

Leipzig University, Institute of Computer Science,
Augustusplatz 10-11, D-04109 Leipzig, Germany
{voinikon, hendrik, irmscher}@informatik.uni-leipzig.de

Abstract. MEMOS is a nomadic computing system to support patients with disturbances in the prospective memory. The system consists of two parts which are loosely connected via GPRS: a mobile electronic device (Personal Memory Assistant - PMA) to remind the patient of important tasks and a Base Station that coordinates the activities of caregivers and notifies them about the result of the task execution. A major requirement of MEMOS is the autonomous operation of the PMA. This is accomplished by avoiding mobile transactions, the specification of temporal aspects in the task description, introduction of different states for tasks of the PMA and the Base Station, and by splitting task and result management. Special XML based languages were developed for data exchange between Base Station and PMA to reflect the temporal aspects of the tasks and for logging the results of task execution.

1 Introduction

The availability of mobile devices, which allow an Internet access, enables the introduction of new kinds of services. The treatment of patients with memory deficits is one possible field to apply the achievements of communication technology. The patients now have an opportunity to receive operative instructions from their caregivers everywhere. The caregivers can control the execution of these instructions and react when the patient got into a critical situation [1]. In this way the patient retrieve autonomy, treatment costs can be reduced and the caregivers (therapists, relatives or friends) are unburdened.

A concept is to furnish the patient with a mobile device containing descriptions of the tasks, the patient would forget otherwise. This device reminds the patient when the next partial stage of a task should be performed, and guides him through different steps of the started task. A task can be understood as the intended action of the patient as well as the formalized representation of this intention in the memory aid system.

The Mobile Extensible Memory Aid System (MEMOS) is introduced to provide the end-to-end task processing for persons with deficits in the prospective memory and for their caregivers. Fig.1. shows the general architecture of the system.

MEMOS consists of the Base Station and mobile components (PMA). The Base Station is a centralized component, which coordinates the actions of the various caregivers of one patient. It assists the caregivers by task settings, coordinates the task settings, transmits the tasks the PMA, handles and presents the results of task execution and performs other jobs.

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 57–68, 2004.

The PMA is a hand-held computer with permanent Internet access over a cellular phone connection, which realizes the presentation of a task to the patient and guides the patient through the task. For this, the mobile device downloads task descriptions from the Base Station, prepares tasks for presentation and traces the task execution.

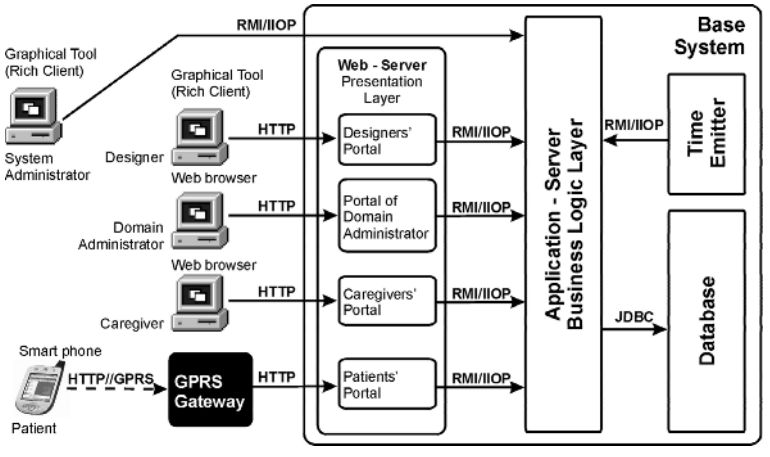


Fig. 1. General architecture of MEMOS

This article discusses the distributed task processing on the Base Station and on the mobile device. The costs of the system operation should be minimized. The charge of batteries in the PMA should be also economized. Furthermore, the system should provide the high availability of tasks, even if patient leaves an area of stable reception for a while. Therefore, the number of connections and connection duration should be minimized. Consequently, the mobile device should be able to work autonomously, i.e. the connection to the Base Station has to be established rarely, and failure of some connection attempts should not significantly influence the operation of the system.

To meet the requirements, the PMA downloads tasks in advance or on demand, stores and presents them to the patient at the specified time. The task caching on the mobile device and low frequency of connections to the Base Station raise some problems for task processing and management. The delay between transmission and presentation of a task requires the introduction of a special language for task description, a language that reflects the temporal peculiarity of such a tasks. A language for logging and transmitting the results of task execution is also required. This article introduces the required languages and describes the approach to ensure consistency and concurrency in the distributed task processing on the MEMOS system.

2 Task Processing

The main goal of MEMOS is to support patients with deficits in the prospective memory in their daily life, by assisting them at the fulfillment of their duties. The challenge is

to develop a system that provides a sufficient control over the task processing to the caregivers, without confining the patients' freedom. This is achieved by the use of a mobile device (PMA) as reminder, which can work autonomously over a longer period. The PMA establishes a GPRS connection to the Base Station only rarely, and even in areas without reception it can work as intended for a while.

In the typical treatment scenario a caregivers plans the task schedule of a patient together with him. Therefore, the caregiver sets the tasks using suitable templates (task plans). The Base Station of MEMOS stores the tasks and produces a description of the task in the XML based description language M2 [4]. Listing 1. shows the simplified M2 description of the task for medicine taking.

```
<deck label="Medicine Taking" id="D2:1:0">
  <control>
    <time start="100" exec="200" end="400" dur="100"/>
  </control>
  <card id="D2:1:0_0">
    <loop time="99" reference="D2:1:0_3"/>
    <log>Task for Aspirin taking is started!</log>
    <text>Please take Aspirin now!</text>
    <button label="OK" reference="D2:1:0_1"/>
    <button label="Cancel" reference="D2:1:0_2"/>
  </card>
  <card id="D2:1:0_1" type="invisible">
    <loop time="1"/> <!-- no reference: exit -->
    <status end="yes"/>
    <log>Aspirin is taken.</log>
  </card>
  <card id="D2:1:0_2" type="invisible">
    <loop time="1"/>
    <status critical="yes" end="yes"/>
    <log>Medicine taking is cancelled!</log>
  </card>
  <card id="D2:1:0_3" type="invisible">...</card>
</deck>
```

Listing 1. Simplified M2 description of a task

A task is a set of linked atomic information units. To avoid that the patient is confused by scrolling through the information, every unit has to fit on the PMAs' screen. Such an atomic information unit is called a "card". To every card several actions (like pressing a button) could be bound, to give the patient the chance to react to the information provided. The PMA downloads new task descriptions periodically, then it parses the transmitted tasks, creates corresponding objects and stores them in a list of scheduled tasks. After that, the mobile device confirms the task reception.

When a task is started, the PMA alerts the patient and guides him through the defined steps of the task, displaying the cards [4]. The required actions are described and a choice of possible reactions is specified and displayed as buttons on the touch screen. The patient navigates through the task pressing one of the specified buttons. Depending on which

button had been pressed, a new card will be displayed. If the patient does not react in a certain time the mobile device navigates through the task graph according to the elements of time flow control. The PMA stores identifiers of each started task and of each displayed card in a log file and transmits the log information to the Base Station periodically. If the task execution reaches a critical state (a card marked as critical), the mobile device transmits the log information to the Base Station immediately.

The Base Station evaluates the log information and displays it to the caregivers of the patient. When a critical state occurs, the responsible caregivers will be alerted by SMS, email or a notification on their screen.

2.1 Concurrent Task Processing in a Loosely Connected Environment

The most complex problem to solve is the consistent management of the task states for the autonomously working mobile devices.

From the moment of task transmission to the PMA, two instances of the task exist: one on the Base Station and another on the mobile device. The management of the distributed instances is realized under concurrent accesses by: (1) the Time Service of the Base Station, (2) caregivers and (3) patient. Examples of concurrent operations are:

- removal of a task by the caregiver (Base Station), postponement or premature execution (manual start) of a task by the patient (PMA);
- start of a new task by the caregiver (Base Station), postponement or premature execution of a task in an overlapping interval by the patient on the mobile device.

The task management model will need to include aspects of long transaction models [6]. Thus, transactions started on the Base Station may require an especially long time for their processing, but the autonomous work of the PMA anticipates a suitable integration of transactions.

Because any of the ACID-transactions may fail, thereby irritating the patient, they are no option. Even the abandonment of the ACID paradigm does not solve the problem because some transactions started by the caregiver may not be compatible with transactions started by patient, e.g. an already executed task cannot be deleted. Since the PMA establishes connections rarely, aborting a transaction started by the caregiver can take place a long time after its start, which is also not acceptable. To avoid mobile transactions, the following measures can be taken:

1. Coarse-grained long-lived transactions have to be split into sequences of short ACID transactions that have to be executed only on the server side. For this, the number of accepted states of a task is increased: e.g. the states "Submitted, not confirmed" and "Submitted, confirmed" (Fig.2.) are introduced on the server side for task transmission. Thereby, the logging of events such as exceeded confirmations of task reception is performed to inform caregivers.
2. The different states are introduced for a task on the server and on the client side, which is dictated by different goals that have to be achieved: the Base Station has to submit the task descriptions to the PMA and the PMA has to provide the task presentation to the patient at the specified time.
3. Locking activities on the Base Station after task submission: caregivers can only watch a tasks state after its submission, all activities regarding the temporal task

relocation and abort are accepted from the patient only. This is acceptable to the caregivers because the tasks are submitted just before their execution, to keep the locking period short.

4. The overlap of time intervals of several tasks is permitted. If one task should be started when another task is active, this task waits in until the active task has been completed. This decision bases on the following facts:
   - A short time of active task execution in contrast to a much more longer time, when a task is valid. The short time of waiting does not influence the achievement of task purposes
   - Long waiting time due to the absence of a patients reaction on the active task is not important for tasks in the background, because, if the patient has not reacted on the active task, no reactions on the waiting tasks can be expected either.
5. The management of tasks on the Base Station is separated from the handling of task execution results. The Base Station assumes that the task execution runs successfully unless the converse is detected. The patient executes the task on the PMA out of a transaction context. As mentioned above, all reactions of the patient and all time events are logged. All these events lead to a state change of the task on the client side. By displaying a new card, the PMA stores the identifier of the card, the history record and the state flags specified within the card. This information gives the caregiver the complete picture of the task processing on the mobile device. The result of the task execution on the PMA has no influence on the task state on the server side. Only the patients' schedule should be updated after the postponement of a task. This gives caregivers the current view on the patient's task schedule. Nevertheless, it has no influence on the state of the task on the server side.

Such measures provide the stable operation of the system without significant infringement on the caregiver's interests.

2.2 Task Processing on the Base Station

The Base Station has to provide the task management from creating a task to the presentation of the task execution results. It has to provide the task submission to the PMA and has to make the process of transmission transparent to the caregivers. A failure in task submission has to be reported immediately. The task states on the server side are introduced to achieve these goals.

According to the wishes of the patient, the caregivers set the tasks in advance. The Base Station calculates the first time of a state alteration for each started task. The calculation is performed based on the parameter values given while setting the task. After that, the Base Station stores the tasks and controls their life cycle. The states of the tasks life cycle on the Base Station are shown on Fig.2.

The events generated by the Time Service of the Base Station, the task transmission and caregiver interventions cause the alteration of the task state. If the execution time of a certain task approaches, the Base Station activates the task (Fig.2. State of the task becomes "activated"), produces the task description in XML (in M2 format, described in part 4) and makes it accessible to the PMA. The download of the task description initiates the transition to the next state (Fig.2. "submitted, not confirmed") demonstrating that the

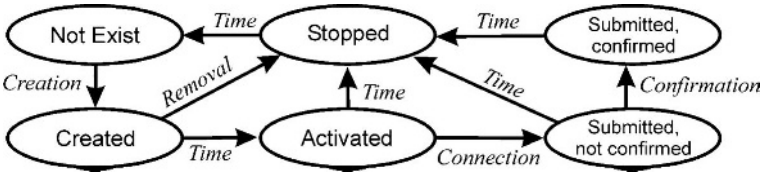


Fig. 2. Simplified task life cycle on the server side

Base Station is waiting for a confirmation of successful submission to the mobile device. This state is introduced because transmissions to the PMA are fault prone. Therefore, the confirmation for the task reception had to be made explicit.

When the PMA confirms the reception of the task, the Base Station changes the tasks into the state "submitted, confirmed" (Fig.2.). This state represents the successful task submission. The introduction of these states makes the process of task transmission transparent to the caregivers: the number of download attempts, the confirmation delays and transmission failures detected by the time check are noted in the history of the task processing engine. Based on this information, the caregiver can decide whether intervention (with other means) is required.

After the transmission timeout, the task becomes inaccessible to the PMA, independent of the state of the task. If no attempts to download the task were made, the Base Station alerts the responsible caregivers. If the task download was not confirmed, a corresponding record is made in the task history. The confirmations made after time expiration are also recorded, what shows caregivers that intervention is not required.

The last state, "stopped", is introduced for further needs such as a task restart on the Base Station. The task remains in this state as long as specified in the configuration data. After expiration the Time Service of the Base Station removes the task.

2.3 Task Processing on the PMA

The PMA has to enable the comfortable management of submitted tasks by the patient and has to display the information of the task. To achieve these goals, the software of the PMA is developed to meet the following requirements [3]:

1. The PMA has to work autonomously
2. The mobile device has to be able to guide a patient continuously through the different steps of a task. Thus, if the patient was interrupted, he should find his way back to the task.
3. The patient has to be able to observe all submitted tasks.
4. To start intentions manuaqlly
5. To postpone intentions and
6. To see missed intentions
7. The caregiver has to be able to define the automatic start time of a task, to draw the patient's attention, e.g. taking medicine
8. To restrict the execution and postponement of an intention.

The mobile device periodically establishes a cellular phone (GPRS) connection to the Base Station to transfer the log file and to download the descriptions of new accessible

tasks. If the task descriptions are successfully downloaded, the PMA parses the descriptions, creates and stores corresponding data-objects, defines and registers the time check points to control the task processing and sends the confirmation of the reception to the Base Station. The loading of tasks and the ability of the run-time environment to handle time events, which are specified in the task description, allows the autonomous work of the PMA (req.1.).

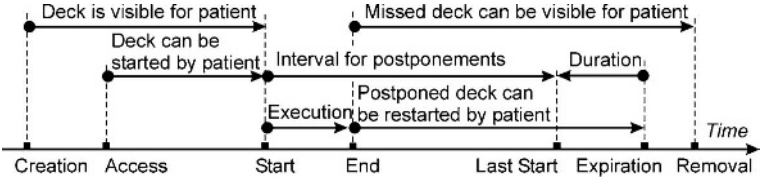


Fig. 3. Time diagram for independent deck

The mobile device uses decks for the task processing in contrast to the Base Station. A deck is a non-interrupt able part of a task, combining the temporally linked actions. The partitioning of tasks to decks facilitates the software development for the PMA, and allows the flexible execution and postponement of decks. The deck structure of a task is defined by the task plan. A caregiver fixes the temporal relations between decks by setting time points under control of the Base Station. The execution of a deck must not be interrupted by another to avoid confusion of the patient. The application of cards and decks for task construction provides continuous guidance of the patient through the task (req.2).

The partitioning of tasks into decks requires two kinds of decks regarding their activation: (1) independent decks, which can be executed without any preparation and (2) dependant decks, which need some precondition to be started. Dependent decks may be started after their activation only. A dependent deck is activated if another deck initiates the navigation to some card of the dependent deck during its execution. The analysis of the requirements allows to divide the lifetime of a deck into some intervals, shown on Fig.3.

After creation (Fig.3. "Creation") or activation of a dependent deck all decks should be visible for the patient, for a better overview of the day's schedule (req.3.).

Some intentions such as work at home or preparation to an appointment can be performed when the patient has spare time. Such kinds of intention should possess a long interval of availability (Fig.3. interval between "Access" and "Expiration"), where they can be started or postponed (req.4, 5.). Some intentions do not need to be started automatically in the specified time. Otherwise, some intentions such as taking medicine should be started at a certain time and should only have a short interval of availability. Caregivers should specify the intervals according to the task purpose. If a caregiver wishes to restrict the execution of a deck, the "Access" time can coincide with the time of the automatic start and the time between "Access" point and "Expiration" should be equal to the duration of the deck (req.8). To fulfill the requirement of an automatic start (req.7), a special time for the automatic start can be specified for the decks. This demands

the introduction of two kinds of decks regarding automatic start. It is provided by means of a task description language and of the run-time environment.

When the specified time (Fig.3. "Access") is reached, a deck can be started by the patient. The start of the deck (Fig.3. "Start") can be performed either manually or automatically at the time specified by the caregiver or chosen by the patient during the postponement of the task.

Decks can offer the patient to postpone themselves after their start. A special element for postponement is included by the run-time environment in the first card, if postponement is permitted according to the deck description and is allowed due to the deck schedule: the deck schedule possesses a suitable gap for execution of the postponed deck. The element refers to an additional card generated by the run-time environment, which proposes some available time points for restart. The time for postponement is calculated due to external conditions: the specified temporal data of the current deck, the schedule of other decks and their assumed duration.

The state diagrams have been developed according to the time diagram. The states of two kinds of decks on the PMA are shown on Fig.4. After the reception of a deck description, the PMA creates the deck object and waits for the activation of dependent decks (Fig.4.2. "Created"). If the activation has not occurred until deck expiration (Fig.3. "Expiration"), the dependent deck is removed from the PMA. After the activation of a dependent deck or after creation of independent decks the PMA includes their titles in the schedule (Fig.4.1. "Displayed"). The further execution of both kinds of decks is similar.

If the navigation to a card of a deck in the state "Displayed" is initiated by the execution of another deck, the time of automatic start for this deck is set just after the time point "Access". Thus, once the time point "Access" is reached, the execution of the deck is started with the card called by the other deck. The deck remains in the state "Displayed" (Fig.4.) until the time point permitting the manual start (Fig.3. "Access") is reached. At this moment, the deck will be changed to "Executable" (Fig.4.). It stays in this state until manually or automatically started.

Then, the state of the deck is changed to "Waiting" (Fig.4.) after the start. The run-time environment checks whether a deck is executed or other decks are waiting for execution, and puts the deck in a priority-controlled queue. When decks turn comes, the decks state changes to "Running" (Fig.4.). As this takes place, the cards of the deck are displayed according to the reactions of the patient and the time events. If the patient decides to postpone the deck, its state is changed back to "Executable" (Fig.4.). All reactions of patient and all time events are logged.

If the execution of the deck reaches a card marked as critical or the deck is postponed, the PMA transmits the log information to the Base Station immediately. Otherwise, the log information transmission is performed as usual.

If no patient reaction follows, the decks state is changed to "Missed" (Fig.4.). Decks in this state are visible for the patient (req. 6), but are not accessible. This helps the patient to keep the overview over failed tasks and facilitates collaboration with the caregiver by rescheduling tasks. If the time for deck execution expires, the decks state is also changed to "Missed".

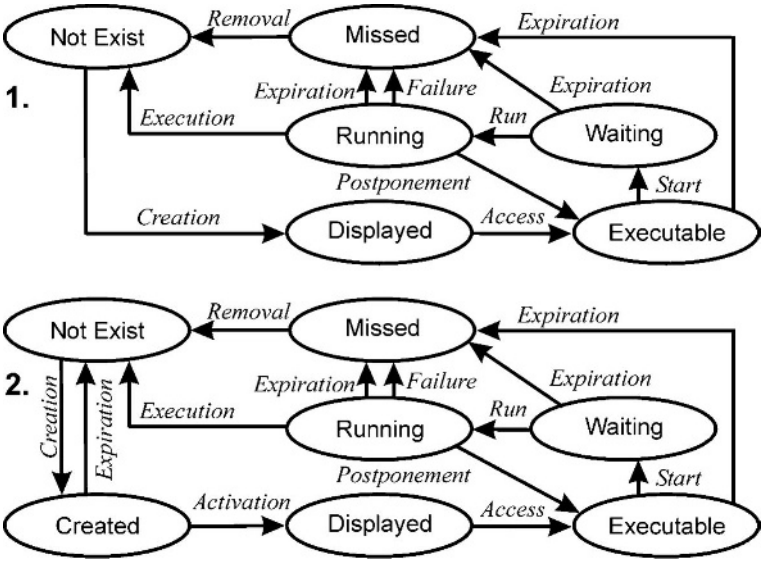


Fig. 4. Simplified deck life cycle on the mobile device 1 - Independent deck; 2 - Dependent deck

This approach allows us to meet all mentioned requirements, provides comfortable task management for the patient and keeps the task processing transparent for caregivers.

3 Communication Languages

Each distributed system needs means to exchange information. As far as possible, MEMOS uses standard protocols such as HTTP, but to reflect the peculiarity of the system, special communication languages are need.

XML is a good choice to construct a language for information exchange. On the one hand, it allows restricting of the syntax to describe a certain domain by DTDs or XML schemas [5] on the other hand, standard tools for parsing, syntax control and creation of object structures an available. Additionally, the specification of XML based languages can easily be adopted and extended.

The bi-directional communication of the Base Station with PMA requires two languages: one for a task description and transmission, other for logging and transmission of the execution results.

3.1 M2 - an XML Based Language for Task Description

M2 stands for MEMOS Markup Language Version 2. M2 was developed because existing markup languages were not completely applicable for MEMOS.

HDML (Handheld Device Markup Language) is a simple language used to create hypertext-like content for small display, handheld devices [2]. WML [8] is intended to

replace HDML, because HDML does not fulfill XML requirements. Both languages do not reflect the temporal aspect of information presentation important for MEMOS.

The task description language for patients with memory deficits should feature: (1) temporal aspects of the deck life cycle for the run-time environment, allowing the autonomous operation of the PMA, (2) structure of task graphs which determines the relations between actions needed to reach a tasks goal, (3) available alternatives for the patient at task processing: postponements and the possibility to start a deck manually.

The task description consists of decks and cards analogous to WML. The actual task execution on the PMA is the navigation through the decks and cards according to the relations between the cards.

M2 specifies two kinds of events: with and without embedded description of event handlers. On the one hand, it has to specify the temporal aspects of the deck life cycle. The time is specified for each deck when its state has to be changed. Thereby, no event handlers are specified for them in the deck description; their handling is realized only by the run-time environment.

On the other hand, M2 has to fixate the structure of the task graph. Elements specified by the event description, such as timer or user events (buttons), contain a reference to the target card and a time interval for the timer resp. a label for buttons. This strict specification of the relations between event description and event handler in one element allows to fix the graph structure. Event elements represent the edge in the graph structure. If an event is initiated, the run-time environment navigates to the card, which is referenced by the event handler, and displays it to patient. If no card is referenced, the run-time environment finishes the deck execution.

Deck. In contrast to the above-mentioned languages, a deck is the autonomous noninterruptable part of the task, defining an organized structure of temporal linked actions to achieve sub-goals of a task. Due to Fig.3., the deck control unit specifies the time when the deck can be made accessible for manual start, the time of expiration, the duration of the deck and the automatic start time. Additionally, the deck control unit can specify the deck timer. The deck timer is intended for the specification of time events within the decks scope, providing a better control of the duration thereby.

Further, each deck possesses a title, which is shown to the patient in a list of scheduled tasks, and a deck can specify the history record that should be added to the log file when the deck is started.

Card. Each deck consists of cards and each card represents a single interaction with the patient. It describes a single step in the task execution and offers the patient the possibility to react, e.g. to confirm or reject the step. Each offered reaction specifies a reference to another card. The navigation through the deck proceeds depending on the patients reactions. Each card possesses a unique identifier to referring to event handlers. Each card has to specify:

- its own timer, which exposes a means of processing patient's inactivity
- frequency of alerts aimed to draw the patient's attention.

Each card can specify:

- updates for the deck timer
- the state of task such as "critical" or "end" state which is set if patient reaches the card
- the record that should be written into the log file if the patient reaches the card
- the action title and description;
- the event handlers for the reactions of the patient.

3.2 Language for Result Logging and Transmission

The simple XML based language was developed for result logging and transmission. As mentioned above, the mobile device logs all patient reactions and time events during the task processing. The log data is transmitted to the Base Station periodically or immediately, if a task reaches a critical state, a patient postpones a task or a reception for a new task should be confirmed. The transmitted information should reflect the alteration of states of tasks on the mobile device and the course of task execution. To this effect, the log data contains the following kinds of notification:

- the trace of the task execution: timestamps and identifiers of started decks and displayed cards with their description and state flags. Each record possesses the timestamp indicating the time of the event. This construction of the history gives the caregiver a complete picture of progress on the PMA
- the confirmation of reception of new decks
- the confirmation of commands reception
- the postponements of decks to refresh the patients' schedule on the server
- debug information, which is logged by the web server and can be used to detect failures and to determine their reason.

The foregoing structure of log data reflects the sufficient picture of task processing and execution.

4 Application

MEMOS is successfully utilized as memory aid system at the Outpatient Clinic for Cognitive Neurology at University of Leipzig, were it is not only used as memory aid system but also as platform for further neuropsychological research [4]. MEMOS has been tested with 10 patients. 80% of patients with deficits of the prospective memory have benefited from it. Further evaluation with a bigger group of patients will be accomplished in the year 2004.

5 Summary

The proposed method of task description and the scheme for data exchange allows the autonomous work of the mobile device. The introduced measures to avoid mobile transactions enable the stable operation of the system without significant infringement on caregiver's interests. The introduction of different task states for the server and for the

mobile device is conditioned by the different goals to achieve. The proposed determination of the task states defines the appropriate task behavior. The developed languages for data transfer represent the peculiarity of system data well.

References

1. Thöne-Otto, A., v.Cramon, Y., Irmscher, K., Schulze, H., MEMOS - Mobile Extensible Memory System: Elektronische Gedächtnishilfe für hirngeschädigte Patienten, Deutsches Ärzteblatt 98, Heft 11 vom 16.03.01, **14**
2. King, P., Hyland, T., Handheld Device Markup Language Specification, URL: `www.w3.org/pub/WWW/TR/NOTE-Submission-HDML-spec.html`, 1997
3. Thöne-Otto, A., and Walther, K., How to design an electronic memory aid for brain-injured patients: Considerations on the basis of a model of prospective memory, INTERNATIONAL JOURNAL OF PSYCHOLOGY, 2003, 38 (4)- Preprint International Union of Psychological Science
4. Schulze, H., Voinikonis, A., Hoffmann, T., Irmscher, K., Modeling a Mobile Memory Aid System, Proceedings zur 13. ITG/GI-Fachtagung "Kommunikation in Verteilten Systemen" (KiVS2003). 25.-28.02.2003, Universität Leipzig. Springer-Verlag, Reihe Informatik aktuell. Berlin, Heidelberg, New York, 2003. S. 143-153
5. Ahmed, K. et. al., Professional Java XML, Wrox Press, 2001
6. Segun, K., at. al., Transaction Management in a Mobile Data Access System. In: Yuen Chung Kwong (ed.), Annual Review of Scalable Computing, Vol. 3, Singnapore University Press, ISBN 981-02-4579-3, 2001.
7. Schulze H., Irmscher K., Mobtel - A Mobile Distributed Telemedical System, Proceedings of USM 2000 - Trends in Distributed Systems: Towards a Universal Service Market Third International IFIP/GI Conference, Munich, Springer, 2000, pp. 176
8. Wireless Markup Language Version 1.3, WAP Forum, WAP-191-WML-20000219-a. URL: `www.wapforum.org`

Towards an Approach for Coordinating Personalized Composite Services in an Environment of Mobile Users

Zakaria Maamar[1], Quan Z. Sheng[2], and Boualem Benatallah[2]

[1] College of Information Systems,
Zayed University, Dubai, U.A.E
zakaria.maamar@zu.ac.ae
[2] School of Computer Science and Engineering,
The University of New South Wales, Sydney, Australia
{qsheng, boualem}@cse.unsw.edu.au

Abstract. This paper presents an approach for coordinating personalized composite services, which are intended to be offered to mobile users. A composite service is an aggregation of several component services either primitive or composite services. By coordination, it is meant the mechanisms that specify the orchestration of the component services of a composite service. The orchestration concerns the execution chronology of the component services, the data that the component services exchange, the states that the component services take, and the actions that the component services perform. By personalization, it is meant the integration of preferences of users into the specification of the orchestration of the component services. Preferences concern when and where the component services need to be executed. This execution is outsourced to software agents, which consider the respective contexts surrounding users and Web services.

1 Introduction and Motivation

Web services are among the technologies that help businesses in being more Web-oriented. Business-to-Customer (B2C) cases identified the first generation of Web services. More recently, businesses are using the Web as a means for connecting their business processes with other partners through what is commonly referred to as Business-to-Business (B2B) Web services. The advantages of Web services shed the light on their capacity to be composed into high-level business processes known as *composite services*. Multiple technologies back the widespread of Web services including WSDL, UDDI, and SOAP [1].

Composing Web services (also called *e-services*)[1] rather than accessing a single service is essential and offers better benefits to users. Composition addresses the situation that a user's request cannot be satisfied by any available service individually, whereas it might be satisfied by a composite service that is obtained by combining a set of

[1] Terms e-service, Web service, and service are used interchangeably.

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 69–82, 2004.

existing services [2]. Discovering and selecting the component services according to the requirements of users, inserting the selected component services into a composite service, triggering the composite service for execution, and finally monitoring the execution of the composite service are among the operations that users will be responsible for. Because of the complexity featuring most of the aforementioned operations, it is deemed appropriate considering *software agents* to assist users in these operations [3]. A software agent is an autonomous entity that acts on behalf of user, makes decisions, interacts with other agents, and migrates to distant hosts if needed.

Berardi et al. report in [2] that up to now research on Web services has mainly concentrated on three aspects: (i) service description and modeling, (ii) service discovery, and (iii) service composition. One of the composition operations that we look into in this paper is the *coordination* of personalized composite services in an environment of mobile users. On the one hand, by coordination it is meant the mechanisms that specify the orchestration of the component services of a composite service. The orchestration is about the execution chronology of the component services, the data that the component services exchange, the states that the component services take according to this exchange, and the actions, either regular or corrective, that the component services perform. On the other hand, by personalization it is meant the integration of the preferences of users into the specification of the orchestration of the component services. Panayiotou and Samaras notice in [4] that the needs of mobile users regarding information access are quite different from the needs of stationary users. Needs of mobile users are not about browsing the Web but about receiving personalized content, which is highly sensitive to their immediate environment and requirements.

In this paper, users' preferences are identified at two levels: *location* and *time* (i.e., where and when a mobile user would like to have services selected to satisfy his needs being performed). Through appropriate mechanisms, users will be given the opportunity to adjust the specification that coordinates the component services according first, to their personal preferences and second, to the features and constraints of the environment where they will be located (e.g., is there any network coverage? Is there any printer in the area?). Keeping track of the status of the user-adjusted specification requires some *awareness mechanisms* that detect for instance the changes of the environment in which the specification is being executed. These mechanisms are installed on top of a structure that is referred to as *context*. Context is the information that characterizes the interaction between humans, applications, and the surrounding environment [5].

To carry out the coordination of personalized composite services, additional software agents besides the agents that support users in their operations are deemed appropriate. These agents are deployed on top of the component services, each component being associated with a *service chart diagram* [6]. This diagram describes a service first as an independent component and second as a member of a composite service. Using a service chart diagram, the description of a service occurs from five perspectives namely state, business, performance, date, and chronology. Because of the mobility of users and the integration of their preferences into the specification of composite services, it will shown in this paper that two extra perspectives, namely *location* and *time*, need to be embedded into a service chart diagram. We base the semantics of location and time perspectives on the corresponding works of [7] and [8].

Web services composition is a very active area of research and development [9]. However, *very little* has been accomplished so far regarding the personalization of Web services for the benefit of mobile users. In particular, several obstacles still exist such as (i) lack of techniques for modeling and specifying the integration of personalized information into Web services, (ii) Web services are dealt with as passive components rather than active components that can be embedded with context-awareness mechanisms, (iii) existing approaches for service composition (e.g., WSFL and BPEL4WS) typically facilitate orchestration only, while neglecting information about the context surrounding users and services, and (iv) existing coordination approaches (e.g., WS-Coordination) are not aimed at supporting the deployment of personalized Web services. In this paper, we discuss our research in-progress on coordinating personalized composite services in an environment of mobile users. Section 2 overviews some of the concepts such as Web services and context. Section 3 is about agentifying Web services coordination and personalization. Section 4 talks about the value-added of context to service personalization. Finally, Section 5 concludes the paper and discusses conversations as part of future work.

2 Some Definitions

A software agent is an autonomous entity that acts to perform tasks on behalf of user [10]. A software agent has a number of features that make it different from other traditional components including autonomy, goal-orientation, collaboration, flexibility, self-starting, temporal continuity, character, communication, adaptation, and mobility. It should be noted that not all these features have to embody a software agent.

A Web service is an accessible application that other applications and humans can discover and invoke. Benatallah et al. provide the following properties of a Web service [11]: (i) independent as much as possible from specific platforms and computing paradigms; (ii) mainly developed for inter-organizational situations (e.g., B2B applications); and (iii) easily composable so that developing complex adapters for the needs of composition is not required.

According to Dey, context is any information that is relevant to the interactions between a user and an environment [12]. This information can be about the circumstances, objects, or conditions by which the user is surrounded. Many researchers have attempted defining context. Schilit et al. decompose context into three categories [13]: computing, user, and physical. For example, user context category is about user profile, location, nearby people, and even sometimes social situation. Physical context category is about lighting, noise levels, traffic conditions, and temperature.

According to Papazoglou and Georgakopoulos in [9], coordination is to "*control the execution of the component services, and manage dataflow among them and to the output of the component service*". In addition to the communication language component, coordination is another core component that contributes to collaboration. The purpose of coordination is to prevent conflicts on various matters (e.g., computing resources) by specifying who is in charge of doing what and when and where this has to be done. Coordination is principally about the specification and assignment of responsibilities to

appropriate participants that could for instance be software agents. Coordination is of type implicit and explicit [14]. On the one hand, implicit coordination assumes that the participants are aware of the existence of certain rules to adopt. The rules are predefined and related to an application domain such as car traffic. On the other hand, explicit coordination requires a clear specification of the responsibilities and the mechanisms by which conflicts are solved. Negotiation, voting, or authority intervention are samples of these mechanisms.

A service chart diagram is a means for modeling and specifying the component services of composite services [6]. In fact, it enhances an UML state chart diagram, putting this time the emphasize on the environment surrounding the execution of a service rather than only on the states that a service takes (Fig. 1). More details on a service chart diagram are given in [6]. Because a composite service is made up of several component services, the process model underlying the composite service is specified as an UML state chart diagram (the value added of UML state charts to Web services composition is discussed in [15]). In this diagram, states are associated with the service chart diagrams of the component services, and transitions are labelled with events, conditions, and variable assignment operations. For illustration purposes, Fig. 2 shows *travel assistant composite-service* as a state chart diagram. It is composed of several dependent services, each service is associated with a service chart diagram: *flight booking* (FB), *hotel booking* (HB), *attraction search* (AS), and *car rental* (CR). There is an AND-state in TAS in which AS is performed in parallel with HB.

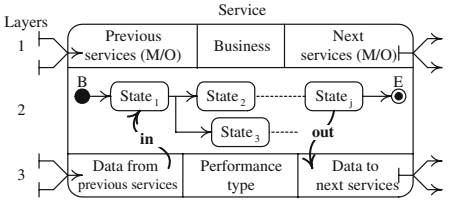


Fig. 1. Service chart diagram of a component service

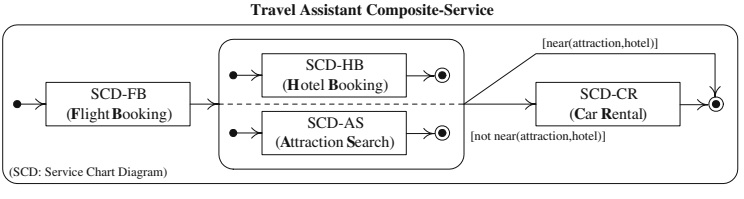


Fig. 2. Travel assistant composite-service as a state chart diagram

Personalization is the responsibility of users who indicate when and where they would like to have the component services performed according to specific periods of time and specific locations. Users may also indicate when and where they would like to see the outcome of this performance delivered. Because of the needs of personalization, two

extra perspectives denoted by location and time are anchored to a service chart diagram (Fig. 1). By location, it is meant specific places such as `classroom`, `mall`, and `home`. By time, it is meant specific moments such as `in the afternoon`, `after 2pm`, and `from 8am to 9am`.

3 Agentification of the Coordination of Personalized Composite Services

Panayiotou and Samaras state in [4] that personalization is complex since many aspects and issues need to be studied and addressed, respectively. Moreover, the authors state that such aspects and issues become more complex when considered in a wireless context. Among the issues reported in [4], we cite what content to present to user, how to show the content to user, how to ensure user's privacy, and how to create a global personalization scheme? While these issues are coupled to a Web-content provisioning perspective, we identified additional issues when personalization is coupled to a Web-service provisioning perspective. These issues are as follows: at what level can a Web service be personalized, does Web service personalization occur before or after composition (pre-composition *vs.* post-composition), to what extent can a user personalize a Web service without affecting its consistency, and does Web service personalization have to comply with certain policies?

3.1 Architecture

Fig. 3 illustrates the outcome of agentifying the coordination of personalized composite services. The agentification associates the components subject to coordination and personalization with multiple types of agents. These components are user, component service, and composite service. Correspondingly, three types of agents are deployed: *user-agent*, *service-agent*, and *composite-agent*.

The role of a designer in the coordination process is to instantiate the perspectives of a service chart diagram, excluding time and location perspectives. This goes back to users who instantiate them according to their needs and preferences. In this paper, it is assumed that all the necessary information to instantiate the perspectives of a service chart diagram are known in-advance for designers (by instantiation, it is meant assigning values to the parameters that constitute each perspective). Therefore, a designer knows the pre- and post-data that are exchanged between the component services, the businesses that provide the component services, and the possible strategies for invoking the component services. Once the specification of a composite service is completed, it is announced and offered to users as a *template*. A template is need-oriented (e.g., travel planning) and can be subject to adjustment. After receiving an announcement, users download the template in case they have an interest and adjust the template by instantiating time and location perspectives. For example, a user may indicate that all the component services related to car-registration composite-service have to be executed `after 9am` once he is `at work`.

To devise composite services, the connection of the component services through their respective service chart diagrams occurs at two levels (Fig. 3). The highest level

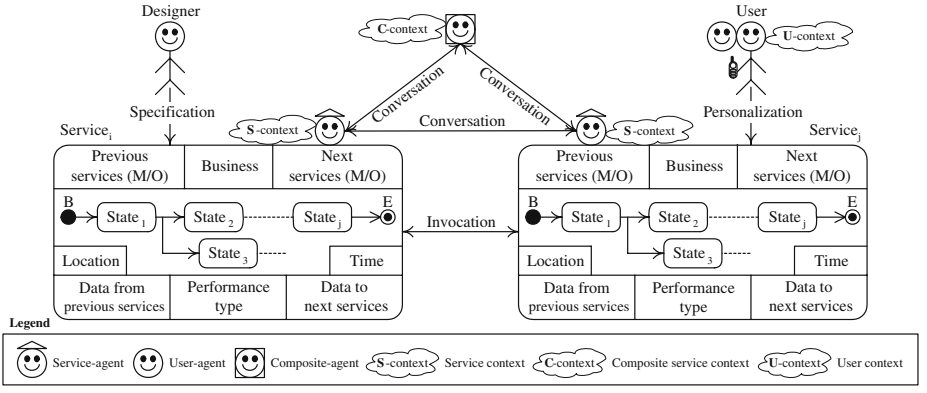


Fig. 3. Agent-based architecture for coordinating personalized composite-services

of connection, referred to as *conversation*, defines the messages that are exchanged between the service-agents of the component services. The messages are on various matters including (i) what is the preparation status of the next component services, (ii) where do the next component services need to be executed by finding out the current location *vs.* the location the user has planned to be, and (iii) when do the next component services need to be executed by finding out the current time *vs.* the time the user has indicated? Acting as a conversation controller, the composite-agent monitors all the conversations that service-agents engage. A composite-agent has to ensure that the right component services are selected, added, triggered, and executed with respect to the specification of the composite service (Fig. 2). To this purpose, the composite-agent takes various actions such as comparing the current execution location and current execution time of a component service to the specification content in terms of execution-location expected and execution-time expected. Analyzing the value-added of conversations to personalization is part of our future work.

The lowest level of connection, referred to as *invocation*, corresponds to the messages that implement the conversation level, including data transfer, service triggering, and request submission. Because of the technical obstacles that may face the deployment of conversations (e.g., network connection interrupted), actions for exception handling are needed. In case of the occurrence of any exception, a service-agent notifies first the composite-agent, and second makes its respective component service takes an appropriate state in which some corrective actions are executed. For instance, if a component service does not acknowledge a data reception, the sender component service has to enter a new state so that the lack of acknowledgement is handled.

Besides the conversations that take place between service-agents and composite-agents, the execution progress of a composite service is also traced because of the information the different agents manage on the components they act on their behalf (Fig. 3). This information is stored in a structure that we denote by context and specialize into $\mathcal{C}$-context (context of $\mathcal{C}$omposite service), $\mathcal{S}$-context (context of $\mathcal{S}$ervice), and $\mathcal{U}$-context (context of $\mathcal{U}$ser). Details on the internal structure of each context are given in Section 4. $\mathcal{U}$-context is fed by the data of two independent sources: user's mobile device for

time-related matters (i.e., current time) and user for location-related matters[2](i.e., user's current location). $\mathcal{S}$-context is fed by the data of $\mathcal{U}$-context and the perspectives of a service chart diagram. A service-agent continuously checks the $\mathcal{S}$-context so that it takes actions and engages in conversations with a composite-agent and other service-agents. Finally, $\mathcal{C}$-context is fed by the data of $\mathcal{S}$-contexts and $\mathcal{U}$-context. $\mathcal{C}$-context is continuously checked by a composite-agent so that it takes actions and engages in conversations with service-agents.

3.2 Operation

The operation of the agent-based architecture for coordinating personalized composite services constitutes of three stages: *specification*, *personalization*, and *deployment*. Designers of composite services are responsible for the specification stage. Future users of composite services are responsible for the personalization stage. Finally, software agents are responsible for the deployment stage. Fig. 4 is a summary of the major operations that occur in each stage.

Specification Stage. Initially, a designer works on the specification of a composite service by establishing the following: the list of component services that constitute a composite service, the businesses that provide the component services, the invocation strategies the component services support (i.e., remote, local, or both) [17], the data dependency between the component services, and last but not least the execution chronology of the component services. The designer is assisted in his specification work with service chart diagrams. In travel-assistant scenario, Fig. 2 has shown that the collaboration of at least four component services are required. These component services are connected according to a specific flow of control. First, flight reservation is completed before both hotel booking and attraction search are initiated. Indeed, a data dependency exists between both services and flight-reservation service at the level of flight arrival's date and time.

```
Specification stage:
  instantiate the perspectives except time/location perspectives.
  announce the composite service to users.
Personalization stage:
  express interest in a composite service.
  instantiate time and location perspectives.
  check the personalized composite service.
Deployment stage:
  monitor user-requested location and time.
  execute relevant component services.
  monitor next component services.
  update contexts.
```

Fig. 4. Summary of the major operations in the agent-based architecture

[2] Advanced satellite-based techniques for automatic user-positioning can be used [16].

Once the designer finishes his work, the specification of the composite service is stored at the level of a third party (e.g., a server). For brevity reasons, the third party is not discussed further. It is assumed that before storing the specification of a composite service, the third party checks its consistency (different techniques can be used for state charts checking, e.g., [18]). Following a successful check, the third party notifies (e.g., using SMS) users about the availability of the new composite service. A short description outlining the rationale, benefits, outcomes, and potential charges of the composite service always accompanies the notification message that users receive from the third party.

Personalization Stage. If a user has an interest in a composite service, he requests from the third party to deliver a *light* version of this composite service's specification, which is going to be stored in his mobile device upon reception. By light, it is meant the following components: list of component services (e.g., hotel booking, attraction search), their execution chronology, and their respective location- and time-related parameters. After receiving the specification, the user selects the component services that he wishes to adjust their location and time of execution parameters. It should be noted that personalizing composite services is *optional* as users might accept the designer's default execution-location and execution-time of services. The specification established by designers is sufficient to execute the composite services. Afterwards, the user returns his preferences included in the adjusted specification of the composite service to the third party. The duties of the third party consist now of (i) creating an instance from the user-selected composite service, (ii) integrating the user's preferences in this instance, and (iii) devising appropriate agents according to what was described in Section 3.1. Once completed, the third party performs a second round of consistency check since the designer's specification is no longer similar to what was initially stored. This check aims at avoiding conflicts during the deployment of the user-adjusted composite service. We rely for the consistency check on [8] for time-related parameters and [7] for location-related parameters.

In case the third party detects any inconsistency in the user-adjusted specification, the third party informs the user about it. The user is invited either to review his adjustments or to relax certain temporal or location constraints. This exchange between the third party and user keeps going until the specification of the composite service after adjustment is declared free of conflicts. The following is an example of conflict.

Designer specification	$Service_i$`:data` $\mapsto$ $Service_j$
User adjustment	`(`$\cdots$`,` $Service_i$`,` `?`$location$`,` $time_i$`),`
	`(`$\cdots$`,` $Service_j$`,` `?`$location$`,` $time_j$`),`
	`before(`$time_j$`,`$time_i$`)`

In this example, $Service_i$ will be executed before $Service_j$ due to the data dependency that is indicated in the designer's specification (`(`$Service_i$`:data` $\mapsto$ $Service_j$`)`. However, the user suggests a temporal order for these two services where the execution time of $Service_j$ (i.e., $time_j$) is *before* the execution time of $Service_i$ (i.e., $time_i$), which violates the specification. As a result, the temporal relationship `before(`$time_j$`,` $time_i$`)` needs to be relaxed.

Deployment Stage. After the consistency check is over, the deployment stage is triggered. The execution order of the component services depends on two factors: (i) execution chronology as defined by designers, and (ii) time and location values as defined by users. Besides the three factors, information that $\mathcal{C}$-context, $\mathcal{S}$-context, and $\mathcal{U}$-context contain is used. Indeed, the current location of a user is obtained from $\mathcal{U}$-context, and the current state of a service is obtained from $\mathcal{S}$-context. Last but not least, the list of component services that for example have completed their execution are obtained from $\mathcal{C}$-context.

Two major elements feature the coordination of the execution of the specification of a composite service, namely *one-step ahead* and *backward conversation* (Fig. 5). By one-step ahead, it is meant that while a set of component services are under execution, a monitoring of the specification of the next component services is concurrently performed. The monitoring checks when and where the next component services need to be executed and what data need to be received from the previous component services. In case the service-agent of a component service to be executed detects a potential delay in receiving data or that the appropriate location is different from the user's suggestion, the service-agent initiates a backward conversation with the appropriate predecessor service-agents. It should be noted that with whom a service-agent has to converse is available in the service chart diagram of the service that this service-agent represents. The aim of the backward conversation is to make aware the predecessor service-agents of the possibility of violating the specification of the composite service, which means the obligation of immediately taking corrective actions. A note about the backward conversation is also sent to the composite-agent for notification and context update.

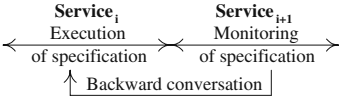


Fig. 5. Concurrent processing of execution and monitoring of services

Because time and location parameters are user-dependent, a tracking of both user's current location and current time needs to be performed. When the third party declares the consistency of the specification of a personalized composite service, the light version of the specification is kept stored in the user's mobile device. The user-agent, also residing in the user's device, regularly compares the time and location parameters as indicated in the specification with the current time and current location of the user. If there is a match, the user-agent wirelessly informs the composite-agent about the match. Once it gets notified, the composite-agent refers to the copy of the specification of the composite service it has so that the composite-agent starts executing actions such as sending invitations of participation to new component services or initiating the execution of certain component services.

4 $\mathcal{U}/\mathcal{S}/\mathcal{C}$-Contexts

In addition to the three types of agents that are put forward for the needs of agentifying the coordination of service personalization (Section 3.1), three types of contexts are used in this agentification. Each agent is attached to a component (i.e., user or component/composite service) and context as well. Details on $\mathcal{U}$-context contribute to the update of $\mathcal{S}$-context and $\mathcal{C}$-context, and details on $\mathcal{S}$-context contribute to the update of $\mathcal{C}$-context and the $\mathcal{S}$-contexts of other peers. By details, it is meant the values that are associated with the arguments of contexts. We use Tuple Spaces [19] for implementing the update operations between contexts. The use of tuple spaces is outside the scope of the paper[3].

The $\mathcal{S}$-context of a component service consists of the following parameters (Tab. 1): label, service-agent label, status, previous component services, next component services, regular actions, start-time requested, location requested, start-time effective, location effective[4], reasons of failure, corrective actions, and date. It should be noted that time-effective and location-effective parameters are execution-dependent (i.e., when and where the execution has effectively happened), whereas time-requested and location-requested parameters are user-dependent.

To ensure that the preference of a user has been considered during the deployment of the specification of a component service, the values of time-requested and location-requested parameters should respectively be equal to the values of time-effective and location-effective parameters. Any discrepancy between the parameters of type requested and the parameters of type effective indicates that the user's adjustment has not been properly handled.

The $\mathcal{C}$-context of a composite service is built upon the $\mathcal{S}$-contexts of its component services and consists of the following parameters (Tab. 2): label, composite-agent label, previous component services, current component services, next component services, status per component service, time, and date. It should be noted that *status per component* service parameter is service-dependent since the value of this parameter is obtained from *status* parameter of $\mathcal{S}$-context.

The $\mathcal{U}$-context of a user consists of the following parameters (Tab. 3): user-agent label, previous locations/component services, current location/component services, next locations/component services, previous periods of time/component services, current period of time/component services, next periods of time/component services, and date.

When a user personalizes a composite service, time-requested and location-requested parameters of the $\mathcal{S}$-context of some (or all) of the component services are instantiated (i.e., given values). Because the user-agent continuously monitors the *light* version of the specification of a composite service, which is stored in the user's mobile-device, the

[3] A sample of a control tuple illustrating an update operation between contexts: **modified($\mathcal{S}$-context s, WebService ws)**[true]|**update($\mathcal{C}$-context c, CompositeService cs)** means that if the $\mathcal{S}$-context **s** of the Web service **ws** is modified, the $\mathcal{C}$-context **c** of the composite service **cs** needs also to be updated prior to collecting information from this $\mathcal{S}$-context **s**.

[4] Parameters of type requested represent a user's preferences, whereas parameters of type effective represent what has really happened. For example, a user may *request* to start the execution of a service at 2pm, but due to some delay this execution has *effectively* started at 2:30pm.

Table 1. Description of $\mathcal{S}$-context parameters

Parameter & Description
Label: corresponds to the identifier of the component service.
Service-agent label: corresponds to the identifier of the service-agent of the component service.
Status: informs about the current status of the component service: in-progress, suspended, aborted, or terminated.
Previous component services: indicates whether there are component services before the component service (null if there are no predecessors).
Next component services: indicates whether there are component services after the component service (null if there are no successors).
Regular actions: illustrates the actions the component service normally performs; these actions are described in the states of the component service's state chart diagram.
Start time (requested and effective): informs when the execution of the component service should start (i.e., user-dependent) and has started (i.e., execution-dependent).
Location (requested and effective): informs where the execution of the component service should happen (i.e., user-dependent) and has happened (i.e., execution-dependent).
Reasons of failure: informs about the reasons that are behind the failure of the execution of the component service.
Corrective actions: illustrates the actions the component service has to perform in case the execution fails; these actions are described in the states of the component service's state chart diagram.
Date: identifies the time of updating the parameters above.

Table 2. Description of $\mathcal{C}$-context parameters

Parameter & Description
Label: corresponds to the identifier of the composite service.
Composite-agent label: corresponds to the identifier of the composite-agent of the composite service.
Previous component services: indicates which component services of the composite service have been executed with regard to the current component services (null if there are no predecessors).
Current component services: indicates which component services of the composite service are now under execution.
Next component services: indicates which component services of the composite service will be called for execution with regard to the current component services (null if there are no successors).
Status per component service: returns the status of each component service of the composite service that has been executed as well as under execution.
Time: informs when the execution of the composite service has started.
Date: identifies the time of updating the parameters above.

user-agent identifies the component services that are due for execution because of a certain location or a certain period of time. In case the user-agent identifies candidate component services for execution, it notifies the composite-agent, which proceeds with contacting the respective service-agents of these component services for the needs of

Table 3. Description of $\mathcal{U}$-context parameters

Parameter & Description
User-agent label: corresponds to the identifier of the user-agent of the user.
Previous locations/component services: keeps track of all the locations, as indicated by the user, that have seen the execution of component services (null if there are no previous locations).
Current location/component services: indicates the current location, as indicated by the user, that should see now the execution of component services.
Next locations/component services: indicates all the locations, as indicated by the user, that will see the execution of component services (null if there are no next locations).
Previous periods of time/component services: keeps track of all the periods of time, as indicated by the user, that have seen the execution of component services (null if there are no successor periods of time).
Current period of time/component services: indicates the current time, as indicated by the user, that should see now the execution of component services.
Next periods of time/component services: keeps track of all the periods of time, as indicated by the user, that will see the execution of component services (null if there are no next periods of time).
Date: identifies the time of updating the parameters above.

execution. If there are no obstacles preventing the execution of the component services, location-effective or time-effective parameters receive the value of their counter-part parameters namely location-requested and time-requested. This means that the user's preferences are being properly handled. Otherwise (i.e., existence of obstacles such as component service overloaded or lack of input data for a component service), the execution of the component services is postponed until these obstacles are dealt with. This means that location-effective or time-effective parameters will take values that differ from the values of their counter-part parameters namely location-requested and time-requested.

5 Conclusion and Future Work

In this paper, we presented our research in-progress for coordinating personalized composite services. By coordination, we meant the mechanisms that specify the orchestration of the component services of a composite service. By personalization, we meant the integration of users' preferences into the specification of the orchestration of the component services. Preferences concerned when and where the component services need to be executed. We have shown that this execution has been outsourced to software agents, which considered the respective contexts surrounding users and Web services. Three types of contexts have been suggested and referred to as $\mathcal{S}$-context, $\mathcal{C}$-context, and $\mathcal{U}$-context.

The personalization of composite services opens up the opportunity of conducting further research in the future. Besides having a proof-of-concept prototype to demonstrate the feasibility of our work, we are at this stage focusing on studying the value-added of conversations to personalization. Backing our work on the importance of integrating conversations into service composition, Ardissono et al. observed in [20] that current

Web services standards are used in systems featured by simple interactions. In addition, they noted that there are Web service-based situations where it would be important to express complex interactions by using conversations. Acting on behalf of services and running on top of their respective service chart diagrams, agents will engage in conversations with their peers when it comes to searching for the component services, checking the availability of the component services, triggering the component services, and coordinating the actions of the component services to avoid conflicts.

References

1. Curbera, F., Khalaf, R., Mukhi, N., Tai, S., Weerawarana, S.: The Next Step in Web Services. Communications of the ACM **46** (2003)
2. Berardi, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Mecella, M.: A Foundational Vision for e-Services. In: Proc. of The Workshop on Web Services, e-Business, and the Semantic Web (WES'2003) held in conjunction with The 15th Conference On Advanced Information Systems Engineering (CAiSE'2003), Klagenfurt/Velden, Austria (2003)
3. Maes, P.: Agents that Reduce Work and Information Overload. Communications of the ACM **37** (1994)
4. Panayiotou, C., Samaras, G.: mPERSONA: Personalized Portals for the Wireless User: An Agent Approach. Journal of ACM/Baltzer Mobile Networking and Applications, Special Issue on Mobile Commerce (forthcoming) (2003)
5. Brézillon, P.: Focusing on Context in Human-Centered Computing. IEEE Intelligent Systems **18** (2003)
6. Maamar, Z., Benatallah, B., Mansoor, W.: Service Chart Diagrams - Description & Application. In: Proc. of The Twelfth Int. World Wide Web Conference (WWW'2003), Budapest, Hungary (2003)
7. Papadis, D., Sellis, T.: On the Qualitative Representation of Spatial Knowledge in 2D Space. The Very Large Data Bases Journal, Springer Verlag **3** (1994)
8. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. Communications of the ACM **26** (1983)
9. Papazoglou, M., Georgakopoulos, D.: Introduction to the Special Issue on Service-Oriented Computing . Communications of the ACM **46** (2003)
10. Jennings, N., Sycara, K., Wooldridge, M.: A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems, Kluwer Academic Pub. **1** (1998)
11. Benatallah, B., Sheng, Q.Z., Dumas, M.: The Self-Serv Environment for Web Services Composition. IEEE Internet Computing **7** (2003)
12. Dey, A.K., Abowd, G.D., Salber, D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction Journal, Special Issue on Context-Aware Computing **16** (2001)
13. Schilit, B., Adams, N., Want, R.: Context-Aware Computing Applications. In: Proc. of The IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, USA (1994)
14. Maamar, Z.: Software Agents as a Support Technology for Collaboration - Application to the Humanitarian-Assistance Scenario. In: Proc. of The Int. ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce (MAMA'2000), Wollongong, Australia (2000)
15. Benatallah, B., Dumas, M., Sheng, Q.Z., Ngu, A.: Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In: Proc. of ICDE'02, IEEE Computer Society, San Jose (2002) 297–308

16. Ratsimor, O., Korolev, V., Joshi, A., Finin, T.: Agents2Go: An Infrastructure for Location-Dependent Service Discovery in The Mobile Electronic Commerce Environment. In: Proc. of The 1st ACM Int. Workshop on Mobile Commerce (WMC'2001) held in conjunction with The Seventh Annual Int. Conference on Mobile Computing and Networking(MobiCom'2001), Rome, Italy (2001)
17. Maamar, Z., Labbé, P.: Moving vs. Inviting Software Agents, What is the Best Strategy to Adopt? Communications of the ACM **46** (2003)
18. Varro, D.: A Formal Semantics of UML Statecharts by Model Transition Systems. In: Proc. of The 1st Int. Conference on Graph Transformation (ICGT'2002), Barcelona, Spain (2002)
19. Ahuja, S., Carriero, N., Gelernter, D.: Linda and Friends. Computer **19** (1986)
20. Ardissono, L., Goy, A., Petrone, G.: Enabling Conversations with Web Services. In: Proc. of The Second Int. Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS'2003), Melbourne, Australia (2003)

Workflow Management in Mobile Environments

Andrea Maurino and Stefano Modafferi

Politecnico di Milano,
Dipartimento di Elettronica e Informazione,
Piazza Leonardo da Vinci, 20133 Milano, Italy
{maurino, modafferi}@elet.polimi.it

Abstract. The young mobile technologies, i.e Bluetooth or Wi-Fi, suffer of limitations and problems especially if the net is composed by many nodes. This trouble aspect affect also the design of Mobile Information System based on wireless networks. Our research focuses on the management of a business process in a mobile environment. This paper is about an approach to divide an unique workflow in different autonomous workflows each one controlled by a different actor. This fact allows the actors independence and the net partitioning limiting the number of hosts working in each subnet. The presented delegation model supports disconnected characteristics and the independent execution of the workflow parts that means the hosting of an independent engine in each controller. The mobile scenario and the necessity of the more automation lead us to choose BPEL4WS as language for the process definition.

1 Introduction

New wireless technologies, i.e Bluetooth [3] or Wi-Fi [21], allow the connection of a number of devices without the need of fixed infrastructure, by creating the technological backbone of Mobile Information Systems (MobIS). Wireless technology is very young and it still presents some important limitations; for example the Bluetooth's piconet limits the connected nodes to seven; moreover the mobile networks suffer of several problems such as roaming, frequent disconnections and security [5, 19]. All these problems affect, with different degree, the modelling and the execution management of business process in a mobile environment. The execution of a mobile business process (such the one described in [12]) on complex MobIS environment, composed by a relative high number of devices connected by using a number of different network technologies, needs new strategies with respect to the traditional centralized workflow solutions where an engine knows and can control all system resources. Our research faces the problem of the execution of business process in complex MobIS by considering that such systems are inter ad-hoc networks connecting several single-technology mobile networks (ST-MobIS). In this paper we present a technique to create a set of sub-processes from an unique process. Each sub-process will be executed over an ST-MobIS, but all these ones will cooperate to complete the business process. The technique is based on two different phases: a set of delegation rules automatically define where the process can be divided by means of insertion of special control tasks in the process; in the second phase, the process is divided in local process views, which are assigned then to simple ST-MobIS.

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 83–95, 2004.

The language we choose to describe the business process is BPEL4WS [17] that is a business process description language strongly oriented to the automation, in particular automatic invocation of a service, that is often a very important feature requested by the user; so in this paper we assume that the business process is initially described in BPEL4WS.

The paper is organized as follow: Section 2 presents existing model for decentralizing workflow, Section 3 shows how to describe BPEL4WS in a graphical modality such as UML; Section 4 concerns the delegation phases of the entire delegation process; finally in the Section 5 we draw our conclusions and present the future work.

2 The Decentralized Workflow Models

The opportunity given by workflow distribution is well studied from about ten years in the business process design research field. This research is mainly looking for cooperation/integration between different firms that is different workflows. This problem reappears even in the MobIS domain as described in the introduction.

In [9] authors present a comparison between the different approaches to the distribution, and the Table 1 shows the main differences. Cross-Flow [7] aims at providing high-level support for workflows in dynamically formed virtual organizations. High-level support is obtained by abstracting services and offering advanced cooperation support. Virtual organizations are dynamically formed by contract-based matchmaking between service providers and consumers. In Agent Enhanced Workflow [10] the agent oriented solution presents interesting aspects, above all the possibility of building execution plans using a goal approaches. Event-based Workflow Process Management [4] includes an event-based workflow infrastructure and model constructs for addressing the time aspects of process management. The main feature of ADEPT [15] is the possibility of modifying the workflow instance at run-time. MENTOR [14] provides for an autonomous workflow engine for each actor by using owner business process language. The METEOR (Managing End to End OpeRations) [2] system leverages Java, CORBA, and Web technologies to provide support for the development of enterprise applications that require work ow management and application integration. It enables the development of complex work on applications which involve legacy information systems that have geographically distributed and heterogeneous hardware/software environments, spanning multiple organizations. This tool also provides support for dynamic work ows processes, error and exception handling, recovery, and QoS management. Exotica [13] is characterized by the possibility of disconnected operations; it does not allow the complete decentralization because it maintains a central unit and all the operations have a client/server paradigm. WISE [1] provides for its engine the web and offers an embedded fault handler. WAWM [16] focuses its attention on the problems related to the wide area network workflow management. XRL/Flowers [20] is a Wfms based on XRL (eXchangeable Routing Language) language; this is an instance-based workflow language that uses XML for the representation of process definitions and Petri nets for its semantics. XRL/flower benefits from the fact that it is based on both XML and Petri nets. Standard XML tools can be deployed to parse, check, and handle XRL documents. The Petri net representation allows for a straightforward and succinct implementation of the

Table 1. Comparison of approaches for distribution in workflow management

System	Focus	Application distribution relationship
CrossFlow	Cross Organizational Workflow by contracts	– The application is distributed over several Wfms – Distribution is specified in the workflow schemes and in contracts
Agent enhanced Workflow	Cross Organizational Workflow	– Workflow process is distributed over participating wfms – Execution plans are calculated by the agents based on goal definitions – Distribution is described by execution plans
Event-based workflow Process Management	Distribution of process, multiple times zones, organizational boundaries and legal domains	– Relationships to model components of the workflow management application are not mentioned
ADEPT	Enterprise wide workflows, cross enterprise workflow	– Server assignments define assignment of activities to wf servers – Assignments are specified in wf schemas (language extension ADEPT)
Mentor	Scalable, enterprise wide workflow management	– Centralized workflow specification – Partitioning by orthogonalization rules – Assignment by rules to users and engines – Rules are defined outside the workflow specification
METEOR	Enterprise applications and integrations	– No specification about distribution in the workflow
Exotica/FMDC	Disconnected clients, mobile computing	– No specification about distribution in the workflow schema needed
Exotica/FMQM	Resilience, scalability, flexibility	– Process is partitioned and allocated to engines reflecting closeness to users and workload – Allocation is based on users associated with different nodes – Allocation is described outside a workflow schema
WISE	Virtual enterprises and business process	– Distribution is described in separated process model
WAWM	Inter/intra organizational workflow	– Workflow application is distributed over autonomous Wfms – Distribution is modelled in each workflow schema or in a separated virtual process definition
XRL/Flower	Inter/intra organizational workflow	– Wfms is based on XRL language and uses petri nets formalism.
Mobile	Enterprise wide, cross organizational workflow	– Original $Wfms$ is partitioned and allocated to different WFMSs – Distribution is modelled in the workflow schema – Technical and organizational impacts are considered

workflow engine. Mobile [8] is developed to support inter-organizational and is strongly based on the modularity; this model characteristic alleviates change management and also allows to customize and extend aspects individually.

By analyzing Table 1 it is possible to characterize two different and dual approaches to the problem of workflow coordination. The former provides for the integration of different autonomous preexisting workflows and its principal aim is the coordination between different and independent actors. The latter provides for the decomposition of an originally unified workflow to allow its autonomous execution. Cross-Flow, Agent Enhanced Workflow, Event-based Workflow process Management, Adept, Meteor, WISE, WAWM and XRL/Flowers belong to the first approach, Mentor, Exotica and Mobile belong to the second one.

The described system offer three different solutions for the partitioning and allocation rules definition. The first one defines a specific definition language (Cross-Flow, Agent enhanced workflow, mentor, Exotica). The second approach consists in the extension of workflow language with distribution rules (Cross-Flow, ADEPT, WISE, WAWM, Mobile). The third one does not consider the language of distribution rules (Event-based, workflow Process Management, Meteor, XRL/Flowers). Cross-Flow belongs to more than one class because the distribution rules are split into several definitions parts.

Our delegation model supports disconnected characteristics such as Exotica, the independence of workflow engine such as MENTOR and the possibility of modifying

the workflow instance at run-time such as ADEPT. Moreover we argue that the mobile environment needs a language strongly oriented to the automatic execution such as BPEL4WS. Another important aspect is the necessity of lightness that is a requested feature if the system run on portable devices in ad-hoc network. As far as the rule definition is concerned our system defines the partitioning rules but does not define the allocation one demanding them to the specific business process and application domain.

3 Describing BPEL4WS with UML

The business process Execution Language for Web Services (BPEL4WS) [17] provides an XML notation and semantics for specifying business process behavior based on Web Services. A BPEL4WS process is defined in terms of its interactions with partners. A partner may provide and/or require services to/from the process or participate in a two-way interaction with the process. The Unified Modelling Language (UML) is a language, with a visual notation, for modeling software systems. The UML is an OMG standard and is widely supported by tools, it can be customized for the use in a particular modelling context through a "UML profile". We use an extended version of the UML Profile for Automated business processes described in [6]. In this way a BPEL4WS description can be described by an execution graph which can be manipulated in order to create several execution graphs, which cooperate to obtain the same goal.

3.1 Mapping Basic Activity

The basic activities represented in the BPEL4WS specification can be associated to a set of stereotypes of activity state. Table 2 shows the mapping.

Table 2. UML description of basic activities

invoke	$<< invoke >>$
receive	$<< recevive >>$
reply	$<< reply >>$
assign	$<< assign >>$
wait	$<< wait >>$
empty	$<< empty >>$
throw	$<< throw >>$
terminate	$<< terminate >>$

3.2 Special Activities

Special activities are defined for delegating business process control. Although the syntax is the same of *Invoke* and *Receive* activities the semantic difference and its importance leads us to define them in terms of stereotype.

These special activities are:

– **Do Start**: syntactically it is an invoke operation on an actor that gives the control of the flow to this actor;

Table 3. UML description of special activities

Do Start	$<< DoStart >>$
Receive Start	$<< ReceiviveStart >>$
Remote Assign	$<< ReceiveAssign >>$
Receive Assign	$<< ReceiveAssign >>$
Assign Evaluation	$<< AssignEvaluation >>$
Reply End	$<< ReplyEnd >>$

- **Receive Start**: syntactically it is a receive operation that takes the control of the flow given by another actor;
- **Remote Assign**: syntactically it is an invoke operation that allows the transmission of parameters needed for the evaluation of a condition;
- **Receive Assign**: syntactically it is a receive operation that allows the actor to wait for the transmission of parameters needed for the evaluation of a condition;
- **Reply End**: syntactically it is a reply operation that returns the control of the flow to the actor that before passed it.

These special activities can be associated to a set of stereotypes of activity state shown in Table 3.

3.3 Mapping Structured Activities

Structured activities prescribe the order in which a collection of activities take place. They describe how a business process is created by composing the basic activities it performs into a structure that express the control patterns, data flow, handling of faults and external events, and coordination of message exchanges between process instances involved in a business protocol. BPEL4WS provides three kinds of control structure; they are:

- ordinary sequence: *sequence*, *switch*, *while*
- concurrency activity: *flow*
- nondeterministic choice: *pick*

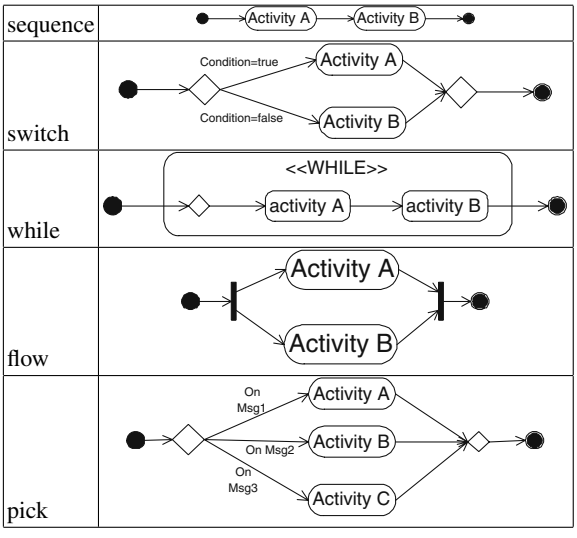
Table 4 shows the UML description of such structures. It is worth noting that we do not consider the link.

3.4 Data in Decentralized Environment

BPEL4WS provides for global variables in its processes. This solution, with lots of advantages, is not allowed in a totally distributed environment where independent actor works.

The model language allows the use of messages for the communication among different actors; so to solve this problem we are planning to transform global variables in part of the messages exchanged. The behavior of each actor will be modelled according to this specification.

Table 4. UML description of structured activities



4 Delegate Business Process Control

As described before, we are interested in considering the control of a portion of a workflow rather than the possibility of remote invocation of a service that is already implemented in BPEL4WS.

The control flow delegation needs the creation of BPEL4WS subschema that will be executed by controller of ST-Mobis. This task is executed by defining the section of the global BPEL4WS description that has to be controlled by a given actor. We introduce the notion of ***local process view*** representing the view of the whole process by the point of view of an actor involved in the control of execution process.

4.1 Identification of Delegation of Control

In order to allow disconnected operations we define some rules to ensure the correctness and the conclusion of the workflow. In particular we define the following rules:

- every time an activity (both basic or structured) controlled by an actor A_1 is followed by another one controlled by an actor A_2 we introduce new task for the control of the delegation process;
- it is necessary to establish a controller of each structured activity (with the exception of *sequence*). So it has a *Start Controller* and an *End Controller*. In the While construct the controls have to be executed by the same actor;
- the execution of control flow of a specific set of tasks can be assigned to only one device;
- in the business process does not exist global variables and all the variables have to be passed as parameters between different actors;

For each activity, a figure helps us for the comprehension of delegation process. In the following figures the terms A_i indicate the actor associated with the control of that task (set of tasks); the terms $<< Make >> Task_n$ indicate the (set of) basic operation(s) (see Table 2) needed to perform a (set of) task(s) of original process. The *Start Controller*, (resp. *End Controller*) is indicated for each structured activity start (resp. end).

In the next paragraphs we describe how to delegate the process of each structured activity showed in Table 4.

Sequence. This is the fundamental brick and each other structured activities will use a set of opportunely structured sequences.

If two activities A and B are in a sequence, we add between them an *activity DoStart* executed in the actor A_1 and controlled by A_1 itself and a *activity Receive Start* executed in the actor A_2 and controlled by A_2.

Flow. The Flow construct represents the parallel execution of different tasks. The delegation process considers each branch such as a sequence traducing them in the same way. The only difference is at the end of the sequence where the delegation model provides for a reply action able to communicate to the *End Controller* the conclusion of the assigned tasks. Fig. 1.a shows an example of such case.

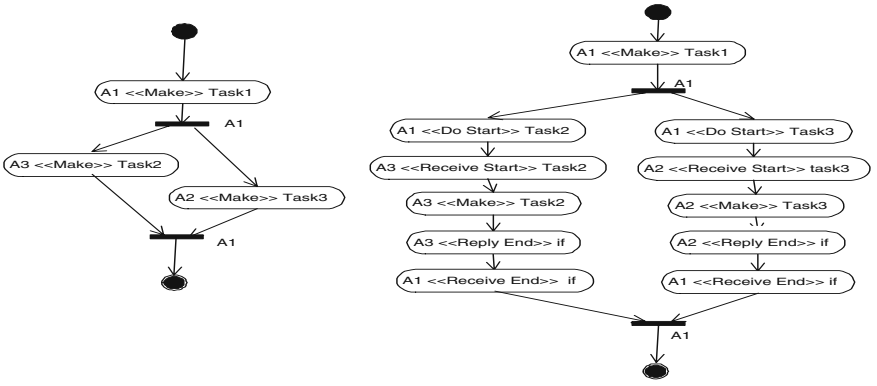
Switch. The switch construct models the possibility of using different path according to the value of a specific variable. In this case it is necessary to propagate to all the actors potentially involved in the various branch the value of the control variable. This operation is carried out by using for each branch a flow with the *activity Remote Assign* assigned to switch controller and the *activity Receive Assign* assigned to all task controllers (in case to the different task controllers of that branch).

After this operation the appropriate delegation rules concluded by the final operation activity $<< ReplyEnd >>$ is applied to each switch branch for returning the control to the *End Controller*. The Fig. 1.b shows the delegation process.

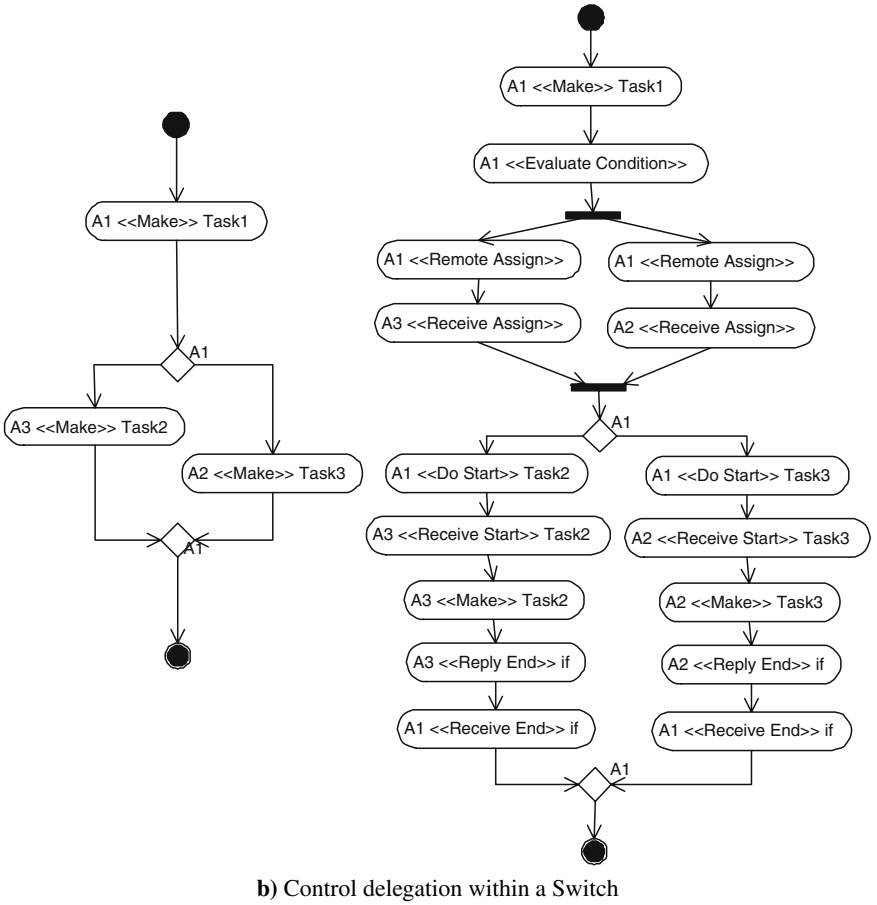
Pick. The Pick activity awaits the occurence of one of a set of events and then performs the activity associated with the event that occurred.

The problem of control delegation for the Pick activities is solved with a solution very similar to the one used for the Switch (see Fig. 2). This is possible because both the switch and the pick models a conditional operation driven by a variable. The difference variable nature affects our model only for the elimination of a receive activity in the delegated actor.

While. The while construct models a loop. This powerful possibility arises some problems in its management; in fact the number of iterations is not a-priori known and the loop control variables can be updated from any actor involved in the loop tasks. Thus for each cycle the evaluation (the cycle has to be made or not) made by the loop controller has to be propagate to all the actors. The last operations of the loop controller are a new evaluation of the decision variables (in case modified) and their communication to the other actors to allow them the exit from the loop. The Fig. 3 shows this process.



a) Control delegation within a Flow



b) Control delegation within a Switch

Fig. 1. Delegation of control within a Flow and a Switch

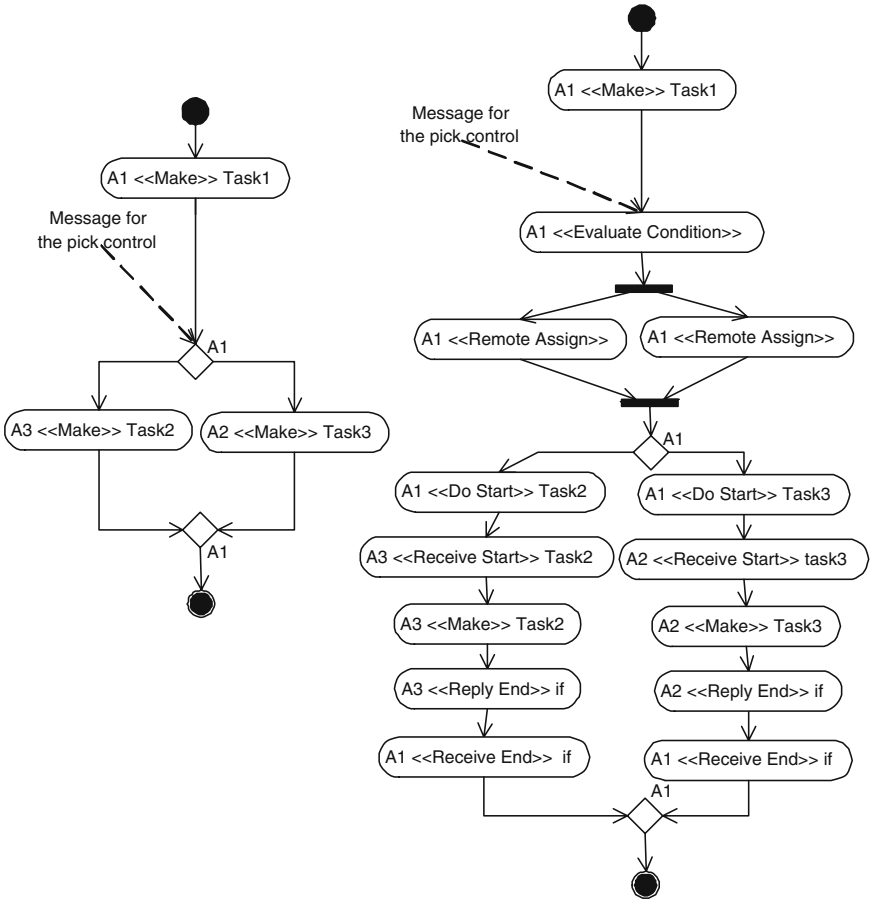


Fig. 2. Delegation of control within a Pick

4.2 Local Process View

The local process view for an actor A, includes the part of the process that has to be controlled by A. More details about local views in a Business Process can be found in [18].

In our approach the construction of local view is the second and final step of business process delegation. The first step inserts in the workflow the opportune special tasks able to manage the process delegation. The second step is the building of the local view, obtained by applying to the general process the following simplification rules:

- All activities, whose execution is not controlled by the actor are removed
- All structured activities, with the exclusion of *sequence*, which do not include activities, are translated into a direct arc
- Within *pick* or *switch* all branches not including activities are substituted into direct arc;

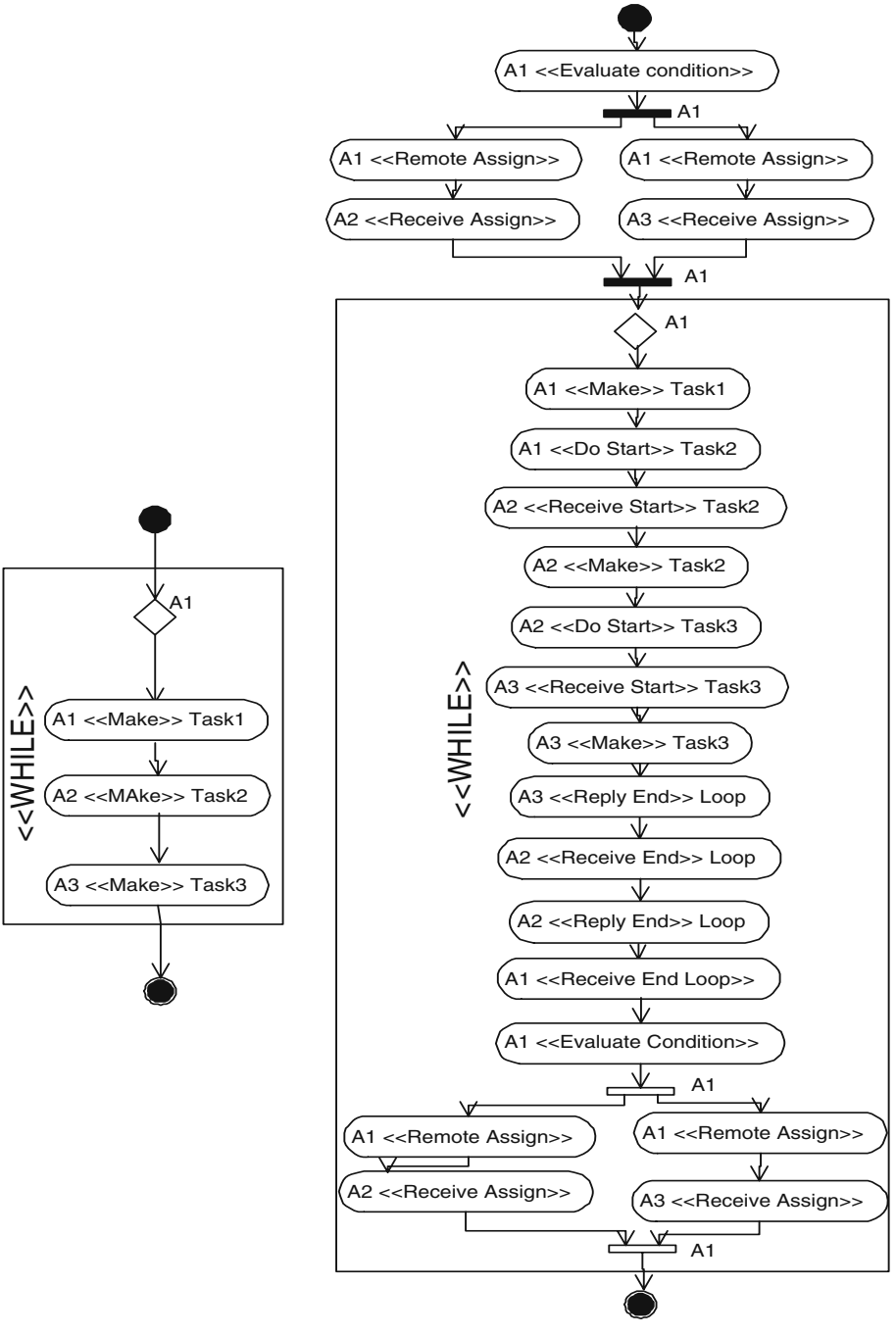


Fig. 3. Delegation of control within a While

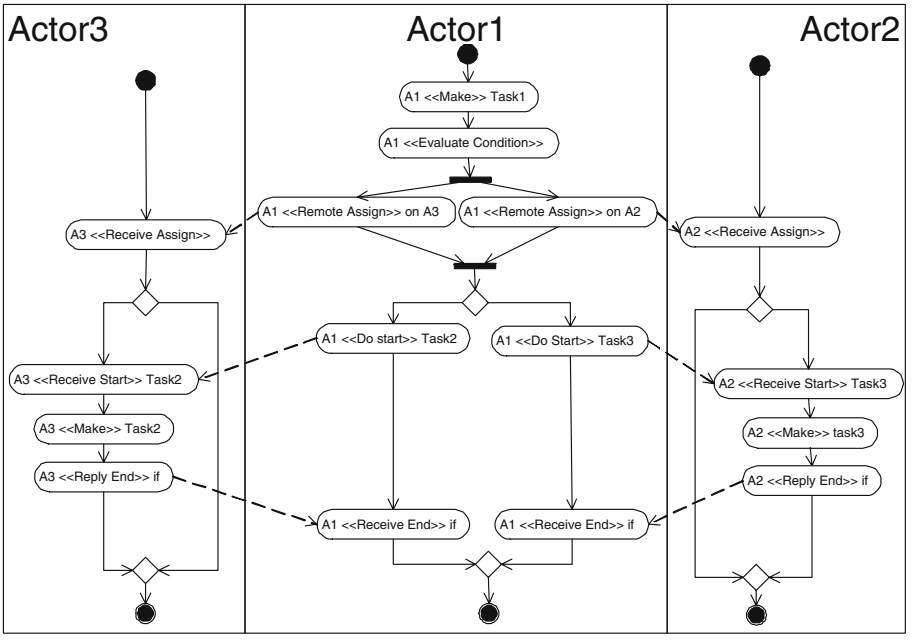


Fig. 4. The three local views after the delegation of control within a Switch

The Fig. 4 shows the local view of three actors involved in the delegation process within a switch control before presented in Fig. 1.b. Now each actor has its own business process and it can work in an independent way.

5 Conclusion

In this paper we have presented a new methodology for decentralizing the control flow of a business process. Our study considers the problems of decomposition of an originally unified workflow rather than the composition of different business process. We have provided the needed rules to decompose each primitive and structured activity for building an actor local view that allows an autonomous workflow execution maintaining the original properties of the process.

Our model provides for the possibility of autonomous execution of the different workflow parts besides in off-line mode as the case may be useful. This is a necessity if the business process is deployed in a mobile environment where the ad-hoc networks do not allow the participation of too many actor at the same time. Moreover our rules could be applied at design time but also at run-time supporting the dynamic workflow modification according to the context.

Although other models following the decentralized approach offer some of this feature, the choice of BPEL4WS allows us a more efficient use of automatic invocation and deployment of the process. This is important if the scenario is adaptive and multichannel and one of the issues is the limitation of the human/system interaction.

We are now studying the problem related to the data modelling, especially the translation of global variables into messages parameters. Moreover BPEL4WS offers a powerful event/fault handler but it has a centralized way of working and the correct delegation among different and independent actors of this specific section is still an open issue and we are investigating it. The next step will be the study of choreography model able to take all the possible advantages of the presented one.

Acknowledgments. This work has been developed within the Italian MURST-FIRB Project MAIS (Multi-channel Adaptive Information Systems) [11].

References

1. G. Alonso, U. Fiedler, C. Hagen, A. Lazcano, H. Schuldt, and N. Weiler, *WISE: Business to business e-commerce*, RIDE, 1999, pp. 132–139.
2. K. Anyanwu, A. Sheth, J. Cardoso, J. Miller, and K. Kochut, *Healthcare enterprise process development and integration*, Journal of Research and Practice in Information Technology **35:2** (2003).
3. Bluetooth consortium, *Bluetooth project*, 1999, `www.bluetooth.org`.
4. J. Eder and E. Panagos, *Towards distributed workflow process management*, In proc. of Workshop on cross-Organizational Workflow Management and Coordination (San Francisco, USA), 1999.
5. G. Gaertner and V. Cahill, *Understanding link quality in 802.11 mobile ad hoc networks*, Internet Computing **8:1** (2004), 55 – 60.
6. T. Gardner and al., *Draft uml 1.4 profile for automated business processes with a mapping to the bpel 1.0*, IBM alphaWorks, 2003.
7. P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig, *Crossflow: Cross-organizational workflow management in dynamic virtual enterprises*, International Journal of Computer Systems Science & Engineering **15** (2000), no. 5, 277–290.
8. S. Jablonski and C. Bussler, *Workflow management: Modeling concepts, architecture and implementation*, International Thomson, 1996.
9. S. Jablonski, R. Schamburger, C. Hahn, S. Horn, R. Lay, J. Neeb, and M. Schlundt, *A comprehensive investigation of distribution in the context of workflow management*, In proc. of International Conference on Parallel and Distributed Systems ICPADS (Kyongju City, Korea), 2001.
10. D. Judge, B. Odgers, J. Shepherdson, and Z. Cui, *Agent enhanced workflow*, BT Technical Journal (1998), no. 16.
11. MAIS_Consortium, *MAIS: Multichannel Adaptive Information Systems*, http://mais-project.it.
12. A. Maurino and S. Modafferi, *Challenges in the designing of cooperative mobile information systems for the risk map of italian cultural heritage*, Proc. of WISE Workshop on Multichannel and Mobile Information Systems, IEEE press, 2003.
13. C. Mohan, G. Alonso, R. Gunthor, and M. Kamath, *Exotica: A research perspective of workflow management systems*, Data Engineering Bulletin **18** (1995), no. 1, 19–26.
14. P. Muth, D. Wodtke, J. Weisenfels, A. Kotz Dittrich, and G. Weikum, *From centralized workflow specification to distributed workflow execution*, Journal of Intelligent Information Systems **10** (1998), no. 2, 159–184.

15. M. Reichert and P. Dadam, *Adeptflex – supporting dynamic changes of workflows without losing control*, Journal of Intelligent Information Systems **10** (1998), no. 2, 93–129.
16. G. Riempp, *Wide area workflow management*, Springer, London, UK, 1998.
17. S. Thatte, *Business process execution language for web services*, 2003, `www-106.ibm.com/developerworks/webservices/library/ws-bpel/`.
18. W. Van Der Aalst and M. Weske, *The p2p approach to interorganizational workflows*, Proc. of Conference on Advanced Information Systems Engineering CAiSE (Interlaken, Switzerland), 2001, pp. 140–156.
19. S.J. Vaughan-Nichols, *The challenge of wi-fi roaming computer*, IEEE Computer **36:7** (2003), 17–19.
20. H.M.W. Verbeek, A. Hirnschall, and W.M.P. van der Aalst, *XRL/Flower: Supporting interorganizational workflows using xrl/petri-net technology*, In proc. of Web Services, E-Business, and the Semantic Web, International Workshop (WES) (Berlin (Germany)), vol. 2512, Lecture Notes in Computer Science, Springer-Verlag, 2002, pp. 93–108.
21. Wi-Fi consortium, *Wi-Fi project*, 1999, `www.wi-fi.org`.

DIWE: A Framework for Constructing Device-Independent Web Applications

Engin Kirda and Clemens Kerer

Technical University of Vienna, Distributed Systems Group,
Argentinierstrasse 8/184-1 A-1040 Wien, Austria
{E.Kirda, C.Kerer}@infosys.tuwien.ac.at

Abstract. Recent developments in mobile computing software and hardware have highlighted the importance of *device-independent* access to Web content. This paper introduces a novel conceptual framework for constructing device-independent Web applications. The Device-Independent Web Engineering (DIWE) framework is composed of an XML-based Web language that is used to separate the layout, content and application logic and to model the Web applications and four run-time processors that provide device-independence support during application execution.

1 Introduction

Until the late 90s, the main focus of Web engineering research was the development of tools, technologies and methodologies for the design, implementation and maintenance of HTML-based Web applications. The common assumption was that a Web application would always be accessed by a browser found on a personal computer or a laptop. Recent developments in mobile computing software and hardware not only have changed this view, but have also increased the importance of *device-independent* access to Web content: The ability to access Web content using a wide variety of *Web devices*. A Web device is any hardware or software that can be used to access Web content [1] such as telephones equipped with speech recognition software, digital televisions and Personal Digital Assistants (PDAs).

This paper introduces a novel conceptual framework for device-independent Web engineering. The Device-Independent Web Engineering (DIWE) framework is composed of an XML-based Web language that is used to separate the layout, content and application logic and to model the Web applications, a visual Integrated Development Environment (IDE), and four default run-time processors that provide device-independence support during application execution.

The framework uses two techniques, *page splitting* and *process partitioning*, that allow the Web developer to tune the selected information and the sizes of generated pages according to the characteristics of a device that is being targeted. These techniques attack the problem of displaying Web pages on devices with small displays and limited memory capacity. The adaptation of content for a Web device is performed during the design and implementation stages of Web engineering.

The framework also introduces a novel technique called *XSL stylesheet pre-processing* that allows the reuse of *existing* XSL stylesheets when adding new device support to a

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 96–110, 2004.

Web application. Stylesheet pre-processing reduces the maintenance overhead because redundancies are avoided.

The paper is structured as follows: The next section gives an overview of the DIWE framework and discusses the main concepts. Section 3 describes how DIWE is used to create device-independent Web applications. Section 4 briefly presents the Vienna International Festival case study where DIWE was used to create a device-independent e-commerce Web application. Section 5 presents related work and Section 6 concludes the paper.

2 Overview of the DIWE Framework

There are four important requirements that a device-independent Web engineering framework should meet: It should be *platform and implementation language-independent*, it should support the *definition and generation of content and layout in XML* for non-HTML Web devices, it should use *industrial standards* to enable the use of existing tools and most importantly, it *should not increase the maintenance effort significantly*. The design of the DIWE framework was guided by these requirements.

The DIWE framework consists of the *MyXML language* (a Web application modeling language), a compiler that can process the language, a visual Integrated Development Environment (IDE), *MyXMLDesigner*, and four run-time *processors* that are configured and deployed on the Web server at run-time to provide device-independence support.

Figure 1 illustrates the usage of the framework in the design, implementation, deployment and maintenance stages of Web applications.

A device-independent Web application in the DIWE framework is a Web application that can be *extended* to support different Web devices of widely varying technical capabilities. During the design, so called *device families* need to be identified by the developers that the application will support. A device family is made up of a collection of Web devices that have similar characteristics. PDAs with high memory capacity and a display size larger than 200x300 pixel size, for example, could make up a device family for a particular service. Another example of a device family is the collection of WAP-enabled phones and PDAs. Each Web application will at least support one device family during its life time: the *default* device family. For example, in a cultural Web site we built, our decision was to support a full HTML browser interface for the entire site as the default device family and a WAP-based mobile phone interface for the ticketing service only.

A *device-independent* Web application in DIWE is modeled with the MyXML language. The language supports the complete Layout/Content/Logic (LCL) separation and is used by the Web developers to design and define the content and the interfaces to the application logic. A MyXML language compiler then integrates the layout and generates static content embedded in HTML or XML, or source code that provides interactive functionality.

The MyXML language used for application specification is a simple XML-based language that uses loops, variables and database access functions. One of main advantages of an XML-based Web specification language is that it allows the definition of functionality that is technology independent. Any popular programming or scripting

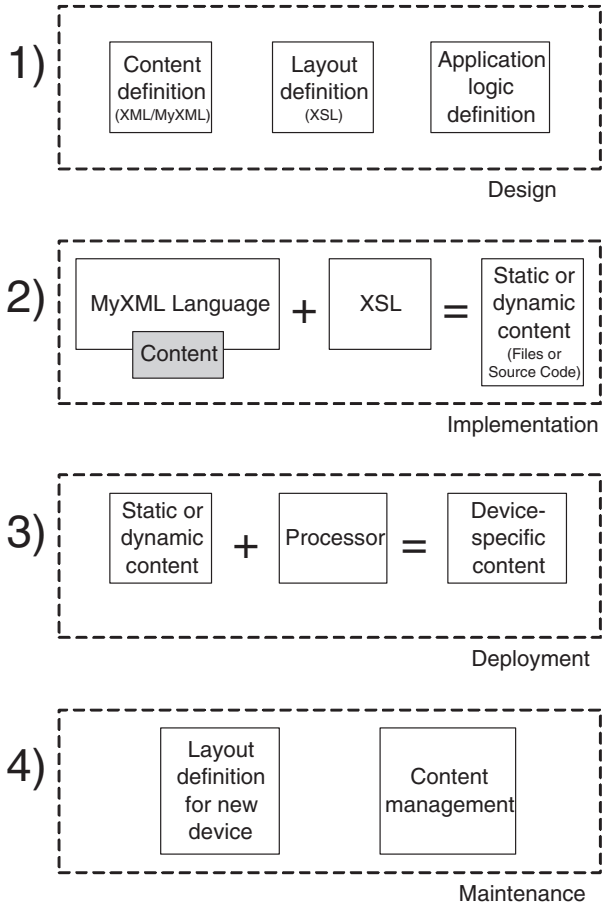


Fig. 1. Web application design, implementation, deployment and maintenance

language can be generated from the XML and XSL specifications with an appropriate MyXML language compiler. A complete description of the MyXML language is beyond the scope of this paper. The reader is referred to [2, 3] for more information.

After a Web application has been generated from the XML/XSL specifications and the application logic has been written and integrated, the run-time processors in the framework are used to filter and adapt the output stream produced by the application to suit a particular Web device.

2.1 Run-Time Processors

The four default run-time processors in the DIWE framework are: The *device detection*, *logic interfacing*, *page splitting* and *process partitioning* processors. The Web developer can optionally build and deploy application-specific processors that can be used to process and adapt the run-time output stream produced by the constructed Web applications

(e.g., to generate PDF receipts for an e-commerce order by using the information in a Web page).

Device Detection. The *device detection* processor is responsible for device detection and identification. It can be configured to detect the device (i.e., the device family) a user is using based on the HTTP request header.

Logic Interfacing. The *logic interfacing* processor provides application logic integration and invocation support to the applications modeled by the MyXML language. It allows the application logic to be written once and used for different layouts without the need for any modifications. The traditional approach to supporting different layouts with the same application logic is to build conditional (e.g., *if/then/else*) statements into the code and to present the layout based on some criteria (e.g., the user chooses a device name from a list). For example, the following pseudo application logic code checks the value of a variable named *device* and presents the appropriate HTML or WAP layout:

```
  ... do some domain specific task ...
if (device="html") present HTML_LAYOUT
else if (device="wap") present WAP_LAYOUT
  ... do some domain specific task ...
```

The disadvantage of this approach is that the application logic has to be modified and extended for every new device that is being supported. Using this approach, while writing the application code, the Web developer often needs to know in advance what type of devices will be supported. She has to try to design and optimize the code so that it can easily be extended: A task that is not always easy to achieve.

The logic interfacing processor allows the application logic to be reused for arbitrary devices by acting as a wrapper to the generated device-specific Web applications. Depending on the device (i.e., the identification of the device family) that is requesting the Web page, the logic interfacing processor invokes the appropriate, pre-defined layout and content that is embedded in a Web application.

Page Splitting. The *page splitting* processor provides *layout adaptation* support. Layout adaptation in Web application construction deals with the problem of displaying pages on device families with small display and limited memory capabilities.

The *page splitting* technique deals with the page size problem by using a combination of special tags that are encoded into the XSL stylesheets that are interpreted by the page splitting processor at run-time. The main idea behind page splitting in Web application construction is to split and partition the content in XSL files by *grouping* layout elements (e.g., the header information in one group, the footer information in a second group, the navigation information in another and so on). A *group* identifies a logical unit of information on the page that a device family is able to display. Groups can also be partitioned and split using *subgroups* and thus, different splitting granularities can be achieved.

Figure 2 illustrates the concept of grouping and subgrouping on a commercial Web page (belonging to the Vienna International Festival Web site case study, `http://www.festwochen.at`) that displays a list of cultural events (i.e., exhibitions, ensembles,

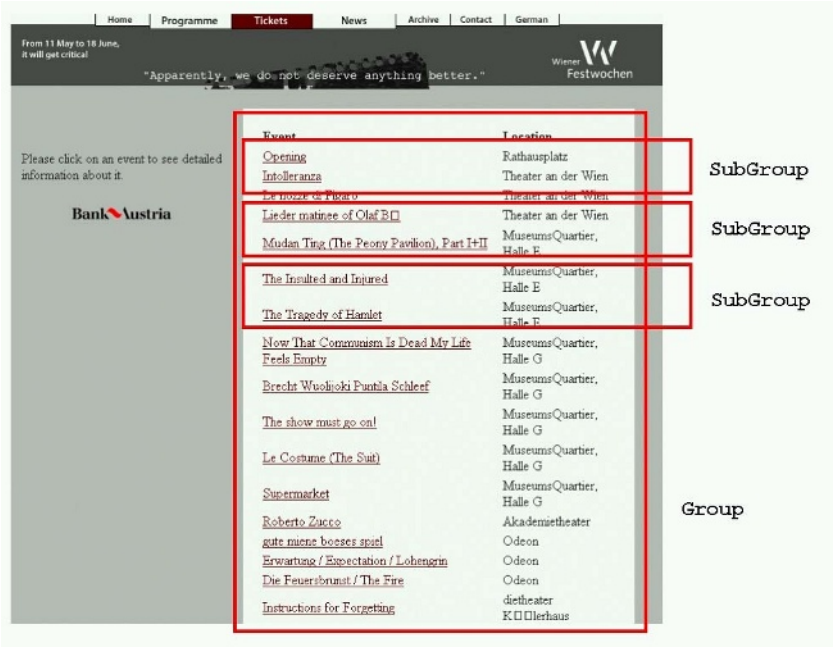


Fig. 2. Page splitting using groups and subgroups

theaters, performances) and their locations. On the page, the entire event information has been marked as belonging to a group. Every two events on the page make up a subgroup.

Depending on the order they appear on a page, each group and subgroup implicitly receives an ID to make it uniquely identifiable (the ID count starts from 0). Each time the layout is presented, only the information in a single group is displayed and the other groups are ignored. If a group contains subgroups, similarly, the subgroups are displayed one after one. Only the groups and subgroups with the given ID are displayed.

By setting a so called *step* value, the subgroups that are to be displayed can be further adjusted. For example, a step value of 2 and a subgroup ID of 0 would display the first and the second subgroup and then stop. The next time the layout is presented, the next two subgroups would be displayed. With a step value of 3, the first three subgroups would be displayed, then the next three and so on.

The described simple mechanism allows the selection of portions of a page during Web application construction so that these portions can be incrementally displayed on devices with small displays and limited memory capabilities. The page splitting processor in the DIWE framework keeps track of the group and subgroup numbers and can receive commands on which splits (i.e., layout fragments) to give out.

The page splitting concept solves the problem of dealing with different page sizes various Web devices are able to display. A database query, for example, can be made to display three results per page for WAP devices and ten results per page for desktop browsers.

Process Partitioning. An important problem page splitting does not solve is how to deal with interactive Web pages that use Web forms. If a Web form in a page is split and distributed over two other pages, for example, it will not work because every Web form has a corresponding *target URL* (i.e., the application logic) that it invokes with the parameters it collects. Hence, if the parameters on the first page are submitted, the information in the following pages will be missing. *Process partitioning* is a technique that allows Web developers and designers to deal with this problem. It uses the page splitting technique to incrementally display Web forms and provides a mechanism to partition the interactive process over a number of independent steps.

Using process partitioning in a WAP e-commerce application for selling cultural event tickets, for example, the ordering process would be distributed over several WML pages. Each time a part of the required information would be collected (e.g., customer's name in the first step, her address in the second, and so on) and sent to an intermediary processor that temporarily stores the input. The intermediary processor would invoke the application logic with the input data it has collected when all necessary data is submitted. In the DIWE framework, the functionality of this intermediary processor is provided by the *process partitioning* processor.

2.2 XSL Stylesheet Pre-processing for Stylesheet Reuse

Imagine an HTML layout for a Web application that is defined in 5 separate XSL files that contain many processing statements such as <xsl:if> and <xsl:template> commands. The traditional and the most popular way to provide a new layout (e.g., a lightweight HTML layout without tables) for the Web application would be to define another 5 XSL files with the appropriate layout information. Most Web engineering approaches that use the XSL standard use XSL in this way.

The problem is that as the number of devices increases, the number of stylesheets increases proportionally. The stylesheets are often quite similar with respect to the processing rules, but mainly differ with respect to the formatting specifications and the Web format being used (e.g., HTML, WML, etc.). One usually ends up copying and adapting the existing XSL stylesheets. As a result, there is often much redundancy in the XSL processing rules and one of the advantages of XSL, ease of layout maintenance, is reduced. Whenever an <xsl:template> statement is modified, for example, all similar device-specific stylesheets have to be updated separately.

Before a layout is added to the content in the DIWE framework, a technique called *XSL stylesheet pre-processing* may be used to eliminate the described duplication and to enable the reuse of existing XSL stylesheets: Instead of the traditional approach of using a new XSL stylesheet for every new device, the information necessary for the device can be *integrated* into the existing stylesheets using special descriptors that help differentiate between the device-specific layout in the stylesheets. The MyXML language compiler processes these specifications and generates the appropriate XSL stylesheet for a particular device. Hence, stylesheet pre-processing reduces the maintenance overhead because a lesser number of XSL stylesheets are needed and XSL processing statement redundancies are eliminated.

2.3 MyXMLDesigner

A user-friendly visual development environment is important for device-independent Web engineering because of the increased complexity of Web applications that are built with XML and XSL. The Web application planning, organization and maintenance overhead may increase significantly with the use of XML and XSL technologies [4]. Web applications may become even more complex when application logic separation support is also provided and separate layouts have to be managed for different Web devices. The MyXMLDesigner visual development environment in the DIWE framework aims to ease device-independent Web application development and maintenance.

MyXMLDesigner provides a user-friendly interface to the MyXML compiler and the run-time device-independence processors in DIWE. It provides the following functionality to Web developers: 1.) Customizable, menus for layout, content and application logic integration, and page splitting and process partitioning support. 2.) Configuration mechanisms that allow the definition and management of devices and the configuration of the device-independence processors. 3.) User-friendly code editing facilities with syntax highlighting for MyXML, XML, XSL and Java documents. 4.) Creation, organization and management of Web projects and project files. 5.) Visual definition and management of Web pages that support multiple layouts. 6.) Generation, deployment and compilation of static and dynamic content (using the MyXML compiler). Figure 3 presents a screenshot of the application.

One of the main distinguishing features of MyXMLDesigner is its support for device-independent Web page creation and management. The Web developer can add device-specific layouts to pages and multi-device support is part of the page creation and management process. MyXMLDesigner provides visual mechanisms for: 1.) Listing the pages that constitute an application. 2.) Displaying information about each device a page supports. 3.) Grouping of pages to ease organization and management. 4.) Displaying which XML content and XSL stylesheets each page uses.

A page is created with a dialog that allows the Web developer to enter descriptive information about the page such as its name and purpose. The user is then presented a *page property* dialog that displays the XML documents and layout stylesheets in the project.

The developer can later choose a page and add arbitrary numbers of devices to a page by simply specifying the stylesheet in the project that applies the suitable layout for the device.

The devices each page supports are listed in a collapsible tree in the project view. In the screenshot in Figure 3, for example, the project panel contains a page *ListOfEvents* that supports the PDA, WAP and HTML device families. By expanding each device node in the tree, the layout components that they support become visible.

2.4 Prototype Implementation

The MyXML language compiler in the DIWE framework has been implemented as a pluggable as well as a stand-alone application. The compiler provides an Application Programming Interface (API) to its functionality and can be configured and invoked within other programs. It supports arbitrary content types, XML content and Java source code generation. It has been written in Java and uses the Apache Xalan [5] XSL processor

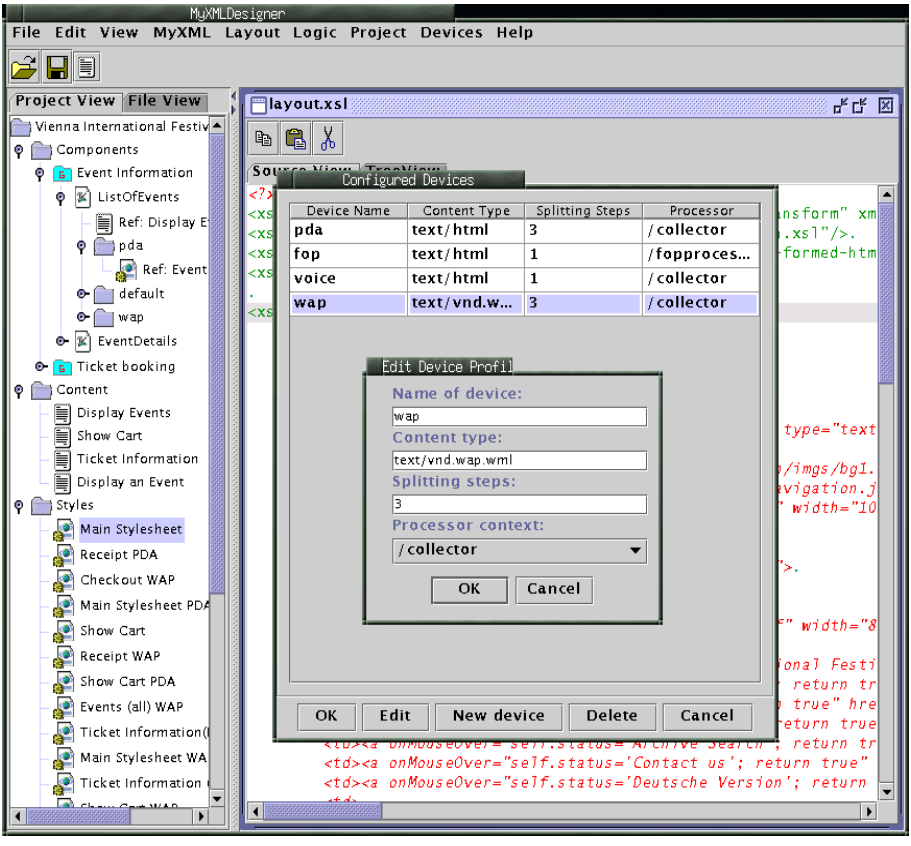


Fig. 3. The MyXMLDesigner visual Integrated Development Environment (IDE)

and the Apache Xerces [6] XML and to add a layout to them. The resulting documents are then processed by the JLex lexical analyzer [7] and the JCup code generator [8] and the embedded MyXML elements are interpreted and resolved.

The device-detection processor has been implemented as a stand-alone Java servlet and uses the Apache Xerces XML parser for processing configuration files. The processor is instantiated and invoked by the servlet engine (i.e., Web server). The *RequestWrapper* class is used in the Java Servlet API Version 2.3 to wrap and modify/extend an incoming HTTP request.

The page splitting and process partitioning processors in the framework have also been implemented as Java servlets and use the session management features provided in the Servlet API.

The logic interfacing processor has been implemented as a Java component and uses the Java reflection mechanism to instantiate and invoke other classes.

All servlet components are based on the Java Servlet API Version 2.3 and have been tested with the Tomcat Servlet Engine version 4.1.24 (Catalina).

MyXMLDesigner has been implemented as a Java Swing application.

3 Device-Independent Web Application Creation with DIWE

The first step in creating a device-independent Web application with DIWE is to configure the device detection processor on the Web server. At run-time, based on the device, the device detection processor responds by dispatching the HTTP request to the corresponding device-specific pages that have been modeled and generated (i.e., XML/HTML files or layout/content encapsulating Java classes in the prototype) using the MyXML language and the compiler.

If the application the user is accessing is static, the device detection processor reads the generated HTML/XML file and passes the result stream to the page splitting and process partitioning processors. These processors process the result stream and split the pages and interactions accordingly. They return HTML/XML to the requesting client device. Figure 4 shows the sequence diagram that illustrates the interactions between the default processors in the framework for processing static content.

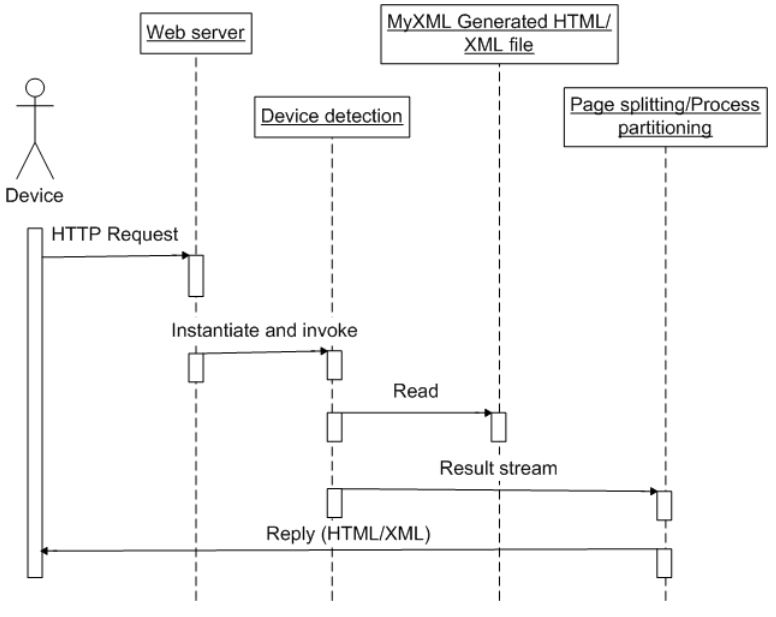


Fig. 4. Sequence diagram showing the interactions between the device-independence processors for static content

If the application being constructed is dynamic, the logic interfacing processor has to be configured and deployed on the Web server. When invoked, the logic interfacing processor transparently and automatically invokes the corresponding layout/content classes based on the name of a device family.

Figure 5 shows a sequence diagram that illustrates the interactions between the default processors in the framework at run-time for processing dynamic content. The device detection processor receives the HTTP request and invokes the application logic.

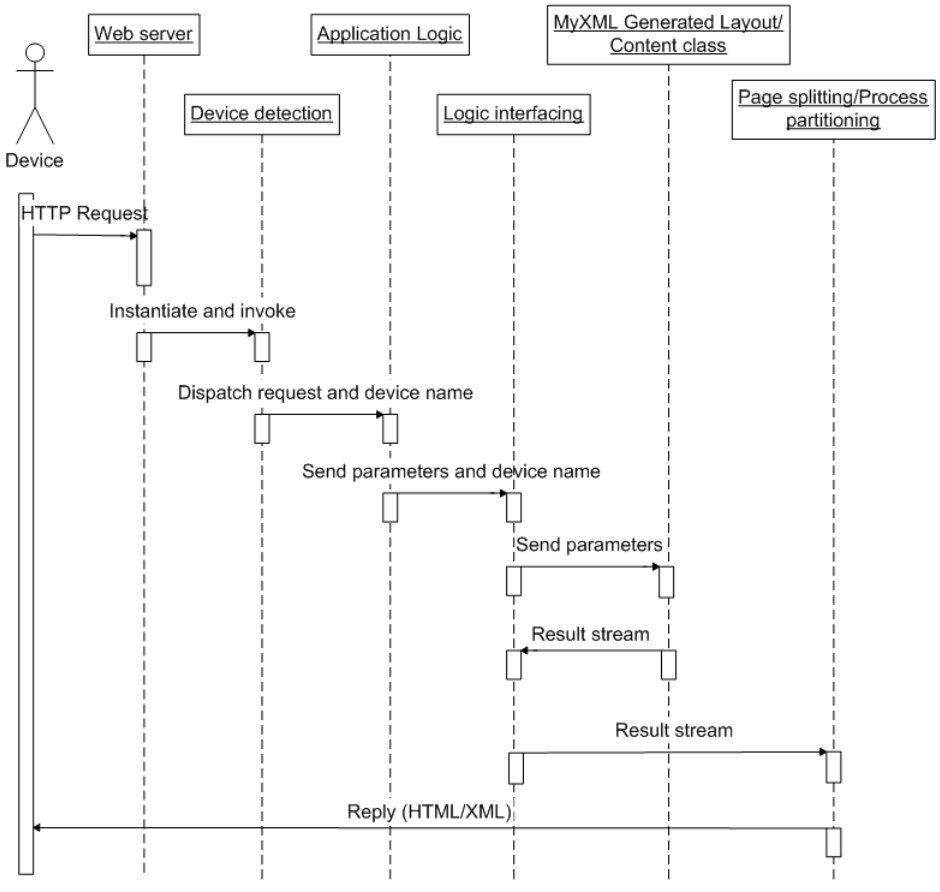


Fig. 5. Sequence diagram showing the interactions between the device-independence processors for dynamic content

The application logic instantiates and invokes the logic interfacing processor with the device name it has received from the device detection processor and parameters it would like to pass to the layout/content class. The logic interfacing processor then instantiates the corresponding layout/content class and invokes it. It passes the result stream returned from the layout/content class to the page splitting and process partitioning processors. These processors process the page splitting and process partitioning information in the stream and return HTML/XML to the calling client device.

4 Case Study: The Vienna International Festival e-Commerce Web Application

To evaluate the DIWE framework and its concepts of LCL separation, page splitting, process partitioning and XSL pre-processing, DIWE was used to design, implement

and extend a device-independent version of the *online ticket booking and ordering Web application* of the Vienna International Festival (VIF) Web site.

The VIF (*Wiener Festwochen*) is the major cultural event in Vienna. Visitors from around the globe come to Vienna during the festival. The festivities take place in various famous theater locations and concert halls. The annual festival, which usually lasts five weeks, presents operas, plays, lectures, concerts, musicals and exhibitions, featuring and hosting eminent international directors, performers and ensembles.

The VIF Web application supports a default full-fledged HTML layout for traditional Web browsers on medium to large displays, a simpler HTML layout for PDA micro-browsers and small displays, and a WAP-layout for WAP-enabled mobile phones. Furthermore, after the user has completed an order, she has the possibility of downloading the receipt as a PDF file. The PDF information is generated dynamically and is treated as an additional device layout that the developer can add to an existing application.

The reader is referred to [3] for more details on the case study.

5 Related Work

Since the mid 90s, the special importance of the different phases in the Web application life cycle has been identified by many authors (e.g., [9]). Several models and methodologies have been proposed for the construction of Web applications and hypermedia systems (e.g., OOHDM [10], RMM [11], Strudel [12]).

With the increasing importance of mobile computing, more researchers have started working on the mobile Web access problems. Several *transcoding* techniques have been proposed that attempt to convert and adapt *existing* content in HTML to be viewable on mobile devices (e.g., [13, 14, 15, 16]). The aim of these approaches is to provide effective algorithms that can convert the content with minimal information loss and provide a satisfactory surfing experience for users.

The page splitting technique discussed in this paper is similar to [14]. [14], however, focuses on the annotation and conversion of existing HTML pages whereas our focus is the annotation (i.e., marking using descriptors) of stylesheets during the design. Furthermore, [14] does not provide a technique to deal with Web forms.

Since the beginning of the year 2000, device-independent Web engineering has been receiving growing interest. Several Web engineering approaches have been proposed. The most notable are OO-H and WebML.

The Object-Oriented Hypermedia (OO-H) method [17] attempts to provide a standard-based framework to capture all the relevant properties involved in the modeling and implementation of Web application interfaces. The methodology contains two views: the navigation view extends a class diagram with hypermedia navigation features and the presentation view uses the different elements regarding the interface appearance and behavior to model a number of interconnected template structures expressed in XML. In comparison to DIWE, OO-H does not use the XML/XSL standards for defining the content and the layout and does not provide support for page splitting or dealing with Web forms.

The Web Modeling Language (WebML) [18] is a high level modeling and specification language for Web applications and is completely XML-based. WebML enables designers to express the core features of a site at a high level without committing to architectural details. A CASE tool is provided that can be used to create XML specifications that are then used to automatically generate device-specific server-side scripts. One drawback of WebML is that it does not support application logic integration.

In [19], an architecture is proposed for building multi-device thin-client Web user interfaces. Developers specify Web-based interfaces using a high-level markup language based on the logical structure of the user interface. At run-time, this single interface description is used to automatically provide an interface for multiple Web devices. The key difference between the approach described in [19] and DIWE is that DIWE uses the standard XSL language for specifying layouts and supports the full layout, logic and content separation.

In [20], Giannetti presents a framework, DIWAF, for constructing device-independent Web applications. The framework aims to adapt content to make it suitable for delivery to many kinds of devices. Similar to our approach of device families, DIWAF supports the idea of designing for the most capable device. The content is then automatically adapted for different device classes, based on author-provided meta-data that guides the adaptation process. The main differences between DIWAF and DIWE are that DIWE uses the XSL standard for specifying layouts, is accompanied by an integrated visual development environment, and the XML content it uses does not need to be specified in a pre-defined way.

Microsoft's new ASP.NET framework [21] has extensive support for the creation of Web pages and Web applications. The Visual Studio graphical development environment enables Web developers to rapidly create Web pages, Web sites and Web applications. The way ASP.NET deals with Web applications is quite low-level: The framework lacks a higher-level, language-independent model for dealing with device-independent Web applications.

Recently, Microsoft has started shipping the Mobile Developer Toolkit that is an extension to the Visual Studio Development Environment. This toolkit provides a visual environment for creating and deploying Web applications that are explicitly designed for mobile devices. The developer creates an application by placing components such as *buttons* and *text fields* into forms. Content is also inserted into these forms in terms of *label* components. Based on the characteristics of a device (e.g., PDA, WAP phone, etc.), the platform automatically adapts and renders the forms to be viewable on the device.

The main advantage of this development platform is that applications can be rapidly developed without a high technical knowledge. The disadvantage is that the created applications are not flexible and rather difficult to maintain because of the use of forms (e.g., changing a logo on each page could mean that the developer has to manually delete each logo component on every form).

Cocoon [22] is a Java servlet-based application server that is based on freely available XML parsers (e.g., Xerces [6]) and XSL processors (e.g., Xalan [5]). Cocoon can be used for the real-time translation of XML files on a web server to HTML and

any XML-based Web format such as WML. The platform does not provide mechanisms to deal with small screens and memory limitations. Furthermore, similar to the Mobile Developer Toolkit, cocoon provides an implementation platform and technology, but does not aim to support the design or maintenance stages of Web applications.

The Mobile Station Application Execution Environment (MExE) [23] is a wireless protocol that is designed to be incorporated into smart mobile phones. MExE is primarily aimed at the next generation of powerful smart phones and will support a wide range of man-machine interfaces such as voice recognition.

The MPEG-7 (and more recently the MPEG-21) standard have been gaining increasing attention by the content modeling community. MPEG-7 and MPEG-21 are ISO/IEC standards developed by MPEG (Moving Picture Experts Group) and provide standardized core technologies allowing the description of audiovisual data content in multimedia environments. The MPEG standards will probably play an important role on the Web in the near future. Because these standards are XML dialects, they can easily be used in the DIWE framework for defining content and layout (e.g., similar to Web formats such as WML and VoiceXML).

iMode [24] is NTT DoCoMo's mobile internet access system that has gained large popularity in Japan. iMode uses cHTML (compact HTML) which is a subset of traditional HTML. Hence, pages and content have to be specifically designed for iMode devices and cannot be reused for other devices such as VoiceXML browsers. We use XML in DIWE because it provides more flexibility with respect to the number of devices and the Web formats being supported (e.g., MPEG-7 for multimedia, VoiceXML for speech access and WML for WAP devices).

The Open Pluggable Edge Services (opes) [25] working group is currently working on a framework and protocols to both "authorize and invoke distributed application services while maintaining the network's robustness and end-to-end integrity". Like DIWE, OPES aims to adapt the provided service to client devices.

6 Conclusion

This paper introduced a novel conceptual framework for device-independent Web engineering. The Device-Independent Web Engineering (DIWE) framework consists of the *MyXML language*, a compiler that can interpret the language, the *MyXMLDesigner* visual development environment, and four basic run-time *processors* that are configured and deployed on the Web server at run-time to provide device-independence support. These processors are Web applications themselves.

The DIWE framework uses two techniques, *page splitting* and *process partitioning* that allow the Web developer to tune the selected information and the sizes of generated pages according to the characteristics of a device that is being targeted. The framework also introduces a novel technique called *XSL stylesheet pre-processing* that allows the reuse of *existing* XSL stylesheets when adding new devices to a Web application.

In 2002, DIWE was used to construct a device-independent version of the Vienna International Festival e-commerce Web application.

References

1. Hakon Wium Lie and Janne Saarela. Multipurpose Web Publishing: Using HTML, XML, and CSS. *Communications of the ACM*, 42(10), October 1999.
2. Clemens Kerer and Engin Kirda. Layout, Content and Logic Separation in Web Engineering. In *Proceedings of the 9th International World Wide Web Conference, 3rd Web Engineering Workshop, Amsterdam, Netherlands, May 2000*, number 2016 in Lecture Notes in Computer Science, pages 135–147. Springer Verlag, 2001.
3. Engin Kirda. *Engineering Device-Independent Web Services*. PhD thesis, Technical University of Vienna, 2002. `http://www.infosys.tuwien.ac.at/staff/ek/phd.pdf`.
4. Clemens Kerer, Engin Kirda, Mehdi Jazayeri, and Roman Kurmanowytsch. Building XML/XSL-Powered Web Sites: An Experience Report. In *Proceedings of the 25th International Computer Software and Applications Conference (COMPSAC), Chicago, IL, USA*. IEEE Computer Society Press, October 2001.
5. Apache. Xalan XSL Processor, 2001. `http://xml.apache.org/xalan-j`.
6. Apache. Xerces XML Parser, 2004. `http://xml.apache.org/xerces-j`.
7. Eliot Berk. JLex: A Lexical Analyser Generator for Java, 2004. `http://www.cs.princeton.edu/~appel/modern/java/JLex/`.
8. Scott Anian. JCup: CUP Parser Generator for Java , 2004. `http://www.cs.princeton.edu/~appel/modern/java/CUP/`.
9. V. Balasubramanian, Bang Min Ma, and Joonhee Yoo. A Systematic Approach to Designing a WWW Application. *Communications of the ACM*, 38(8):47–8, August 1995.
10. Daniel Schwabe and Gustavo Rossi. The Object-Oriented Hypermedia Design Model. *Communications of the ACM*, 38(8):45–6, August 1995.
11. Tomas Isakowitz, Edward A. Stohr, and P. Balasubramanian. RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 38(8):34–43, August 1995.
12. Mary Fernandez, Daniela Florescu, Jaewoo Kang, and Alon Levy. Catching the Boat with Strudel: Experiences with a Web-Site Management System. In *Proceedings of Sigmod '98, Seattle, Washington, USA*, pages 414–425, June 1998.
13. Timothy W. Bickmore and Bill N. Schilit. Digestor: Device-Independent Access To The World Wide Web. In *Proceedings of the 6th World Wide Web Conference, Santa Clara, CA, USA*, 1997.
14. Masahiro Hori, Goh Kondoh, Kouichi Ono, Shin ichi Hirose, and Sandeep Singhal. Annotation-based Web content transcoding. In *Proceedings of the 9th International World Wide Web Conference, Amsterdam, Netherlands*, May 2000.
15. Orkut Buyukkokten, Hector Garcia-Molina, and Andreas Paepcke. Focused Web searching with PDAs. In *Proceedings of the 9th International World Wide Web Conference, Amsterdam, Netherlands*, May 2000.
16. Orkut Buyukkokten, Hektor Garcia-Molina, and Andreas Paepcke. Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices. In *Proceedings of the 10th International World Wide Web Conference, Hong Kong, China*, May 2001.
17. Jaime Gomez, Christina Cachero, and Oscar Pastor. Conceptual Modeling of Device-Independent Web Applications. *IEEE Multimedia*, 8(2):26–39, April-June 2001.
18. Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. In *Proceedings of the 9th World Wide Web Conference, Amsterdam, Netherlands*, volume 33 of *Computer Networks*, pages 137–157. Elsevier Science B.V, May 2000.
19. J. Grundy and Wenjing Zou. An architecture for building multi-device thin-client web user interfaces. In *Proceedings of Advanced-Information-Systems-Engineering.-14th-International-Conference,-CAiSE-2002.*, pages 728–732. Springer-Verlag, 2002.

20. Fabio Giannetti. Device independence web application framework (diwaf). In *Proceedings of the W3C Device Independent Authoring Techniques Workshop*, September 2002.
21. Essential .NET :Component Development with C#. Technical report, Developmentor.
22. Stefano Mazzocchi. The Cocoon Project Home Page, 1999-2000. `http://xml.apache.org/cocoon/`.
23. Mobilemexe. MExE, 2003. `http://www.mobilemexe.com`.
24. Eurotechnology. Imode, 2003. `http://www.eurotechnology.com/imode/`.
25. IETF. Open Pluggable Edge Services (opes), 2003. `http://www.ietf.org/html.charters/opes-charter.html`.

A Conceptual Framework for Monitoring and Control System Development

Stefania Bandini, Alessandro Mosca, Matteo Palmonari, and Fabio Sartori

Department of Computer Science, Systems and Communication (DISCo),
University of Milan - Bicocca,
via Bicocca degli Arcimboldi, 8
20126 - Milan, Italy
tel +39 02 64487857 - fax +39 02 64487839
{bandini, alessandro.mosca, matteo.palmonari
fabio.sartori}@disco.unimib.it

Abstract. This paper illustrates a conceptual framework for the development of Monitoring and Control Systems (MCS), based on a four level agent-based architecture. Traditional MCS are designed according to a three–level architectural pattern, in which intelligent devices are usually devoted to evaluate if data acquired by a set of sensors could be interpreted as anomalous or not.

Possible mistakes in the evaluation process, due to faulty sensors or external factors, can cause the generation of undesirable false alarms. To solve this problem, our framework introduces a fourth level to the traditional MCS architecture, named *correlation level*, where an intelligent module, usually a Knowledge–Based System, collects the local interpretations made by each evaluation device building a global view of the monitored field. In this way, possible local mistakes are identified by the comparison with other local interpretations. The framework has been adopted for the development of Automotive MCSs.

1 Introduction

Pervasive (or Ubiquitous) Computing (PC) has become a very important topic for researchers in Computer Science. Today, the measurement of physical quantities from the real world and its translation into a digital representation is supported by many sophisticated technologies. Old problems, like size, quality and reliability of measurements, manufacturing and cost of devices have been solved, but new ones are arising. Thus, while most of the industrial and academic research has been focused in past years on the development of efficient communication protocols, now the trend is the design of applications suitable for dealing with the huge amount of information continuously transmitted by devices daily used by a large amount of people. Basically, PC is concerned with a new way of conceiving the interaction among humans (users) and computing devices. According to [22], we can look at PC nature as its ability to disappear into the user subconscious: mobile devices, sensors and integrated environments depict a scenery in which users will interact with embedded devices, each one dynamically connected with each other and almost disappearing. This extremely general characterization makes PC a paradigm rather than a specific research area. To make some clearness on Lyon's

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 111–124, 2004.

footprints [14] PC can be classified into three wide application areas as long as personal, spatial location or biometric data are collected. The first area deals with Personal Digital Assistants, massively networked devices, smart cards, intelligent textiles, and so on. The second one is about infrastructures concerned with space and spatial location, monitoring, intelligent buildings, and so on. The third is close to the first one, but is related to storing and processing medical and biological information. Within the second research area, we focus our attention on Monitoring and Control Systems (MCSs). A MCS aims to support humans in taking decisions about problems which can arise in critical domains, and can be characterized as follows, on the basis of its functionalities:

1. collect data about the monitored situation (i.e. monitor), by using a group of sensors;
2. evaluate if these data concern an anomalous situation;
3. in case of anomalous situations, take the proper actions (i.e. control) in order to solve problems.

An action is generally the generation of an alarm in order to notify humans about the problem. Due to their features, MCSs should be intelligent. For this reason, MCSs have been typically developed exploiting Artificial Intelligence technologies, like Neural Networks [8], Data Mining [11], Knowledge-Based Systems [1], and so on. One of the most important problems of current MCSs is that the interpretation of data is local: each sensor is typically linked to a dedicated device for the evaluation of measured data, but this device is not able to communicate with other ones. Thus, each device gives a local evaluation of the monitored field. This is a real problem when sensors are faulty or the interpretation of data is not precise due to external factors, so that false alarms are generated. In this paper, we propose a conceptual framework for the solution of this problem in the development of MCSs through the correlation of alarms [20] coming from single evaluation devices. The framework consists of a four-level agent-based architecture, introduced in Section 2, where each level is governed by a specific agency [21], which is a system module acting as an agent. A correlation agency is responsible for the creation of a global view of the monitored area, by collecting all local evaluations and elaborating them in order to highlight possible errors or omissions. Since correlation is a typical human activity, a correlation agency should be designed as a knowledge-based system, but this choice is not mandatory. Since MCSs are intrinsically distributed over wide areas (sensors are placed in different zones of the field to be monitored), the agent model seems to be the most natural AI approach for the development of such kind of systems.

With respect to agent technologies, our architecture lays at a higher level of abstraction. "Agency" is a functional concept, that is supposed to meet a set of specifications and to perform some functions which can be entirely lead to a given definition of agent (e.g that found in [7] [13] or [4]). An agency may be implemented

- by means of agents having the same functionalities as the agency itself or whose interactions and cooperations allow the emerging of agency functionalities;
- by means of a software architecture that does not exploit an agent–based approach.

In the first case, this means that it is not necessary to assume a unique definition of agent or a unified approach to agent technology choosing between deliberative agents (e.g., according to a BDI approach as in [12]) or reactive ones [9]. On the other hand, the

architecture proposed in this paper tries to supply a well–defined framework within which different approaches and techniques can be combined to fully exploit the advantages coming from the peculiarities of each of them.

Next section briefly introduces the framework pointing out the main differences from the traditional ones. Section 3 describes possible application domains of the framework: in particular, the Automotive System field [19] looks very promising, since the framework is able to meet a lot of constraints coming form it. As an example of the framework suitability to the automotive domain, Section 4 presents the SAMOT system, a knowledge-based application for traffic monitoring and control currently working on some important Italian highways. Then, the paper focuses on the problem of mobility in Automotive MCS. With respect to a system like SAMOT, whose detection devices are fixed sensors which monitor fixed road portions, allowing to such devices to move in a complex space could bring to the arising of both conceptual and computational problems. Although from a conceptual standpoint the framework presented in this paper seems to be suitable for the design of MCSs characterized by mobile sensors too, a new research project named $CO^{2*}PC$ has recently started in order to effectively demonstrate this hypothesis, or to negate it. The motivations for this project are briefly described in in Section 5. Finally, some conclusions are briefly pointed out.

2 A Four Level Architecture for MCS Development

Today, most MCS are structured on three logical levels, shown in Figure 1:

- observation, where the state of a monitored field is periodically captured by a specific monitoring agency (MA), usually a set of sensors;
- interpretation, where values detected by sensors are evaluated by a specific interpretation agency (IA);
- actuation, where specific actions are taken by a specific actuation agency (AA), depending on the interpretation results.

Global monitoring and control functions of a system emerge from the interactions among the different agencies. The most critical step in the process above is the interpre-

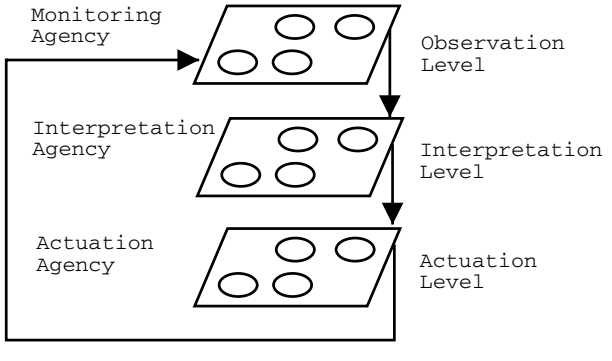


Fig. 1. The general architecture of a monitoring/control pervasive system

tation of data: an error could cause the system takes a wrong decision. Although a lot of sophisticated technologies have been developed, and efficient detection algorithms have been implemented, interpretation of complex data, like images, smells and so on is still a problem. Moreover, MCS have only a local perception of the environment, while a global one would be more suitable to avoid errors in the interpretation. Thus, they tend to be brittle and often ineffective, since they are too subject to noise, possibly triggering not necessary or even undesirable control actions.

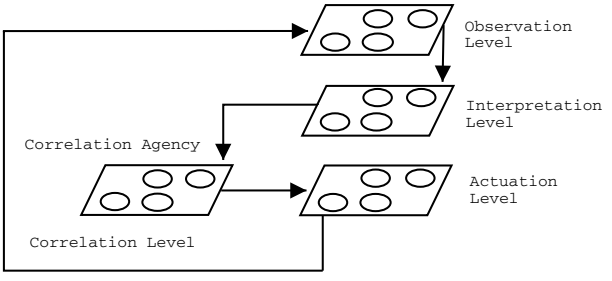


Fig. 2. The general architecture of a monitoring/control pervasive system, extended with the correlation level

Our solution to this problem is shown in Figure 2, where a fourth logical level has been added between the interpretation and actuation ones. This level is characterized by correlation: interpretations of data are correlated, by a specific correlation agency (CA), in order to identify possible mistakes and avoid the generation of wrong decisions. The correlation agency has the role of managing, filtering and correlating information coming from the interpretation agency, so that:

- a global view of the monitored situation that contextualizes different local interpretations is created;
- control actions are taken according to this global view, through the adoption of a precise and explicit control policy.

Finally, actuation agency is committed to effectively carry out the control task, influencing the environment. Currently, many monitoring and control technologies take control actions only on the basis of local interpretations of data collected by sensors. The introduction of correlation level within the architecture of MCSs causes a change in the functionalities of interpretation level, switching from take a decision to suggest a decision. All the suggestions are evaluated by correlation level, so that a MCS will chose the proper actions to be adopted according to a global view of the state detected by sensors rather than a local one. The correlation is guided by a specific policy, depending on the kind of problems to be solved. The policy should at least specify:

- actions to be taken in response to a global critical situation;
- rules to correlate different local situations into a global one.

Probably, the most suitable paradigm for the realization of the architecture above is the agent-based model. In this case, control and monitoring agencies can be considered as sets of agents, with at least one element. When more than one agent belong to them, the relationship between an agency and agents which compose it can be described as follows:

- the functionalities of each agent *correspond to* the functionalities of the agency. For example, if the goal of the interpretation agency is to interpret data flows coming from homogeneous sensors in order to detect possible presence of smoke in the air, this detection can be viewed as the goal of interpretation agency and of single agents as well;
- the functionalities of the agency result to be a *plain sum* of the functionalities of different agents. This can happen when the task of an agency is the detection of a local anomalous situation resulting from the interpretation of different sensor signals;
- the functionalities of the agency *result from the interaction* (cooperative or competitive) of different agents. For example a correlation agency may be made of different agents each one carrying out a specific task. The agency's correlation task may be the result of interactions among different agents.

As pointed out in the introduction, no assumptions about a particular approach to agents or a specific definition of 'agent' are required: this architecture is based on the idea that different kinds of agents turn out to be useful to accomplish different tasks. Generally, with respect to the four levels introduced above, is is possible to make the following considerations:

- elements of MA, IA and AA can be typically represented as reactive agents. Each of them receives an input and makes an action immediately: MAs' input is the observation of local states and the action is its transmission to IAs; IAs' input is the value received from MAs and the action is to suggest a decision,; AAs' input is the decision taken by CAs and the action is to accomplish it;
- CAs' members can be represented by utility–driven agents [6]; in this case, CA agents receive all the evaluations of local state from interpretation agency, correlate them and make deliberations about the proper actions to be taken. Then, they notifies actuation agencies about their decisions. On the other hand, CAs' members could be defined according to a BDI (Beliefs-Desire-Intention) approach [10] [23] too. From this point of view, the world representation (i.e. *belief*)is partially provided by the interpretation agency, goals (i.e. *desires*) almost meet the system ones and committed plans or procedure (i.e. *intentions*) outputs must map over the actuation agency.

Nevertheless, independently from the adoption of a real agent–based approach, the four agencies can be designed and implemented exploiting different technologies and techniques. Different approaches can be used at both a specific level and considering more than one level, so that the system behavior results a combination of symbolic and sub–symbolic subsystems [5] [2]. In particular, as illustrated in section 4 about the SAMOT system, the first two levels (and possibly the fourth one) are often developed by means of numeric techniques such as neural networks, genetic algorithms and so on,

which are particularly suitable to accomplish local detection tasks and perform pattern recognition. On the contrary, the correlation agency may take advantage of explicit knowledge on the domain. Taking atomic interpretations as inputs, correlation can be naturally made exploiting a symbolic approach (e.g. a Knowledge Based System, Case–Based Reasoning, BDI agent approach, and so on). For example, if the correlation agency consists of an utility–driven agent its reasoning may depend on the adopted *policy*, which is a sort of utility function. In this case, a very suitable representation for a policy is a rule-based system, whose rules are activated by local state values.

3 Application Domains

The framework previously introduced is general enough to be applied to a lot of real systems, devoted to solve problem concerning critical domains. For example, a system for the detection of dangerous gases inside a building could be designed as follows:

- An observation level, where MA is made of a set of sensors for the detection of gases. The sensors could be placed in every room of the palace;
- An interpretation level, where IA is composed by a set of devices able to distinguish abnormal quantities of potentially dangerous substances in the air. Such devices could be based on neural network or genetic algorithm technology. In case of potential danger, the device could launch an alarm;
- A correlation level, where CA is a Knowledge-Based System that receives one or more alarms from IA and correlates them temporally and spatially, in order to chose the best way to tackle the problem and avoid false alarms;
- An actuation level, where AA is made of speakers and lightening panels, placed in each monitored room, which are activated to notify people that a danger has been detected.

The presence of a correlation agency is fundamental to avoid the generation of false alarms. Let's suppose that a device of the interpretation agency is faulty and detects the presence of a dangerous gas in the air. It would immediately activate speakers and lightening panels to notify people about a not existing abnormal situation, probably causing a not justified panic in them. In order to avoid such a situation, a correlation agency would have compared the alarm generated by the faulty device with the messages received by the other devices spatially adjacent to it, over a sufficient period of time. On the basis of this analysis, the correlation agency would have been able to deduce that the considered device was faulty, so that no false alarm was produced. Many different domains can be characterized as the example above: in particular, the Automotive System Development domain has been recently being an important research field, due to the efforts of European Union in reducing car accident victims and improving highway security. Such systems may be divided into a lot of typologies:

- systems for traffic monitoring and control (TMCSs);
- systems for stolen vehicle monitoring and control (SVMCSs);
- systems for emergencies monitoring and control (ETMCSs);
- other kinds of MCSs.

In the following, we show how the framework described above could be a very useful guide to the design and implementation of such kind of systems.

3.1 Traffic Monitoring and Control Systems

The aim of a traffic monitoring and control system is to identify traffic flow anomalies, to alert operators and to support them in the management of emergencies [3]. Such systems can exploit a lot of sophisticated technologies for traffic anomalies detection, like magnetic-loop sensors, infrared, video image processors and so on. In particular, the adoption of video-based technologies allows the improvement of TMCS effectiveness, providing them with the possibility to clearly distinguish different critical situations. According to our framework, a TMCS can be divided into four levels:

- an observation level, where an observation agency is made of a set photo-cameras, video-cameras, and so on, each one devoted to periodically or continuously acquire information about a specific portion of the monitored road;
- an interpretation level, where an interpretation agency made of Video Image Processors [17] (VIPs) locally evaluates the traffic conditions, generating an alarm in case of problems (queue, car accidents, slow traffic and so on);
- a correlation level, where a correlation agency could be a knowledge-based system, which receives local alarms from VIPs and creates a global view of traffic conditions according to a specific policy, that is a representation of the knowledge owned by human experts, in order to distinguish false alarms from real ones. Then, the correlation agency may chose an action to be taken for solving a critical traffic situation on its own or support humans in their decision making process;
- an actuation level, at which an actuation agency made of lightening panels or speakers receives instructions from correlation agency about which lightening panels or speakers have to be activated in order to notify drivers, and turns them on.

A working TMCS, named SAMOT, will be presented in the next section as an example of successful application of the framework.

3.2 Stolen Vehicle Monitoring and Control Systems

The aim of a system for monitoring and control of stolen vehicles is to support vehicle owners and police in the process of individuation and recovery of stolen cars, trucks, motorcycles, and so on. As in the case of TMCSs, a lot of sophisticated technologies are nowadays available for the design of SVMCSs, such as Global Positioning System (GPS), efficient wireless communication protocols (i.e. GSM, GPRS, UMTS, Bluetooth, and so on), and powerful car–computers [15] [18]. By the exploitation of these technologies the architecture of a SVMCS based on our framework could be described as follows:

- an observation level, where an observation agency made up of a GPS antenna and other sensors (e.g accelerometers, crash sensors) is devoted to periodically or continuously acquire information about the vehicle position;
- an interpretation level, where an interpretation agency made up of a car computer and one or more wireless communication devices is placed. The car computer evaluates

if the vehicle is moving; in this case, it tries to establish if the movement is legal or not (e.g. the car start-key is present or not, the door closure mechanism is working, and so on). In the last case, wireless communication devices are activated to send information collected by the GPS antenna;
- a correlation level, where a correlation agency made of a knowledge-based system and a geographical map database receives GPS data and uses them to identify the precise geographical position of the car onto a map in order to define all the possible stolen vehicle itineraries. Then, the correlation agency may notify the nearest police departments placed along such itineraries.
- an actuation level, at which an actuation agency is made of a set of police departments, each one devoted to control a specific itinerary defined by the correlation agency, in order to block thieves and recover the stolen vehicle.

3.3 Emergency Monitoring and Control Systems

The goal of a system for monitoring and control of emergencies is to support the personnel of an Emergency Room in the identification of car accidents, in order to quickly send ambulances for the recovery of drivers. The architecture of an EMCS could be characterized as follows:

- an observation level, where an observation agency made of a GPS antenna and a set of car sensors is devoted to periodically or continuously acquire information about the vehicle state (e.g. position, speed, acceleration, air-bag functioning and so on);
- an interpretation level, where an interpretation agency made of a car computer and one or more wireless communication devices is placed. The car computer evaluates if the vehicle has crashed; in this case, wireless communication devices are activated to send information collected by the observation agency;
- a correlation level, where a correlation agency made of a knowledge-based system and a geographical map database receives GPS data and uses them to identify the precise geographical position of the car onto a map. Then it tries to evaluate the accident impact on the driver, exploiting other information sent by interpretation agency (i.e. speed and acceleration);
- an actuation level, where an actuation agency is made of a set of ambulances among which the correlation agency chooses the nearest one to the geographical position of the vehicle, allowing it to reach the driver quickly.

4 SAMOT, a System for Automatic Monitoring of Traffic

In the previous section, the automotive system development has been introduced as a very suitable domain for the application of our framework. In this section, we describe a real automotive system, named SAMOT, designed and implemented on the basis of it. SAMOT [16] is a traffic monitoring and control system developed in collaboration with Project Automation S.p.A. and Societá Autostrade (i.e Italian Highway Company), currently installed and working on some Italian highways. Figure 3 shows its four-level architecture.

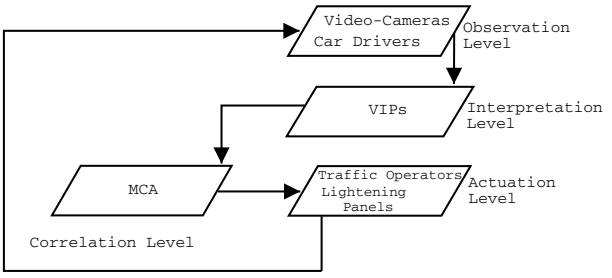


Fig. 3. The four–level architecture of SAMOT

In SAMOT, the monitoring agency is made of a set of video-cameras, each one devoted to the surveillance of a small portion of the monitored road. A Video-Image Processor (VIP) is associated to every camera, in order to make an interpretation of the traffic situation. The set of VIPs is the interpretation agency of SAMOT. The MCA is the correlation agency of the system, and it will be further described in this section Finally, traffic operators and lightening panels placed along the road are the elements of actuation agency: according to the output of MCA, traffic operators (or the MCA itself) activate one or more lightening panels, showing warning messages to car drivers. MCA supports traffic operators in the interpretation of traffic condition, correlating atomic traffic anomalies and filtering them taking into account their spatial and temporal location. MCA is a knowledge-based module acting as a goal-driven agent: its utility function implements a policy, which is a set of rules expressing the expertise of operators. The main entities involved in the correlation process are the monitored highway, alarms coming from VIPs and Anomalous Traffic Patterns (ATP), complex events dynamically built up starting from alarms. In the SAMOT model, the highway is viewed as a composition of spatially adjacent and regular sectors, each one monitored by a single video-camera linked to a VIP. The management of ATPs represents how the correlation agency of SAMOT works, and it is the result of a deep knowledge acquisition campaign. The last function of MCA is to filter alarms coming from VIPs. On the basis of spatial and temporal correlations, MCA is able to create a global view of the traffic situations, recognizing possible false alarms and supporting traffic operators in identification of real problems. Then, traffic operators or the MCA itself can undertake the proper actions to solve them (e.g. activating the correct lightening panels and notifying car drivers).

Figure 4 shows a high-level view of how MCA works with respect to the other agencies SAMOT is composed by. An anomalous traffic pattern represents a traffic anomaly referring to multiple cameras and, consequently, a sequence of local traffic situations detected by VIPs. The creation of ATPs reflects precise rules of spatial adjacency: two or more local spatially adjacent anomalies can be merged by MCA. Then, ATPs are compared by MCA on the basis of temporal adjacency: for instance, if two or more temporally near ATPs reports a common anomaly, such as slow traffic, MCA can deduce that there is a queue caused by a car accident. The development of MCA has been fundamental: in fact, despite of their high level technology, VIPs often fail the evaluation of traffic conditions, due to a lot of reasons, generating false alarms about critical situations. The main reason for this is that VIPs don't work exploiting specific knowledge:

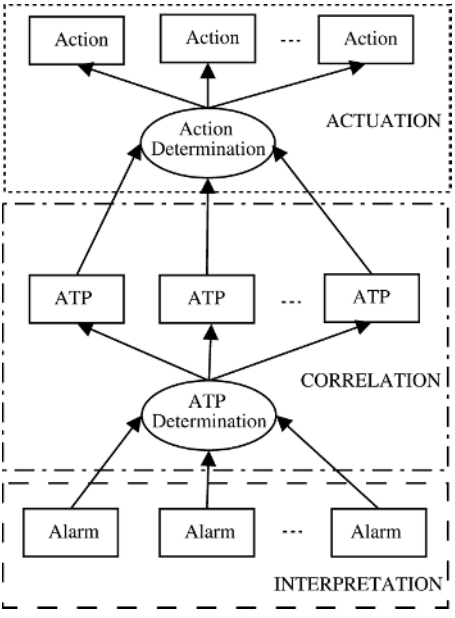


Fig. 4. Relationships between MCA and other SAMOT agencies

their interpretation is limited to the analysis of images supplied by video-cameras, by which they're able to recognize possible anomalies, but their response can be negatively influenced by a lot of natural and unpredictable factors (i.e. shadows on the road-ground, water and so on). An ATP can be considered as a set of adjacent highway sectors characterized by common local conditions (i.e. normal traffic, queue, incident, slow traffic and so on). Moreover, the distribution of ATPs over the road dynamically changes: an ATP characterized by queue at certain instant could modify its state to normal traffic in the future.

5 Adding Mobility to Automotive MCS: The $CO^{2*}PC$ Project

Last section has presented the SAMOT system, a practical application of the framework in the context of MCS. In SAMOT the elements of interpretation agency (i.e. VIPs) are fixed in the space. Today, the complexity of Automotive MCS is increasing rapidly, due to the development of more and more sophisticated devices and technologies which allow to collect a wide set of data and use them to supply new functionalities. We can define such systems as *Mobile Automotive Monitoring and Control Systems* (MAMCS). An example of MAMCS are Emergencies MCS described in section 3.3, in which the adoption of a mobile data acquisition device that exploits a GPS antenna allows a driver to be quickly reached by an ambulance in case of car accident. Figure 5 shows a high level view description of an element of IA when mobility is considered.

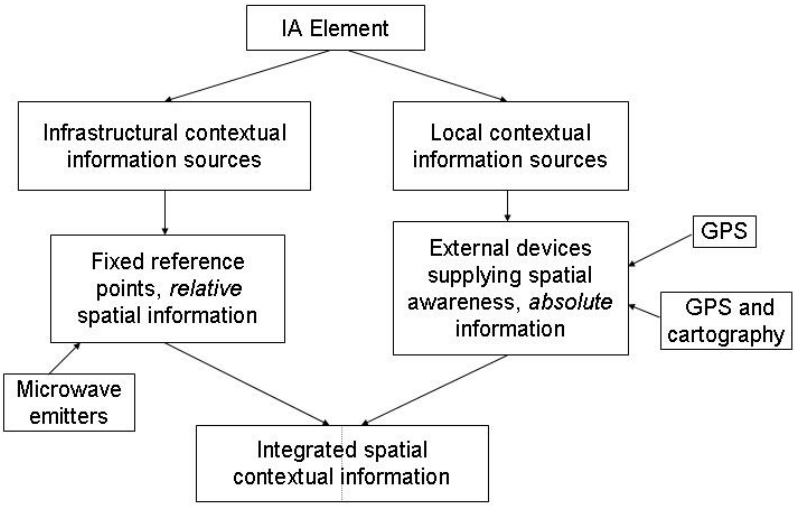


Fig. 5. A high level view of IA elements when mobility is considered

From the conceptual point of view, our framework is suitable for the design of MAMCS too, due to the presence of correlation agency that works independently from the nature of IA elements (mobile or not), but new problems could arise from an engineering perspective as the mobility of IA elements introduces the need for more complex communication protocols with respect to the SAMOT system.

Nevertheless, even assuming to have suitable communication protocols, MACMS introduce new challenges from a modelling point of view, since no hypothesis about the geographical distribution od IA elements can be made, i.e. the space in which IA components move is more complex than the one monitored by VIPs in the SAMOT system. Adding mobility to a MCS component prevent from taking location data provided by IA as given independent from the current state of the system. For example, in SAMOT the correlation agency correlates alarms coming from different IA on the basis of their spatial adjacency relations, and these relations are fixed and persist along time. When IA elements are mobile, spatial location information must be determined *dynamically*.

Moreover, the required spatial location information may not be a simple set of coordinates returned by a GPS (sensor level), but something more complex including, e.g. the specification of a street over a cartography, a room inside a building, and so on. By the exploitation of Knowledge–Based approaches, the system should be able to infer spatial information from other data (e.g. integrating a GPS data with a cartography) avoiding errors or omissions due to technological limitations, like, for example, GPS imprecisions. In order to do this, the needed knowledge could be placed both at the interpretation and the correlation level within the architecture of MAMCS, and this choice involves both architectural and computational issues according to

different types of application domains (e.g. in the previous example it can depend on the computational power of on–board devices, that on its turn is influenced by other domain dependent parameters like the required reliability in detecting and sending data).

Our framework could be useful to handle these problems because it identifies different layers at which it is possible to distribute computational load and ”intelligence”.

In order to test the computational effectiveness of a MAMCS designed according to our four level architecture, a new research project, named $CO^{2*}PC$ (COrrelation and COntextualization of data collected by mobile sensors in Pervasive Computing domains) has recently started. The project is a collaboration between the University of Milano–Bicocca and SarasLab, whose aim is the definition of an architectural pattern for the development of complex monitoring and control systems, like MAMCS, based on the correlation of heterogeneous data acquired by different kinds of sensors through the exploitation of Knowledge–Based approaches.

6 Concluding Remarks

The paper has presented a framework for the development of MCSs. The aim of these systems is to take decisions in order to response to anomalous situations which can arise in critical contexts. Most times, such decisions concern the generation of alarms to alert human operators that a potentially dangerous problem has been originated. MCSs are typically designed according to a three-level architecture, in which the values related to monitored field measured by a set of sensors are evaluated as normal or not by some intelligent devices. Then, actions are taken on the basis of such evaluations. The biggest problem concerning this architectural pattern is the locality of evaluations: each intelligent device is devoted to the interpretation of measurements made by a specific set of sensors, monitoring a very specific area, and it doesn't communicate with other devices in order to compare its evaluation with other ones. In this way, a lot of false alarms are generated with no possibility to identify them before their creation. The framework presented in this paper tackles this problem by adding a fourth level of elaboration, at which correlation among the different local evaluation is made in order to create a global view of the monitored field. Although the implementation of the correlation policy through a knowledge-based system is suggested, this choice is not mandatory. The framework is extremely modular:

- The correlation level can be present or absent depending on the kind of system to be developed and the spatial extension of the area to be monitored;
- each level can be designed and implemented exploiting different approaches (neural networks, genetic algorithms, knowledge-based systems, and so on) with no side-effects on other ones.

Moreover, the framework is general enough to be applied to the design and implementation of a wide variety of MCSs. In particular, the framework is suggested to be adopted for the development of automotive systems.

Acknowledgements

Authors wish to thank all the people that have contributed to this paper. In particular, we thank Davide Bogni (Project Automation S.p.A.) and Piercarlo Ravasio (SarasLab) for their collaboration in the SAMOT and $CO^{2*}PC$ projects respectively.

References

1. R. L. Westra, *Design of a knowledge-based monitoring and control system*, Technical report, Department PRG/Faculty Math. and Comp. Sc., University of Amsterdam, Esprit 2439 report D31, N. J, 1989.
2. Wilson, A. and Hendler, J., *Linking Symbolic and Subsymbolic Computing*, Technical Report, Dept. of Computer Science, University of Maryland, 1993.
3. Ferrier, N.J., Rowe,S.M., and Blake, A., *Real–time traffic monitoring*, in WACV94, pp. 81–88, 1994.
4. Genesereth, M.R., *Software agents*, In: Communication of the ACM, 37(7):48-53, 1994.
5. Honavar, V. , *Symbolic Artificial Intelligence and Numeric Artificial Neural Networks: Toward a Resolution of the Dichotomy*, Invited chapter In Sun, R. and Bookman, L. (Eds.), Computational Architectures Integrating Symbolic and Neural Processes, Kluwer, New York, pp. 351-388, 1994.
6. S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Engelwood Cliffs, Nj, 1995.
7. Woolridge, M., and Jennings, N.R., *Intelligent agents: Theory and practice*, Knowledge Engineering Review, 10(2):115-152, 1995.
8. D. Sobajic, *Applications of neural networks in environment, energy and health*, Progress in Neural Processing 5, World Scientific Publishing Co. Ptd. Ltd., P. O. Box 128, Farrer Road, Singapore 912805 chapter 11, 1996
9. Ferber, J., *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*, Harlow: Addison Wesley Longman, 1999.
10. Georgeff, M., Pell, B., Pollack, M., Tambe, M., and Wooldridge, M., *The Belief-Desire-Intention Model of Agency*, In Proceedings of The 5th Int. Workshop on Intelligent Agents, 1999.
11. S. Brossette et al., *A data mining system for infection control surveillance*, Technical report, Department of Pathology, University of Alabama at Birmingham, USA, 2000.
12. Craciun, F., Kope, Z., Letia, I.A., and Netin, A., *Distributed Diagnosis By Bdi Agents*, In Proceedings of the IASTED International Conference APPLIED INFORMATICS February 14-17, Innsbruck, Austria, 2000.
13. Jennings, N.R., *On agent-based software egineering*, Artificial Intelligence 117 (2000), pp. 277-296, Elsevier Press, April, 2000.
14. D. Lyon, *Surveillance Society: Monitoring Everyday Life*, Open University Press, 2001.
15. V.K. Prasad, *What Pervasive Computing Brings to Automotive Consumer Experiences*, Presented at PC2001 NIST, Gaithersberg, Maryland, available at `http://www.nist.gov/pc2001/agenda.html,2001`.
16. S. Bandini et al., *Knowledge-Based Alarm Correlation in Traffic Monitoring and Control*, Proceedings of the IEEE 5th International Conference on Intelligent Transportation Systems (ITSC02), Singapore, 2002.
17. M. Egmont et al, *Image processing with neural networks: a review*, Pattern Recognition, 35(10), pp 2279-2301, 2002.

18. D. Heffernan and G. Leen, *Expanding Automotive Electronic Systems*, In IEEE Computer, Vol. 35, No 1, pp 48-53, 2002.
19. R. G. Herrtwich, *Ubiquitous Computing in the Automotive Domain*, In F. Mattern and M. Naghshineh (eds.): Pervasive 2002, LNCS 2414, 2002.
20. S. Bandini et al, I*ntelligent Alarm Correlation*, In Proceedings of 2003 IEEE International Conference on Systems, Man and Cybernetics (Invited Session on "Modelling and control of transportation and traffic systems"), Washington DC, 2003.
21. M. Palmonari and F. Sartori, *Towards the Improvement of Monitoring and Control Agencies through Knowledge-Based Approaches*, Proceedings of WOA 2003, Villasimius, Italy, 2003.
22. M. Satyanarayanan, *Privacy: the achilles heel of pervasive computing?*, IEEE Pervasive Computing-Mobile and Ubiquitous Systems. Available at `http://www.computer.org/pervasive/pc2003/b1002.pdf, 2003`.
23. Chang-Hyun Jo, Guobin Chen, James Choi, *A new approach to the BDI agent-based modeling*, In Proceedings of the 2004 ACM Symposium on Applied Computing (SAC), Nicosia, Cyprus, 2004.

Evolution of Mobile Services: An Analysis of Current Architectures with Prospect to Future

Ivar Jørstad[1], Schahram Dustdar[2], and Do van Thanh[3]

[1] Norwegian University of Science and Technology, Dept. of Telematics,
O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
ivar@ongx.org
[2] Vienna University of Technology, Distributed Systems Group (DSG),
Information Systems Institute A-1040 Wien, Argentinierstrasse 8/184-1, Austria
dustdar@infosys.tuwien.ac.at
http://www.infosys.tuwien.ac.at/Staff/sd/
[3] Telenor R&D, Snarøyveien 30 N-1331 Fornebu, Norway
thanh-van.do@telenor.com
http://www.item.ntnu.no/~thanhvan

Abstract. With the advent of the Internet and the plurality and variety of fancy applications it brought with it, the demand for more advanced services on cellular phones is increasingly becoming urgent. Unfortunately, so far the introduction of new enabling technologies did not succeed in boosting new services. The adoption of Internet services has shown to be more difficult due to the difference between the Internet and the mobile telecommunication system. The goal of this paper is to examine the characteristics of the mobile system and to clarify the constraints that are imposed on existing mobile services. The paper will also investigate successively the enabling technologies and the improvements they brought. Most importantly, the paper will identify their limitations and capture the fundamental requirements for future mobile service architectures namely openness, separation of service logic and content, multi-domain services, personalisation, PAN-based services and collaborative services.

1 Introduction

With digitalisation, the difference between telecommunication and computer networking is fading and the same technologies are used in both fields. However, the convergence does not progress as rapidly as expected. Moving applications and services from one field to the other has proven to be very difficult or in many cases impossible. The explanation is that although the technologies in use are rather similar there are crucial differences in architecture and concepts. The paper starts with a study of how mobile services are implemented in mobile telecommunication systems and an identification of their limitations.

2 Analysis of Current Mobile Service Architectures

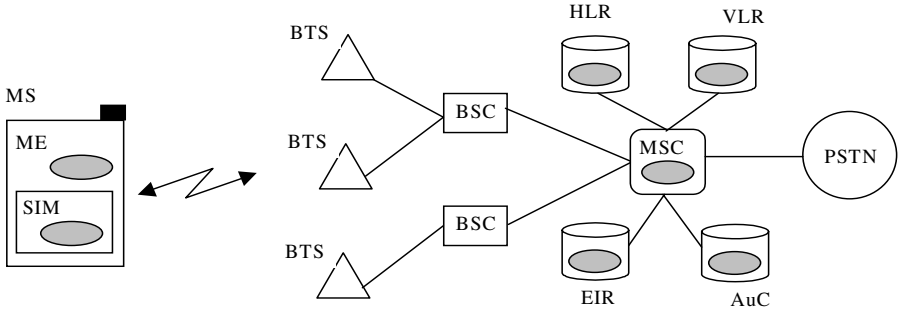
The following sections provide a walkthrough of mobile service architectures, where the GSM telecommunication system is used as a reference.

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 125–137, 2004.

2.1 Voice Communication

As indicated by its name, the objective of mobile telecommunications systems is to provide communication between mobile distant persons. In Europe, the Nordic Mobile Telephony (NMT) system became available in the 1980s. In 1982, development of the Global System for Mobile telecommunication (GSM) started. Initially, these systems only supported direct voice communication or telephony between two participants, but supplementary services like call forwarding, barring and voice mail were added later on.

Fig. 1 shows that the mobile telephony service is realised by components represented by grey ovals that are distributed both on the mobile phone, also called Mobile Station (MS), and on the mobile network. On the MS, there are components both on the Mobile Equipment (ME) and on the Subscriber Identity Module (SIM).



BTS: Base Station Transceiver – BSC: Base Station Controller – HLR: Home Location Register – MSC: Mobile service Switching Center - VLR: Visitor Location Register – EIR: Equipment Identity Register - AuC: Authentication Center – PSTN: Public Switched Telephony Network

Fig. 1. Telephony service components in mobile communications system

To establish a telephone conversation the service components on the MS are collaborating with the ones on the HLR (Home Location Register), VLR (Visitor Location Register), MSC (Mobile service Switching Center), EIR (Equipment Identity Equipment) and AuC (Authentication Center) to allocate a channel and to maintain it throughout the session even when the MS is moving and changing base stations.

Thanks to the clearly defined interfaces between them, the components can be designed and implemented by different parties. The components on the mobile phone are installed by the manufacturer while the ones on the network are delivered by network equipment suppliers.

As observed above, the telephony service is implemented in a very robust way and shall be operative 99,999% of the time. On the other hand, the architecture is also very rigid and does not allow the introduction of new services.

2.2 Supplementary Services with IN

It does not take long time before there is a need for more advanced call control services like call forwarding, barring, voice mail, premium call, etc. As shown in Fig. 2

an IN (Intelligent Network [1]) Service Control Point (SCP) is introduced in the mobile network to allow the implementation of supplementary services. It is worth mentioning that these services are derivatives centred around the voice communication service. Another restriction is that the SCP is implemented on equipment manufacturer proprietary technologies. The SCP is also located inside the telecom operator domain making third party service development difficult.

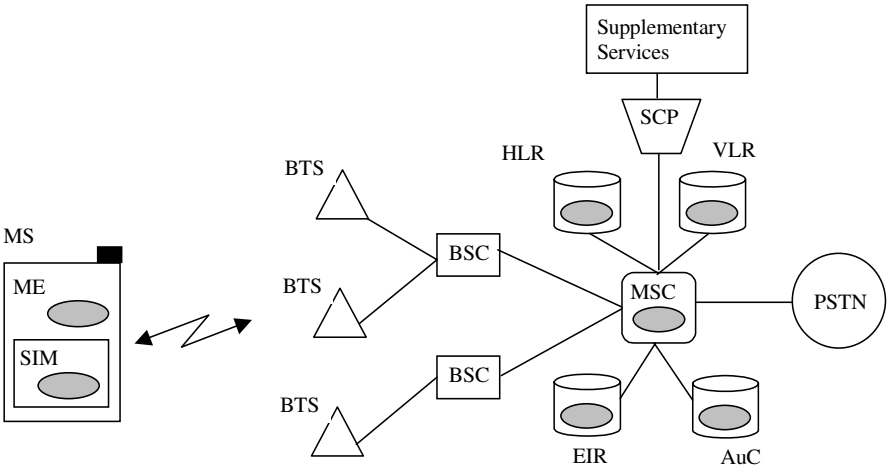


Fig. 2. The implementation of supplementary services

2.3 Enabling Services on the SIM with SIM Application Toolkit (SAT)

The telecom operators want to have other services than telephony and its derivatives and turn to the SIM, which are their property. Unfortunately, although the SIM is a smart card having both processing and storage capabilities necessary for new services, it is useless due to the lack of interfaces with input unit (keypad, microphone) and output units (display, loudspeaker). The SIM is supposed to be the slave executing orders from its master, the ME. To remedy this, the SIM Application Toolkit (SAT) [2] is introduced to allow applications/services residing on the SIM to control the input and output units.

With SAT it is possible to develop applications on the SIM but there are many restrictions (see Fig. 3). First, SAT applications should be small in size and developers must have access to SIM application development environment, which is both difficult and costly. Second, the installation of applications on the SIM is controlled by operators who are reluctant to open the access due to security. The results are that SAT applications are usually operator-owned and are typically security related since the SIM is a tamper-resistant device.

Recently, the JAVA SIM cards start to emerge and it will be very interesting to have collaboration between SIM JAVA components and JAVA components on the Mobile Equipment enabled by J2ME.

2.4 Text Services with Short Message Service (SMS)

Although voice communication is a big success there is still a demand for sending text messages from one mobile phone to another. An SMS client was introduced in the ME and responsible for sending short messages to the Short Message Service Center (SMS-C). The SMS-C is responsible to store and forward messages to and from mobile phone (see Fig. 3). In the illustration, components used for SMS are the client (C) in the ME, the SIM (for storage) and services connected to the SMS-C in the network. Note that in Fig. 3 the components of telephony services are indicated by a "T", SIM Application Toolkit by "SAT".

SMS substantially increased the value of mobile telecommunication systems by providing an alternative, more informal, way of communication between customers.

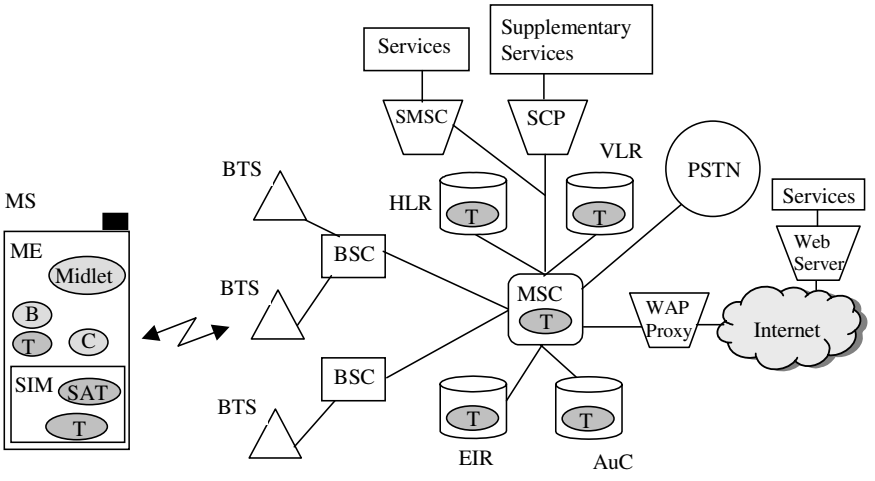


Fig. 3. Current mobile services including SAT applications, SMS-based services, WAP services and J2ME applications

Not long after its introduction, there is a need for SMS-based services that deliver or collect specialised content requested by users. Typical example is an application that sends SMS messages to key personnel when an emergency occurs. Another example is to use SMS for voting in TV shows or contest. Provisioning of SMS services requires installation of the above mentioned application on an SMS Gateway that either communicates directly with an SMSC using one or several protocols defined for this purpose (e.g. CIMD [3] or SMPP [4]). Another solution is for the system running the SMS Gateway to act as an SMSC itself (e.g. a PC using a radio modem through a serial port).

The development of SMS services is quite simple. Most services can actually be implemented as Perl scripts or with any other programming language capable of reading data from standard input and producing some output. However, the SMS services as indicated in their names are centered around the sending or receiving of SMS messages. The SMS environment does not provide adequate input and output capabilities to other data applications such as word processor, web content, database access, etc.

2.5 Internet Access with WAP

Wireless Application Protocol (WAP) [5] was intended to provide access to the World Wide Web on handheld terminals. The technology is often referred to as the Mobile Internet. In concordance to the World Wide Web, a micro browser installed in the Mobile Equipment is communicating with a WAP Proxy introduced between the Internet and the mobile network to convert Internet protocols to Wireless binary protocols as shown in Fig. 3. On the terminal side, a WAP browser is located in the ME (symbolised by an oval with a B inside) and services are connected to a Web server on the network side. WML (Wireless Markup Language) is an XML-based markup language tailored for handheld devices. However, in WAP 2.0 the main markup language is XHTMLMP, which extends the Basic profile of XHTML as defined by W3C [7]. The realisation of WAP 2.0 is radically changed from the previous standards, because a WAP proxy is no longer needed between the handset and the server; WAP 2.0 supports the standard Internet protocols, so a conversion between WAP protocols and Internet protocols is no longer needed.

Started with static WML content, WAP evolves rapidly to support dynamic content via WMLscripts that request processing on back-end server in the similar way as CGI (Common Gateway Interface)-script. There exists numerous WAP Software Development Kits (SDK) available for developers. Both Ericsson and Nokia provide their own solutions. The development of WAP services is simple. The only requirement for deploying and provisioning WAP services is a HTTP server that is accessible from the Internet and that supports the content types (MIME types [8]) required by WAP (most importantly text/vnd.wap.wml and text/vnd.wap.wmlscript).

One restriction of the technology is that it is not possible to access ordinary webpages using a WAP browser. Instead, services must be designed and implemented specifically for WAP. Although the new WAP 2.0 standard can render documents written in XHTML Basic, many devices do not yet support the WAP 2.0 standard and most services on the WWW do not conform to the XHTML Basic language. Thus, providing services to WAP almost always requires a re-implementation of already existing services.

The biggest limitation, however, lies in the fact that WAP does only enable limited Web browsing. Indeed, browsing for a particular service of interest is not satisfactory way to access services on a small, handheld terminal. A more definite and intelligent approach is needed; services on mobile terminals need to be more “at-hand” than on stationary computers. This difference in requirements for accessibility follows naturally from the different contexts the two terminals are used in; mobile terminals are often used while in transit between two locations, thus minimal attention should be required for using the services. One solution might be to provide better search engines designed for WAP services, potentially exploiting emerging Semantic Web [9] technology.

2.6 Dynamic Applications on Mobile Phones with J2ME (CLDC/MIDP)

Till now, the functionality of the mobile phones is defined at manufacture time and it is not possible to install new applications. Indeed, the mobile phone is a terminal, i.e. a device terminating the network system and not a computer allowing the selection and installation of applications. It is desirable that mobile phones evolve to be closer to computers but at the same time retain the reliability and robustness of terminals. With introduction of the J2ME CLDC/MIDP platform, development of unlimited types of

applications has become possible. A vast amount of such applications, called MIDlets, can be found on the Internet, some for free and some to be purchased. In Fig. 3, these applications are represented in the ME as an oval with the text "Midlet" inside.

J2ME CLDC/MIDP [10] is a runtime environment for small footprint Java applications, targeted at handheld terminals with limited resources (processing capacity, memory etc.). With J2ME, it is possible to develop dynamic standalone applications, but also clients that are part of a larger distributed system. This means that most of the services provided by the earlier discussed platforms, in theory, can be developed using J2ME as well, provided there exists resources for the developer to exploit; most importantly implementations of standard communication protocols (e.g. TCP, HTTP, SOAP).

A WAP client, or any other type of browser, can potentially be developed in J2ME. Also, specific services available through WAP can be developed on a per service basis, often with a more flexible user interface. These services can be developed using the HTTP protocol, which is mandatory in any implementation of the J2ME CLDC/MIDP 1.0 platform.

Unfortunately, J2ME, as the previously mentioned technology enablers, does have severe limitations. First, only rather simple and stand-alone applications can be built because of the absence of standardised APIs for telecom functions like call control and SMS, for extended support for communication protocols (TCP, SOAP) and for accessing components integrated in the handset like Bluetooth, camera or storage. For SMS for example, the Wireless Messaging API (WMA) [11] has quite deceiving limitations. Access to the standard inbox for SMS messages on a handset is not allowed, although this is actually the really interesting part with an SMS API. The WMA instead provides an API for sending specialised SMS messages between J2ME MIDlets. With an SMS API providing access to the SMS inbox of a handset, it would be possible to provide services like:

- Incoming SMS filtering (spam filter)
- Incoming SMS forwarding (e.g. to an email address or other storage)
- SMS Auto reply (e.g. away messages)

Another drawback is that the Over-The-Air (OTA) provisioning of J2ME MIDlets was not a part of the MIDP 1.0 specification. A supplementary document describing a "Recommended practice" was released after the standard was completed. The MIDP 2.0 standard includes an updated version of this document. Most likely because of the late arrival of this OTA specification, handset manufacturers defined their own ways of performing OTA, thus making it more difficult to implement a standard J2ME OTA server supporting any J2ME MIDP 1.0 compliant device.

A major limitation is the incompatibility issue. Although J2ME is a standardised technology, performed through the Java Community Process, the "write-once-run-anywhere" concept is not valid for this platform. Some terminal manufacturers have adopted the platform and added their own touch to it, especially in the presentation layer (Graphical User Interface, GUI). Motorola has developed its own library for GUI construction called the Lightweight Windowing Toolkit (LWT) which is distributed for free. Although enhancing the potential of J2ME, it ruins the concept behind, namely that the same services should be available using any J2ME CLDC/MIDP compliant device.

3 Future Mobile Service Requirements

As discussed in earlier sections, the enabling technologies aiming at promoting mobile data services do have specific focuses and limitations. The IN architecture only enables construction of supplementary services around telephony. The SMS technology focuses only on the delivery of messages from and to mobile phones. The WAP technology provides limited Web browsing. J2ME is intended to enable dynamic functions and capabilities for mobile phones but has too limited APIs and non-standardised OTA downloading of applications. Furthermore, these technologies are either fragmented or overlapping and put together they are not sufficient to fill all the holes in the picture of the future mobile service arena. This section aims at identifying and elucidating these missing pieces and hence contributes to the definition of future mobile service architectures and concepts.

3.1 Service Openness

Many unresolved issues impacting the future of mobile services, are related to openness of specific systems. Today, the chain of activities between developer, provider and consumer are still characterized by a closed system where the terminal manufacturers and telecom operators are controlling everything and focusing on not loosing control of the value-chain. A potential growth of services, as well as revenue generated by them, might be realised by allowing anyone to take on any of the roles behind the four specified activities (see Fig. 4).

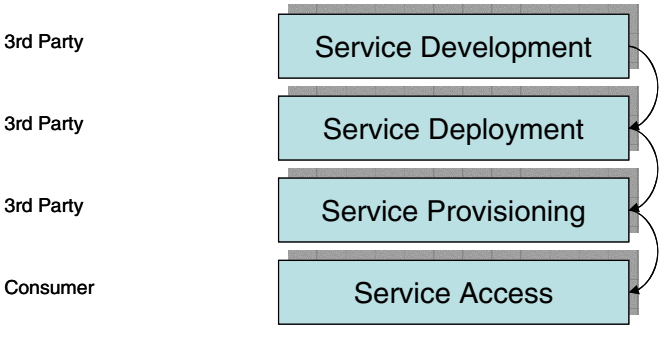


Fig. 4. Openness in all activities

The activity which is currently most open is the development of services. Both operators and 3rd party can today successfully create some types of new services. What is most important to further support this in the future, is that APIs of the service platforms are open, specifications of them are readily available and good Software Development Kits (SDK) are freely available.

Some service platforms do not require a developer to cooperate with operators to deploy services (e.g. WAP and J2ME) and some do (e.g. SMS). Installation of WAP and J2ME applications can be performed by anyone. SMS requires specific agreements with operators to access SMS-Cs through TCP or X.25 connections.

It is important that future service platforms are as open as possible for each of the mentioned activities. Still, they should include mechanisms for receiving revenue

from service usage. It is important that security and privacy of the users are preserved at the same time as the mobile phone continues to be a user-friendly, robust and reliable device. It is difficult to reach compromise for this matter and further studies should be carried out.

3.2 Separation of Service Content and Logic

Mobility is the ultimate requirement for mobile services; they should by definition be available at any time, any place using any device with communication capabilities. The mobility properties of a service are dependent on the architecture used to realise the service, and particularly on the *location* of the components making up the service. Considering a service as consisting of two components, *service logic* and *service content*, makes the analysis easier (see Fig. 5).

The primary question is where the service logic should be located. In early mobile telecom services (Voice telephony and SMS), we have seen that the service logic was embedded in the dedicated hardware components of the system. This has been a hindrance for development of flexible services, but more importantly, it means that these services will by default not be accessible from outside an operator domain. Although interoperability between operators has been achieved by roaming features of the mobile telecom systems, the services are still only accessible from specialised terminals, i.e., cellular phones with a particular radio interface (e.g. GSM). Mobility means also that a service should be accessible by any device. To enhance the mobility of services, it is necessary to decouple the service logic from the system components.

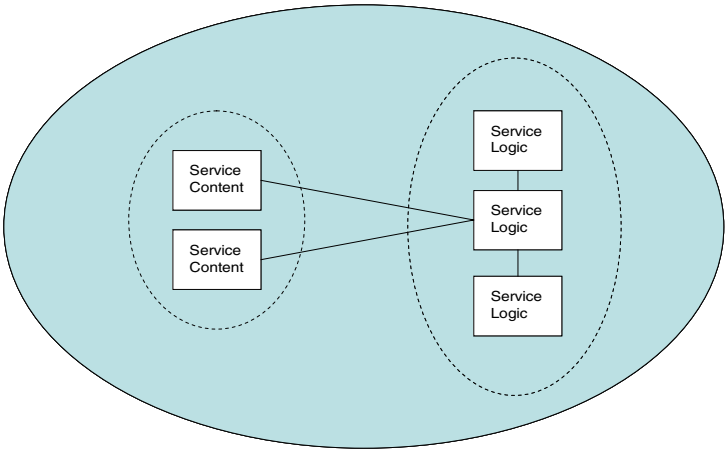


Fig. 5. Generic service composition

Second, users will have an increasing need for accessing content that is related to a service; content that is a product of earlier service usage (e.g. content of an address book or a document created in a word processor). It is thus not enough that the service logic is accessible from any device, on any network. Content generated by the user must be accessible also. This raises the question of where this content should be located, if it should be replicated throughout service domains and networks or if it

should be centralised in the home network of a user. Probably the easiest illustration of this challenge is the bookmark list a user keeps in an Internet browser. Today, the browser on each terminal keeps its own bookmark list, instead of providing access to the same content everywhere. This suggests that future service platforms must also cope with accessibility of service content and not only of the service logic. Alternatively, the service content can be incorporated in the user profile.

3.3 Multi-domain Services

As mentioned earlier all services initially provided by mobile telecom operators were located in the telecom network itself, and developed by telecom operators. With SMS and WAP, many services were moved to a third party service provider, outside the operator network domain, providing access through CPA agreements or similar. Also solutions for accessing enterprise network domains and their services have been introduced (for example Microsoft Exchange with Mobile Features).

A complete service should embrace service elements belonging to all the domains as shown in Fig. 6. The concept of a Virtual Home Environment proposed by 3GPP for third generation mobile systems allows a user to roam to foreign networks while keeping access to all services that they subscribe to in the home network. Unfortunately, the availability is restricted only to mobile telecom services. Taking this concept further, the ultimate goal would be to provide ubiquitous access to data services as well no matter where the service content and service logic are located.

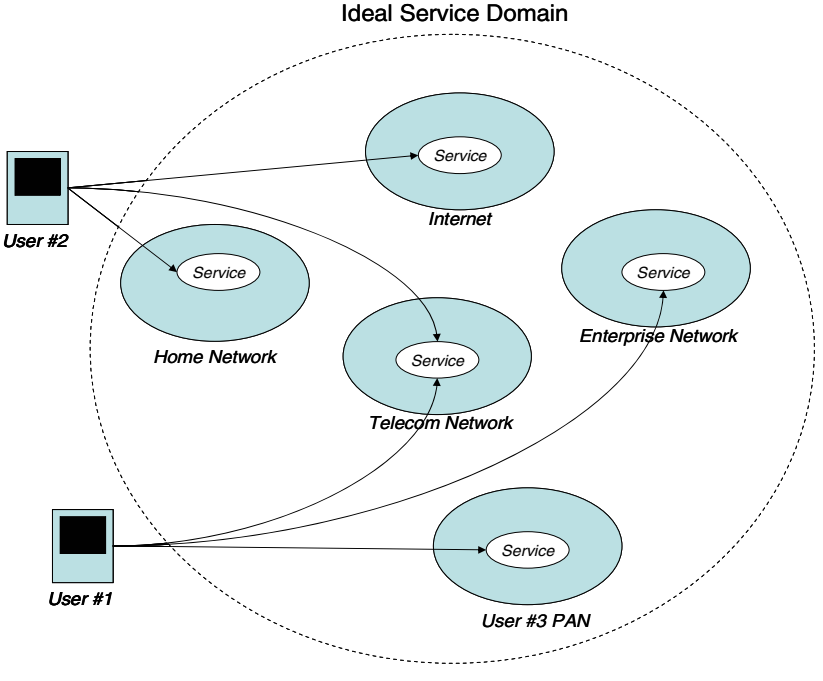


Fig. 6. An ideal composite service spanning over several domains

As shown in Fig. 6, one important domain is the Home Network of the user. In fact, more and more people get connection to the Internet at home through a permanent, broadband connection (xDSL, Cable or similar). Some of these have their own LAN at home. This home network constitutes an important host for personalised services. To offer a more complete service to the consumers, it is important to investigate solutions providing access to services located in the home network.

Ultimately, mobile services will be provided as distributed services where the logics residing in different places will cooperate in delivering the end-user service; i.e. composite services of disjoint service domains.

3.4 Personalisation of Services

Personalisation has for many years been regarded as an important feature of mobile services. However, a clear picture of what personalisation means, and of exactly what should be subject to personalisation, is lacking. A psychology study [12] defines two categories of motivation for personalisation:

- To facilitate work
- To accommodate social requirements

Until now, most personalisation efforts of mobile services have been to accommodate social requirements, whereas a big potential exists for services of the first category. Personalisation can be performed at several levels:

1. Individual Service Personalisation
2. Personalisation of Service Portfolio
3. Personalised Service Composition

Individual service personalisation is the common way of thinking personalisation. It means changing parameters of a single service to accommodate a specific users needs. Such parameters can be either *passive* (look and feel of the service), *active* (behavior of the service) or *content related* (user added content, result of service usage). The user-defined values of the personalisation parameters are captured in the user profile that can be distributed or replicated in all the five domains: Telecom Networks, Internet, Enterprise, Home and PAN. Little work has been done on the distribution and organisation of the user profile and they will be subject to future studies.

Personalisation of service portfolio means that a user makes certain services easily available through a portal or menu system. Such personalisation is done on most PCs, where different users have different services/applications available through the desktop environment (or Quick Launch Bar in MS Windows).

Personalised service composition means that users can “create” their own services by picking discrete service components and combining them into a complete service. This is a novel concept that needs further investigation.

3.5 PAN-Based Services

Nowadays, each individual is using several devices like mobile phones, PDAs, digital camera, GPS, etc. that are autonomous and functioning independently of each other and without any coordination. In fact they are not even aware of the presence of other

devices. As the owner the user is required to handle them all and does not always succeed since as a human being he cannot perform many tasks at the same time. With the emergence of wireless short-range technologies like Bluetooth, WLAN and potentially UWB (Ultra Wide Band), Personal Area Networks can be formed to allow communications between devices. The Virtual Device [13] is a novel concept, which considers all the autonomous devices on the user 's Personal Area Network as one big "Virtual Device" having multiple input and output units and providing a coherent and surround interface to the user. The Virtual Device concept paves the way for innovative and exciting services. An example is shown in Fig. 7.

The User #1's Virtual Device consists of a cellular phone, a digital camera and a GPS receiver. Since it has multiple inputs and outputs multiple services can be executed simultaneously. The first one may supply User #1's location information to User #2 and the second one may show real-time pictures for User #3.

Every PAN becomes a service domain and potential host for service logic and service content as discussed in section 3.3.

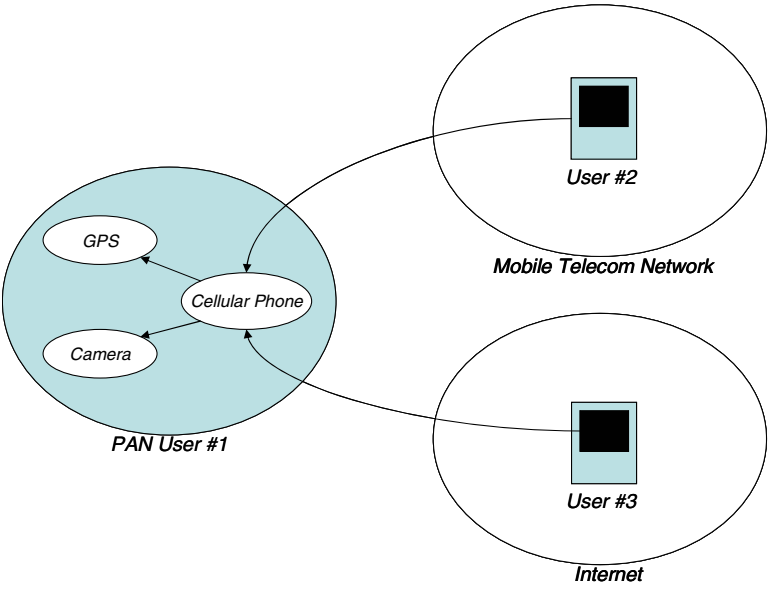


Fig. 7. Providing access to resources in a Personal Area Network

3.6 Collaborative Services

The concept described in the previous section and illustrated in Fig. 7 is particularly important for collaborative systems. With a multi-domain service, it will be possible for people not only to collaborate across network boundaries, but also across terminal boundaries. An employee sitting at home can for example work on the same document that other employees are working on at the office, while another employee on travel can work on the same document using his handheld terminal, be it a GSM cellular phone or a PDA connected through WLAN.

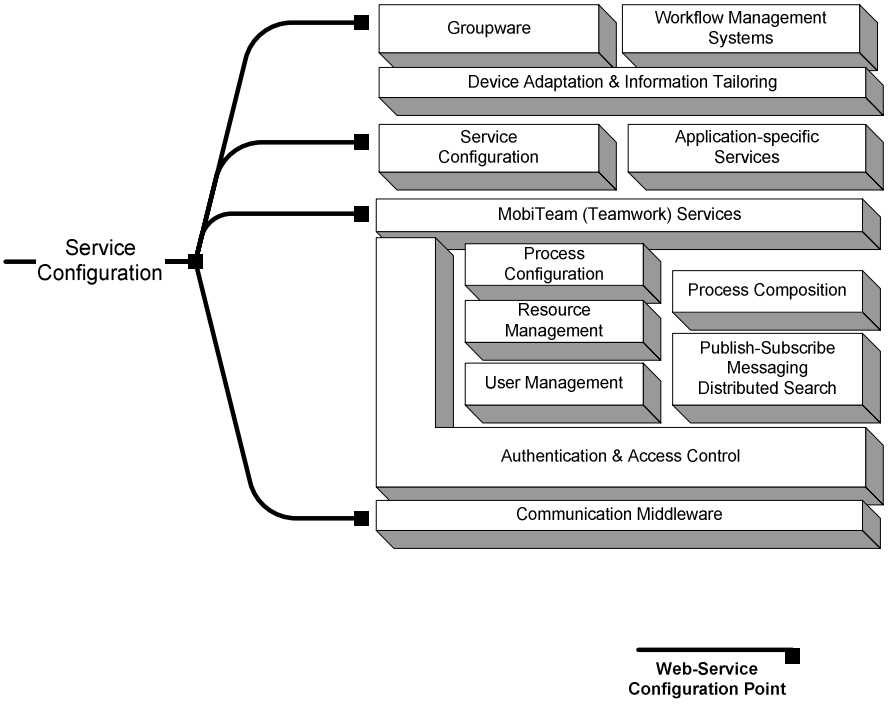


Fig. 8. Web services Management and Configuration Architecture

This will not be restricted to documents, but to any resource that is sharable through communication technology (e.g. Bluetooth enabled devices). It should also be possible for several people to collaborate by exchanging information through several channels and devices simultaneously such as talking on the phones, showing picture on digital cameras, reading documents on PDAs, etc. In [14] we argue that Web services are a suitable technology for achieving service interoperability and suggest an architecture including service configuration points. The layered architecture depicted in Fig. 8 suggests new layers of abstractions for management and configuration of services as well as their compositions.

4 Conclusion

This paper presents an analysis of the evolutionary path of mobile services, from early voice communication services to prospects of future service possibilities. As a result of this analysis, some important concepts of mobile services are identified and further discussed. Activities related to mobile services, and their relative *openness* with regards to actors is discussed. It is argued that increasing this openness can help excel the future of mobile services. General mobility requirements are discussed and put in context of openness, as well as the distribution of the service logic and content. Further elaborating these issues, the various hosting domains for services are discussed.

Personal networks at home and PANs are identified as two important domains for containing service logic and content for future mobile services. Employing Personal Area Networks as a mobile service domain is briefly considered in particular, since this is the most novel of the two recently identified service domains. It is recognized that the future will need generic redirection mechanisms (of input/output) between service components to enhance availability of services and exploit resources in various devices to the fullest.

Personalisation is presented as a more complex construction than hitherto acknowledged, as it initially can be divided into at least three separate levels. Each of these levels can be further elaborated and sub-divided, but only an initial attempt at this is done in this paper; the rest is left for future work. Last, the evolution of mobile services and technologies supporting them are put into the context of collaborative services.

Each of the concepts discussed around mobile services in this paper are on their own advanced and extensive fields of research and they must be further elaborated in separate studies. Thus, the discussions in this paper are preliminary and do address only the basic structures and further works will be carried out.

References

1. Heine Gunnar, *GSM Networks: Protocols, Terminology and Implementation*, ISBN 0-8900-6471-7, January 1999
2. ETSI, *Digital cellular telecommunications system (Phase 2+);Specification of the SIM Application Toolkit (SAT) for the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface (3GPP TS 11.14 version 8.15.0 Release 1999)*, 2003
3. Nokia, *CIMD Interface Specification*, 2002
4. Aldiscon Ltd., *Short Message Peer to Peer (SMPP) Interface Specification*, 1996
5. Open Mobile Alliance, *WAP 2.0 Technical White Paper*, January 2002, `http://www.wap-forum.org/what/WAPWhite_Paper1.pdf`
6. Vineet Rajosi Sharma, Indian Institute of Technology, *Study Of Wireless Application Protocol*, 1999, `http://www.iitd.ac.in/~rajosi/mini/`
7. W3C, *XHTML Basic, W3C Recommendation*, December 2000, `http://www.w3.org-/TR/xhtml-basic/xhtml-basic.pdf`
8. Internet Assigned Numbers Authority, *MIME Media Types*, `http://www.iana.org/assignments/media-types/`
9. W3C, *Semantic Web*, `http://www.w3.org/2001/sw/`
10. Java Community Process, *JSR 37: Mobile Information Device Profile 1.0 Final Release*, 2000
11. Java Community Process, *JSR 120: Wireless Messaging API Maintenance Draft Review*, January 2003
12. Blom Jan, *Why do we personalise?*, Department of Psychology, University of York,
13. Do, van Thanh, Jønvik Tore, Vanem Erik, Tran, Dao van & Audestad, J.A.: The Device Management Service, Proceedings of The IEEE Intelligent Network Workshop 2001 (IN2001), Boston, USA, ISBN 0-7803-7047-3, May 6-9, 2001.
14. Dustdar S., Gall H., Schmidt R. (2004). Web Services For Groupware in Distributed and Mobile Collaboration. 12th IEEE Euromicro Conference on Parallel, Distributed and Network based Processing (PDP 2004), La Coruña - Spain, February, 11-13, 2004, IEEE Computer Society Press.

Collaborative Design of Web Service Networks in a Multilingual User Community

Kurt Englmeier[1] and Marios Angelides[2]

[1] Fachhochschule Schmalkalden, Fachbereich Informatik, Germany
KurtEnglmeier@computer.org
[2] Brunel University, Department of Information Systems and Computing
Uxbridge, Middlesex, UB8 3PH, UK
angelidesm@acm.org

Abstract. This paper presents the Web Service (WS)-Talk interface Layer, a structured natural language interface for the inter-service communication that extends the "find, bind, and execute" paradigm of the web service interaction. This "open building block" can be implemented by both the service designers who as providers are more concerned with the architecture of the underlying service model and the service requesters who as users will seek to specify web services as solutions to specific problems. Through a semantic layer, WS-Talk transforms service descriptions or requests which have been expressed in natural language into task-specific web serviced specifications. Whilst the objective of bringing together the service providers with relevant task-competent end-users in the architectural design of web service applications is, on the one hand, to build connected interoperable applications, on the other hand, the WS-Talk Layer enables service requesters and providers to design and implement ad-hoc new services or fine-tune existing ones.

1 Introduction

Web services is the second generation of Internet tools to connect people to things they are dealing with. They are not connecting people with HTML web pages; they are connecting their business applications with those of their colleagues, customers, partners and suppliers. By employing web services people expect to respond quickly to customer demands and to capitalize on new market opportunities. Web services could in fact revolutionize the way we develop applications like the Internet itself changed our life [1]. And the rapid adoption of web services is spurred on by the benefits they bring to collaborative communities. Connecting, however, exploits human relationships; service designers and service requesters (or consumers) working as a community define the architecture of their web services collaboratively but each for their own benefit. Technologies used to set up and maintain web services, help such communities create small, task-specific applications with software modules that can be used and re-used.

Standards are undoubtedly critically important for web service technologies. They are a pre-requisite for interoperability. Users want their web services to link and interact with those of their partners and colleagues in a standardized way, yet, personalized to their needs and preferences. The launch of XML opened avenues for a completely new type of interoperability of software across networks. The desire to use each other's applications in order to develop new ad-hoc services and appliances

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 138–150, 2004.

seemed to be within reach. In the meantime the sobering truth about semantic web standards is that they are not the silver bullet they were once conceived to be. Even though they support interoperability, developing large and complex domain-specific applications is still incredibly complicated. Once implementation commences, it is often difficult to retract back to the specification in order to correct problems [2].

Semantic web standards are very helpful and without them interoperability would not reach high levels. The context of web services comprises of a stable framework relying on semantic web standards. This framework constitutes the basic principles of the service network structure. The network itself, however, comprises of a lot of details specified by different people and exposed to frequent changes over time. Users should be involved on the basis of their past experience and competence in the specification of service identification and communication and the specification of natural-language processing. The definition of resources [3] benefit more from the direct involvement of all those involved. At the organizational and market levels it is easier to reach an agreement on a suitable vocabulary than to design complicated XML models that can not be understood by users.

The paper is organised as follows: section 2 outlines the rationale and advantages of extending the design of the conceptual service model towards natural language (NL). Section 3 explains how natural language can be applied in service lookup and discovery. It introduces the concept of "enterprise talk" that reflects a certain domain, the organisational business or a market sector, for instance, including its respective tasks, processes, and operations. Section 4 describes how localisation of web services takes place in a semantic context. Design details of WS-Talk enriched with examples from its first prototypical realisation are presented in section 5. Section 6 concludes.

2 Conceptual Service Model and Participatory Design Benefits: Two Scenarios

The conceptual business model or the business architecture [4] is a formal description of business processes established in a certain domain. Typically it is derived from a set of use cases that illustrate processes as they appear in domains such as production, customer relationship management or market analysis, just to name a few. A simple example of such a process could be the on-line reservation system of a cinema which includes the selection of a particular performance of a movie and of the number of required seats, and confirming the reservation back to the filmgoers. In this example a reservation manager co-operates with a performance manager that knows about the different movies and the dates they are shown and with a room manager that knows about the cinema's rooms and their capacities and availability of seats. The different managers and their interaction schema are defined in the conceptual business model or in the conceptual service model if the cinema resorts to web service for the realisation of its reservation system.

The conceptual model usually refers only to the organisational schema of the process and not to the underlying technology. The conceptual service model then maps the organisational schema of (business) processes into an architecture of web services and, that is, it composes adequately web services to realise the required processes [5]. The whole model is typically broken down into an hierarchical structure of sub-models on different levels of granularity. The different web services emerge from their corresponding sub-model and are orchestrated according to the

conceptual model's layout. Throughout this paper we concentrate on the design of the conceptual model and how natural language could enhance the efficiency in designing the model and its mapping features. Designing the conceptual model is realised under the assumption that the domain is completely known *a priori*. "Completely" means both in this case: the designers creating an application have a comprehensive understanding of the business or of the problem domain and they have to know how the problem domain may evolve into the future. Neither is usually the case.

Let us consider for a moment a scenario that reflects the use of an information retrieval environment that emerges from a network of web services: During a meeting dealing with the financial situation of a company one colleague needs some figures on incoming orders and capacities in the chemical industry on all of the company's regional markets. She voices her query into her mobile phone. Like a meeting companion the interface of the mobile passes her query to the remote retrieval system with further keywords retrieved from the most recent meeting. The interface of the retrieval system recognises the colleague's actual working environment and presents the required figures on her hand-held device. The example scenario may sound a bit futuristic, but it is not too far-fetched if we expect that web service technology will spur the development of a series of new applications with those future retrieval environments among them. The processes of this example emerge from an ad-hoc composition of various communities of web service. Among the most important components of this service network is the "meeting companion" that resorts to speech recognition services (which may also be used in a Customer Relationship Management System), a document management system which records minutes, and a retrieval system that supports economic analysis (which may include retrieving newspaper articles, internal company reports, and relevant economic time series which, in addition, are processed using adequate statistical methods).

For the time being the development of the conceptual service model is by large in the hands of service designers and providers. The very last details of the model, however, are quite often unknown to the designers. The end users, the service requesters, for instance, are not always in the position to explain all the facets of their tasks including those that are supposed to be represented in the conceptual model. This phenomenon is not new. It has been well-known in application development for decades. In addition, the future evolution of the application can hardly be foreseen comprehensively in advance. Both factors provoke a continuous re-design process to specify the very last details of the model. The further detached the development gets from the transport layer, the basic design of the services and the communication between them, and the more attached it gets to the application specifics, the more it traps designers and providers in the specification problem.

The conceptual service models for the reservation system and for the information ambience are of different complexity and dynamics. Environment stability and reservation system transparency favour a reliable and durable model. XML interfaces and service design will be valid and probably untouched for a long period. Things look different in the second scenario. If we imagine for a moment the information ambience and its processes in all its ramifications we can easily discern the complexity of the model we have to develop. In addition, it is quite likely that the model requires a lot of adaptation to situations we have neglected or those that have recently come about. We think it could be an advantage or eventually a necessity to pass a part of this modelling task to the service requesters that have in the end greater competence in addressing their working environment and thus greater competence in

breaking down adequately a specific task into smaller pieces. This approach is well within the spirit of Participatory Design [6] that advocates a strong involvement of end users in application design which helps to reveal important but often tacit issues of domain knowledge. Of course, the user's competence ends when design comes closer to specifications of fine-grained services (like processing of time series) and interfaces. Fine-grained models are the area of service designers that develop and agree on XML structures for the service interfaces and implement these services.

3 The Natural Language Component of Web Service Lookup and Discovery

In principle, a user contacts a registry service for getting information about a type of service he or she wishes to use. The registry can be a service at the premises of the user or an external third-party registry. It returns all services available that matches the criteria stated by the user. In a standard situation as shown in figure 1 the user just chooses which service to use, binds it over a transport, and executes its methods based on the description of the service provided by the registry [5]. In a web service environment for the cinema, the system as service requester leases ad hoc the different managers from service providers. The choice among competing providers is based on service quality and price. The data to be transported, such as name and telephone number of the filmgoer, seat lay-out of the room, and information about the performance are wrapped into (i.e. represented by) XML which describes structure and "meaning" of data. This wrapping process is typically defined in the transport layer of the web services. A service requester (or service consumer) requiring a particular service describes this service and retrieves (i.e. "finds") useful services using this description. Typically the requester sends a query containing exactly this description to a registry that – based on this query description – searches for the service most appropriate to fulfil the requester's needs (in terms of quality and costs of use). Using a pointer from the registry the requester "binds" to the provider of the retrieved service. The service description contains all the necessary instructions for executing the service and handling its interfaces. The requester then formats its instructions following the descriptions of the service, binds these instructions to a certain type of transport medium used by the underlying transport layer and sends the service the wrapped instructions over the transport. On the same way the requester gets a reply from the execution of the service.

An application is typically composed of a number of services where each service can be made up of a number of further services. Hence, the situation depicted in figure 1 is simplified because in reality a service requester is a hierarchy of different services that represent the conceptual service model at different levels of granularity. The initial service requester may be a user that describes the service she or he wishes to use. The registry then decomposes the request in accordance with the service model. This process is repeated top-down to the lowest level of granularity. The service requester may also use a service proxy as shown in figure 2.

The proxy knows about the conceptual service model and is therefore in the position to compose correctly the services into the required application. It resides at the user's site which may be more convenient as the users may wish not expose organizational details of a company to the outside world, details that are well-described in the conceptual model.

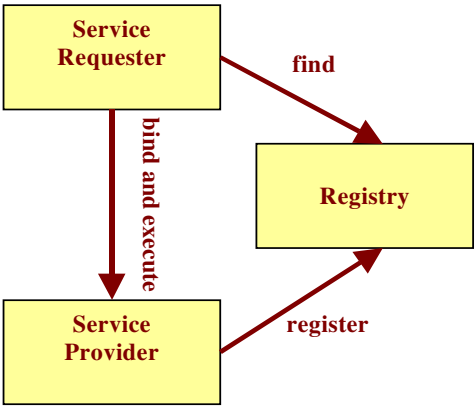


Fig. 1. The "find-bind-execute" paradigm in a standard situation of a service request

We advocate now that the service proxy (or the registry) should have automatic text analysis capabilities. This enables the user to define in a flexible way and on a high level of granularity an application and eventually its composition out of a number of services on to the next lower level of granularity. This means the composition logic of the application is wrapped into natural language. For decomposition purposes the proxy analyses the request (query description) and transforms it into standard descriptions for service requests. The objective of the WS-Talk wrapper is exactly to endow the proxy with automatic text analysis capabilities.

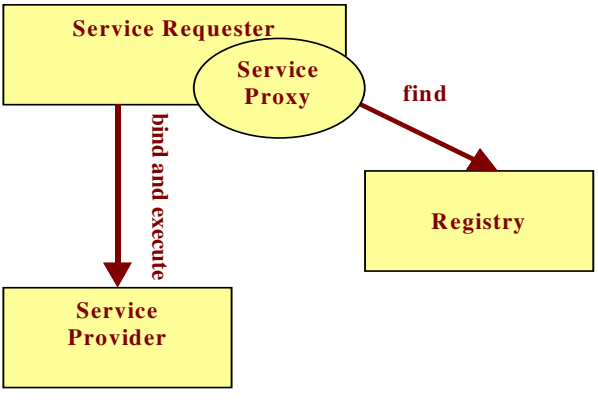


Fig. 2. A user proxy service orchestrates service composition and find-bind-execute processes

This natural language capability puts the end users in the position to express in their own language the application they require. Each company, and each industrial sector, has its specific domain talk used to describe products, services, objectives, processes, tasks, and so on. First of all we distil from this domain talk a structured representation and take the resulting condensed enterprise talk as quasi-standard for

inter-application communication. Quite often domain talk is represented in taxonomies. More and more organizations are making a comprehensive taxonomy of the organization's content a high priority. Taxonomies as well as controlled vocabularies organize content and context of the organization's subject by grouping similar items into broad categories which themselves can be grouped onto ever-broader concept hierarchies. In essence, taxonomies provide a degree of structure to the organization's unstructured content. Because of a taxonomy reflecting the most important business categories, organizations that carry out the same business activities tend to have similar taxonomies. This fact and the rising importance of taxonomies to organizations brought us to the idea to harness this structured information as quasi-standard for the composition of a web service network applied within an organization's scope, and beyond if similar taxonomies allow this expansion.

The WS-Talk service then consists of an interpreter residing at the service proxy (or registry), and a taxonomy maintenance tool for capturing the enterprise talk. Through the structured-language interface of WS-Talk, web services can be implemented by both the software developer and technology end users who operate on the natural language interface. The objectives and advantages of this wrapper are:

1. A high availability for ad-hoc solutions for small technology user communities,
2. A high flexibility in responding to the dynamics of the community's environment to which it is applied,
3. Fostering the proliferation of web services as flexible means to construct complex and dynamic applications, and
4. Passing on the design to technology users who are aware of the application area rather than technical people.

The objective of representing a domain context in natural language is to enable a flexible and easy management of enterprise "standards" necessary for domain-specific web service composition. Such inner-enterprise "standards" have their origin in the enterprise talk, in its traditional way to represent business processes, products and resources. Sometimes the enterprise talk is reflected down to the labeling of functions and parameters in their software codes. Enterprise talk includes also descriptions of products and services. For external observers, it is sometimes hard to catch up on a discussion by car manufacturers, while the latter have problems to follow food producers, and so on. The issue becomes worse when it comes in combination with production details or IT-details.

Proprietary message-oriented middleware is currently the backbone of enterprise interoperability. Its interfaces and protocols reflect IT-specifics within the enterprise talk. The enormous amount of specific detail makes creating even intra-enterprise standards extremely complicated using a high level specification of a web service standard. And in turn, this fact sheds light on the extreme complexity of establishing industry-wide standards. The situation looks equally grim on the side of industry-wide specifications. The problem of standardized specifications in capturing the complexity of the enterprise's reality is a severe technical limitation that hampers the broad proliferation of a technology that in fact could revolutionize our way of building applications. Facilitating the development of an architecture of context through sophisticated design tools eventually mitigates this problem, but cannot tackle it completely.

The reflection on this enterprise reality brought us to the idea of taking the enterprise talk as quasi-standard for inter-application communication. However, it is obvious that the approach can be applied as well in a context that is different from

that of an enterprise. The definition of an open building block then consists of a transport layer representing a typical web service interface and a context layer resorting to WS-Talk features for capturing the enterprise talk. The possibility to express a service description in natural language offers enormous flexibility.

4 Localising Services in a Semantic Context

Natural language (NL) itself is not appropriate for programming, but NL referentiality is a key factor for the design of program instructions that are both machine-processable and understandable to humans[7]. For information retrieval and speech recognition systems [8] NL referentiality is essential and aspect-oriented programming shows how powerful program organisations can be realised based on instructions and structures expressed in NL [9], [10], [11]. In a web service environment, services are propagated and identified through descriptions. To avoid ambiguity the descriptions have to be mapped correctly into a coherent context map that allows, at the same time, to locate the required service. Semantic co-ordinates – i.e. controlled vocabularies derived from taxonomies and structured according to concept hierarchies – are elements of orientation that can be communicated. Concept hierarchies representing semantic co-ordinates enable to develop a context map for the respective application domain. And, in addition, these hierarchies can be mapped into different natural languages in parallel [12]. To ensure that the descriptions are machine-processable (i.e. are interpreted correctly) we apply robust text mining methods that map descriptions into a suitable controlled vocabulary. From a different angle our solution resembles the current successful application of natural language processing in speech recognition systems where command languages are combined with NLP interfaces to provide for robustness.

Localizing a service in the context of WS-Talk is a two-step process. First a linguistic interface layer ensures that the service proxy "understands" a phrase expressed in an ad-hoc way in natural language. "Understanding" means mapping an information item like a web service request into an appropriate request description that is developed by terms of the target language, the controlled vocabulary derived from taxonomies, and structured along concept hierarchies. The WS-Talk interpreter contains thus features to "translate" a service request or a service description (source language) into the target language generating automatically a request description that contains only terms of the controlled vocabulary. The request description can be a mixture of natural language and command language (see example below). The second step can be the translation of the respective standards as required by the web services on a lower level of granularity. The "translation" of the essential content of a web service – the automatic generation of service requests – thus has to contend with text mining methods that translate unambiguously service descriptions into suitable terms of the target language. The way the processes are orchestrated is depicted in figure 3.

A robust text analysis method has to rely on predefined templates and pattern-based extraction rules to extract meaning. TFIDF (weighting-based) or LSI (latent semantic indexing) are the most prominent methods for the classification of content in unstructured textual data. The problem with such methods is that they can estimate the importance of the relevance of a term only in the shadow of global information about a larger corpus. And in addition, they cannot solve sufficiently the problem of

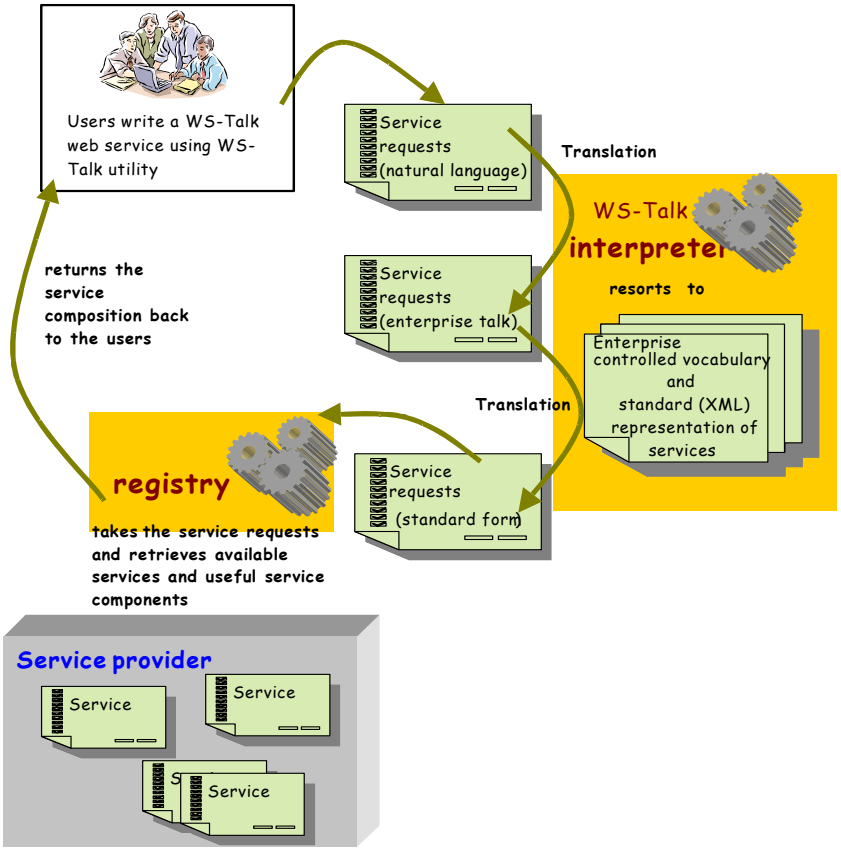


Fig. 3. The schema shows how WS-Talk components operate within a network of web services

ambiguity. This is a crucial drawback when the relevance estimation cannot or should not resort to a larger corpus because of performance purposes. To circumvent this problem in WS-Talk, we resort to the "vector voting method" [13] developed for and successfully applied in IRAIA a service provider platform designed for large and complex information spaces. In principle, each term in a text "votes" for a concept of the concept hierarchy if it contains a matching term. The more "votes" a concept receives, the stronger is the link between the text and that concept. In fact, the method is more complex as concept terms are not only those from the concept itself, but also those from related synonyms and terms associated with the concept ancestors. In the end, the technique uses the whole corresponding concept as index term.

A semantic co-ordinate system helps to determine the correct location of an information item like a service description or a user request within a domain-specific context. This positioning process is tantamount to correctly matching requests and information items as is realized by IRAIA or similar search engines. If we now deploy this approach in the context of web services we implement the same features as presented above for service description and identification:

1. A web service is described in the same way as a user defines a query using the concepts of the controlled vocabulary. This description is in any case easier and faster to specify than developing a corresponding XML or WSDL structure expressing the functionality of the service,
2. The matching mechanism to find a suitable service operates locates a document at the semantic co-ordinates that match a user query or a similar request from an intermediary service,
3. Instructions how to handle data that are passed from one service to another are also expressed in terms of the controlled vocabulary and may include command language components, and
4. The NLP part of the web service hides unnecessary implementation details from service consumers, the non-technical end users.

Providing the controlled vocabulary in different languages in parallel has the advantage of exchanging service descriptions even across language boundaries, and this avoids the users having to apply a language they are not familiar with or having to be trained in a formal description language.

5 An Example of a Scenario from the WS-Talk Prototype

The following example scenario addresses a data mining environment supporting the economic analysis of an enterprise. It illustrates in a living example how WS-Talk-specific components as shown in the schema above (figure 3) co-operate in a network of web services. An analyst may be interested in creating a web service that searches time series related to economic activities (like the amount of incoming orders, productivity, exports, etc.) of all economic sectors that show a certain impact on a specific sector. The principal idea behind this analysis is that present activities in one industrial sector have an impact on the future economic situation of other sectors. In economic analysis, usually such an analysis helps to determine forecast indicators.

The analyst would create a Web service with the following specification: “Search for time series in our own and the OECD databases that represent all economic activities of all industrial sectors that have an impact on the products from the lubricants section of our company.” This service doesn’t exist so far. The service definition tool of WS-Talk “translates” the description into key words of the company’s enterprise talk, i.e. transforms the original phrase in source language into terms of the target language, the company’s controlled vocabulary. The “translation” process may include term replacements or query expansions. The concept of “products from the lubricants section of our company” may be replaced by a list of concrete product definitions. The subsequent localisation process identifies the semantic co-ordinates corresponding to the key words resulting from the “translation” and searches for one or more existing services that have the same semantic co-ordinates. It is quite plausible that the required web service has to be constructed from a number of services that exist within the company or located outside.

Let’s assume that in our example there is a service available at OECD that enables the access to its time series database, a further one of the company that presents these time series in business charts, and a third one that identifies impact relationships between time series using time-warping as known from speech recognition and recently also applied in economic analysis. The latter service may come from the company’s consultant. If the requested service is available, it will be propagated when its description is fully matched

by the semantic co-ordinates of the existing services. Missing pieces are reported back to the service requester. She or he may then re-formulate the description or ask the software department to provide for the missing component(s).

The services themselves have some co-ordination capabilities. From the localization they know which is the superior process they are related to and know, thus, the components they are co-operating with. The time-warping tool knows that it requires input from the OECD tool, and the OECD tool may at first confirm if this web service of this specific company is authorized to retrieve OECD data. The exchange of data may be based on XML-specifications and further meta-data descriptions. We shall extend the above example with implementation details from the actual prototype of WS-Talk, that also show how the semantic co-ordination system is instrumented and how interfacing is developed.

The technology for semantic co-ordinate systems applied in WS-Talk is based on IRAIA. The thematic domain is reflected by concept hierarchies that are derived from existing taxonomies. For the ambience of economic information, for instance, in IRAIA a powerful taxonomy has been produced that merges two of the most important structures in this field: Eurostat's NACE (Nomenclature des Activités dans la Communauté Européenne -systematic of the economic activities of the European Union) and the industry systematic of the ifo Institute for Economic Research.

Figure 4 shows how the unified taxonomy creates a semantic co-ordinate system that enables exact and automatic positioning of coherent information items like documents or service descriptions. The concept hierarchies presented make up the domain-talk of economic analysts. The concept hierarchies can be regarded as a controlled vocabulary of the thematic area a European economic research institute is dealing with. They are constantly checked for completeness against the content covered by the text and time series documents reflecting comprehensively and in detail the organization's business. This approach can be extended easily towards descriptions of tasks that are related to economic analysis and to web services performing or supporting these tasks.

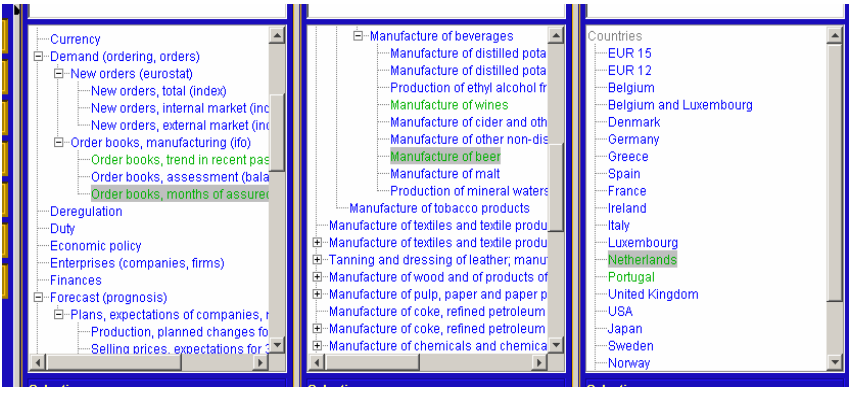


Fig. 4. The unified taxonomy

The following table shows a part of a description of a JXTA service as being used for accessing and processing time series. A well-formulated description usually reflects precisely the nature of the essentials of the service and the way how to use it.

Fig. 5. Translation of a user request into the target language

A concept hierarchy containing an arrangement of task concepts and commands can then be used to analyze a service description and to represent its purpose by its most significant concepts. Commands are added to the title of the respective function like synonyms. The same holds for database specific codes of thematic concepts. In our example “Chemical Industry” is represented by the code “b25000” in the Eurostat database. In the same way a request expressed in natural language can be distilled to its most significant concepts from the same controlled vocabulary. This “translation process” is a preparatory step for the subsequent matching process that identifies service(s) necessary and suitable for the request. Figure 5 shows a user request for a service and the machine-processable “translation” of the request as feedback.

Table 1. Description section of a (JXTA) web service that retrieves and processes time series

```
<Svc>
  ...
  <Desc>
    TimeSeries service
   (Retrieves time series specifies processing methods)
    Parameters:
    Types of processing methods:
    moving averages, smoothing::SMOOTH-MA::int
    period-on-period::PERIOD-ON-PERIOD
    ...
    Time scope, recent n periods::SCOPE::int
    Time scope, from a to b::SCOPE::Calendar::Calendar
    Time scope, from a on::SCOPE::Calendar
  </Desc>
</Svc>
```

Concept hierarchies related to the tasks of economic analysis are organized in the same way. Like the entries of those hierarchies reflecting the thematic domain they are arranged according to main tasks and sub-tasks. In addition, the respective nodes point to the code or command necessary for the web service specification. The example in Table 2 shows a fraction of this hierarchy addressing the processing of time series.

Table 2. Processing of time series

```
87     time series
88            processing
89                   ...
98                   smoothing
99                          moving average::SMOOTH-MA
100                         ...
104                  growth rates
105                                year-on-year::YEAR-ON-YEAR
106                                period-on-period::PERIOD-ON-
       PERIOD
107           ...
```

6 Conclusions

Despite advances in the development of semantic web standards, natural-language processing methods still form part of many techniques for enabling computers to understand and engage in human communication [8]. This is not at all a paradox since much information is passed or stored in natural language. Semantic web standards do not replace NLP and vice versa, and the synthesis of both disciplines offers an enormous potential for a new generation of web service technology. WS-Talk emerges from the cross-fertilisation between semantic web technologies and text mining methods. The idea behind this approach is to enable small communities to set up ad-hoc web services using their own language instead of resorting to tools too complicated for them to define or modify their own context architecture.

A crucial component of such a design environment is the automatic and precise transformer of a web service description into the catalogue logic of the respective application domain. This transformer resorts to methods supporting semantic context-awareness. It resides on the users' side putting them in the position to describe the services they require in their ambience.

Allowing software developers and service consumers to write an interface specification in their own words (i.e. allowing all to become context architectures), to avoid the complicated process of reaching an enterprise-wide agreement on this interface standard, sounds first of all like shutting a Pandora's box in favour of opening another one. However, as we harness only the most stable part of web service specifications we also embark only on text analysis and information mapping methods that are in the position to tackle adequately the problem of ambiguity in interpreting service request and correctly locating the adequate services.

Future applications will emerge from networks of web services that are defined within an organization and are used across the organizations' boundaries and the architects will be service providers and service requesters. The example of WS-Talk shows that, beside semantic web specifications, natural language processing has a powerful role in these networks as it allows non-technical people to set up and maintain essential parts of a company's palette of services.

References

1. Filman, R.E.: Semantic Services. IEEE Internet Computing 7(4) (2003) 4-6
2. Shirky, C.: Web Services and Context Horizons. IEEE Computer 35(9) (2002) 98-100
3. WebServices.Org: New Specifications Intended to Harmonzie Grid and Web Service Standards. http://www.webservices.org/index.php/article/view/1314/ January (2004)
4. Fowler, M.: UML Distilled: Applying the Standard Object Modelling Language. Addison-Wesley, Reading USA (1997)
5. McGovern, J., Tyagi, S., Stevens, M., Mathew, S.: Java Web Service Architecture. Morgan Kaufmann Publishers, San Francisco USA (2003)
6. Winograd, T.: Bringing Design to Software. ACM Press, New York USA (1996)
7. Lopes, C.V., Dourish, P., Lorenz, D.H., Lieberherr, K.: Beyond AOP: toward naturalistic programming. ACM SIGPLAN Notices 38(December) (2003) 34-43
8. Ciravegna, F., Harabagiu, S.: Recent Advances in Natural Language Processing. IEEE Intelligent Systems 18(1) (2003) 12-13
9. Kiczales, G., Lamping, J., Mendhekar, M., Maeda, C., Lopes, C., Loingtier, J.-M., Irwin, J.: Aspect-Oriented Programming. In: proc. of the European Conference on Object-Oriented Programming (ECOOP'97) (1997)
10. Miller, L.A.,: Natural Language Programming: Styles, Strategies, and Contrasts. IBM Systems Journal 20(2) (1981) 184-215
11. Nardi, B.: A Small Matter of Programming: Perspectives on End User Computing. MIT Press, Cambridge USA (1993)
12. Englmeier, K., Mothe, J.: Natural language meets semantic web. http://www.kt-web.org/doc/Englmeier-NLP-SW.pdf July (2003)
13. Mothe, J., Chrisment, C., Dousset, B., Alaux, J.: DocCube: Multi-Dimensional Visualisation and Exploration of Large Document Sets. Journal of the American Society for Information Science and Technology 54(March) (2003) 650-659

Process Mining for Ubiquitous Mobile Systems: An Overview and a Concrete Algorithm

A.K.A. de Medeiros, B.F. van Dongen, W.M.P. van der Aalst, and A.J.M.M. Weijters

Department of Technology Management, Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
{a.k.medeiros, b.f.v.dongen, w.m.p.v.d.aalst,
a.j.m.m.weijters}@tm.tue.nl

Abstract. Ubiquitous Mobile Systems (UMSs) allow for automated capturing of events. Both mobility and ubiquity are supported by electronic means such as mobile phones and PDAs and technologies such as RFID, Bluetooth, WLAN, etc. These can be used to automatically record human behavior and business processes in detail. UMSs typically also allow for more flexibility. The combination of flexibility (i.e., the ability to deviate from standard procedures) and the automated capturing of events, provides an interesting application domain for *process mining*. The goal of process mining is to discover process models from event logs. The α-algorithm is a process mining algorithm whose application is not limited to ubiquitous and/or mobile systems. Unfortunately, the α-algorithm is unable to tackle so-called "short loops", i.e., the repeated occurrence of the same event. Therefore, a new algorithm is proposed to deal with short loops: the α^+-algorithm. This algorithm has been implemented in the EMiT tool.

Keywords: process mining, workflow mining, Petri nets, mobile computing, ubiquitous computing.

1 Process Mining in Ubiquitous Mobile Systems

Since Mark Weiser first described it in 1991 [19], ubiquitous computing has transformed from a fairy-tale to reality. Unlike the traditional situation where people share a computer (mainframe) or where there is a one-to-one correspondence between people and computers (PC), people are surrounded by computing devices. These devices may have a fixed location (cf. "smart homes"), but increasingly these devices are mobile. Typical examples that exist today are PDA's and mobile phones. New concepts like "smart clothes" or "wearable computers" illustrate future applications. Research projects such as UWA (Ubiquitous Web Applications, cf. `www.uwaproject.org`) and MOTION (MObile Teamwork Infrastructure for Organizations Networks, cf. `www.motion.softeco.it`) illustrate the scientific challenges ubiquitous and/or mobile systems are posing.

Clearly, Ubiquitous Mobile Systems (UMSs) will change the way people work [10]. However, in this paper we do not discuss the impact of new technologies on people and organizations but focus on the use of process mining to monitor the processes where UMSs are involved. The main contribution is an extension of the α-algorithm [7] to tackle so-called "short loops" [13], i.e., the paper consists of two clearly distinguishable parts: (1) an introduction to process mining and its relevance for UMSs and (2) a concrete

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 151–165, 2004.

algorithm called the α^+-algorithm. We start with the first part and discuss the relation between process mining and UMSs.

UMSs are enabled by recent developments in various fields. Miniaturization and mass fabrication of "computer-like devices" allows for an omnipresence of computing power. This combined with advances in wireless communication have made mobile computing a reality. The difference between phones and portable computers (e.g., PDAs) is fading, thus illustrating this trend. Current technologies such as Bluetooth and WLAN enable new ways of cooperation. RFID tags allows for new ways of "synchronizing" the physical and electronic worlds. These technologies have in common that potentially it can be used to automatically capture data on work processes and other human behavior. Humans are very good in problem solving but are typically not very accurate or thorough in recording information on events that take place. A human's sensory and short term memories have limited capacity and data entry is considered a tedious job. Therefore, UMSs can serve as natural assistants to record events.[1]

UMSs typically allow for more flexibility than traditional information systems. For many work processes this increased flexibility will trigger the need to monitor things more closely. The combination of availability of event data and the desire to monitor suggests the use of *process mining* techniques. Fueled by the omnipresence of event logs in transactional information systems (cf. WFM, ERP, CRM, SCM, and B2B systems), process mining has become a vivid research area [5, 6]. Until recently, the information in these event logs was rarely used to analyze the underlying processes. Process mining aims at improving this by providing techniques and tools for discovering process, control, data, organizational, and social structures from event logs, i.e., the basic idea of process mining is to diagnose business processes by mining event logs for knowledge.

The event log typically contains information about events referring to an *activity* and a *case*. The case (also named process instance) is the "thing" which is being handled, e.g., a customer order, a job application, an insurance claim, a building permit, etc. The activity (also named task, operation, action, or work-item) is some operation on the case. Typically, events have a *timestamp* indicating the time of occurrence. Moreover, when people are involved, event logs will typically contain information on the person executing or initiating the event, i.e., the *originator*. Based on this information several tools and techniques for process mining have been developed [2, 4, 5, 7, 8, 9, 11, 12, 15, 17, 18]. In this paper we present the α^+-algorithm. This is a new algorithm focusing on the control-flow (i.e., process) perspective. It extends the α-algorithm [7] by addressing the problem of "short loops" [13].

The remainder of this paper is organized as follows. Section 2 introduces the concept of process mining. Section 3 describes the α-algorithm and its supporting definitions. Section 4 presents the new approach to tackle length-two loops using the α-algorithm. Section 5 shows how to extend the approach in Section 4 to mine also length-one loops. Section 6 discusses related works. Section 7 has the conclusions.

[1] Note that in most mobile systems there is a communication asymmetry, i.e., the bandwidth downstream (server-to-client) is much larger than upstream. This may complicate data collection in a mobile system. However, the bandwidth required to record events is typically moderate.

2 Process Mining: An Introduction

We assume that in UMSs, information on events (e.g., the execution of a task by a worker) is recorded in a log.

Table 1. An event log

case identifier	task identifier
case 1	task A
case 2	task A
case 3	task A
case 3	task B
case 1	task B
case 1	task C
case 2	task C
case 4	task A
case 2	task B
case 2	task D
case 5	task E
case 4	task C
case 1	task D
case 3	task C
case 3	task D
case 4	task B
case 5	task F
case 4	task D

To illustrate the principle of process mining, we consider the event log shown in Table 1. This log contains information about five cases (i.e., process instances) and six tasks (A..F). Based on the information shown in Table 1 and by making some assumptions about the completeness of the log (i.e., if a task can follow another task, there is an event trace to show this) we can deduce for example the process model shown in Figure 1. The model is represented in terms of a Petri net [16]. After executing A, tasks B and C are in parallel. Note that for this example we assume that two tasks are in parallel if they appear in any order. By distinguishing between start events and end events for tasks it is possible to explicitly detect parallelism. Instead of starting with A the process can also start with E. Task E is always followed by task F. Table 1 contains the minimal information we assume to be present.

For this simple example, it is quite easy to construct a process model that is able to regenerate the event log. For larger process models this is much more difficult. For example, if the model exhibits alternative and parallel routing, then the process log will typically not contain all possible combinations. Moreover, certain paths through the process model may have a low probability and therefore remain undetected. Noisy data (i.e., logs containing exceptions) can further complicate matters [18]. These are just some of the problems that we need to face in process mining research. In this paper we assume perfect information: (i) the log must be complete (i.e., if a task can follow another task directly, the log contains an example of this behavior) and (ii) the log is noise free (i.e., everything that is registered in the log is correct).

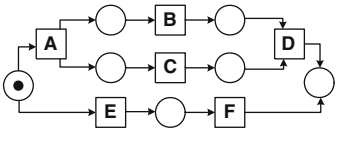


Fig. 1. A process model corresponding to the event log

Process mining can be viewed as a three-phase process: *pre-processing*, *processing* and *post-processing*. In the pre-processing phase, based on the assumption that the input log contains enough information, the ordering relations between tasks are inferred. The processing phase corresponds to the execution of the mining algorithm, given the log and

the ordering relations as input. In our case, the mining algorithm is the α-algorithm [7]. During post-processing, the discovered model (in our case a Petri-net) can be fine-tuned and a graphical representation can be build.

The focus of most research in the domain of process mining is on mining heuristics based on ordering relations of the events in the event log. Considerable work has been done on heuristics to mine event-data logs to produce a process model. Typically these models can be characterized as workflow models. Existing heuristic-based mining algorithms have limitations as indicated in [13]. Typically, more advanced process constructs are difficult to handle for existing mining algorithms. Some of these problematic constructs are common in workflow applications and, therefore, need to be addressed to enable application in practice. Among these constructs are short loops (see Figure 2) .

The main aim of our research is to extend the class of nets we can correctly mine. The α-algorithm is able to correctly mine sound SWF-nets without short loops [7]. In this paper we prove that it is possible to correctly mine *all nets in the class of sound SWF-nets*. The new mining algorithm is called α^+ and is based on the α-algorithm.

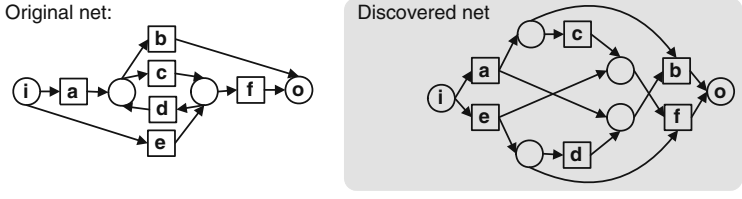


Fig. 2. An example of a sound SWF-net the α-algorithm cannot correctly mine

3 WF-Nets and the α-Algorithm

This section contains the main definitions used in the α-algorithm that are also relevant to the new α^+-algorithm presented in this paper. For more information on the α-algorithm and Structured Workflow Nets (SWF-nets) the reader is referred to [7]. We assume some basic knowledge of Petri nets. Readers not familiar with basic concepts such as (P, T, F) as a representation for a Petri net, the firing rule, firing sequences, preset $\bullet x$, postset $x\bullet$, boundedness, liveness, reachability, etc. are referred to [1, 16].

3.1 Workflow Nets

Before introducing the α-algorithm we briefly discuss a subclass of Petri nets called a *WorkFlow nets* (WF-nets). This subclass is tailored towards modeling the control-flow dimension of a workflow.[2] It should be noted that a WF-net specifies the dynamic behavior of a single case in isolation [1].

Definition 3.1. (Workflow nets) Let $N = (P, T, F)$ be a Petri net and $\bar{t}$ a fresh identifier not in $P \cup T$. N is a *workflow net* (WF-net) iff:

[2] Note that we use the words *workflow* and *process* interchangeably.

1. *object creation*: P contains an input place i such that $\bullet i = \emptyset$,
2. *object completion*: P contains an output place o such that $o\bullet = \emptyset$,
3. *connectedness*: $\bar{N} = (P, T \cup \{\bar{t}\}, F \cup \{(o, \bar{t}), (\bar{t}, i)\})$ is strongly connected,

The Petri net shown in Figure 1 is a WF-net. Note that although the net is not strongly connected, the *short-circuited* net with transition $\bar{t}$ is strongly connected. Even if a net meets all the syntactical requirements stated in Definition 3.1, the corresponding process may exhibit errors such as deadlocks, tasks which can never become active, livelocks, garbage being left in the process after termination, etc. Therefore, we define the following correctness criterion.

Definition 3.2. (Sound) Let $N = (P, T, F)$ be a WF-net with input place i and output place o. N is *sound* iff:

1. *safeness*: $(N, [i])$ is safe,
2. *proper completion*: for any marking $s \in [N, [i]\rangle$, $o \in s$ implies $s = [o]$,
3. *option to complete*: for any marking $s \in [N, [i]\rangle$, $[o] \in [N, s\rangle$, and
4. *absence of dead tasks*: $(N, [i])$ contains no dead transitions.

The set of all sound WF-nets is denoted $\mathcal{W}$.

The WF-net shown in Figure 1 is sound. Soundness can be verified using standard Petri-net-based analysis techniques [1, 3].

Most workflow systems offer standard building blocks such as the AND-split, AND-join, XOR-split, and XOR-join [3]. These are used to model sequential, conditional, parallel and iterative routing. Clearly, a WF-net can be used to specify the routing of cases. *Tasks*, also referred to as *activities*, are modeled by transitions and causal dependencies are modeled by places and arcs. In fact, a place corresponds to a *condition* which can be used as pre- and/or post-condition for tasks. An AND-split corresponds to a transition with two or more output places, and an AND-join corresponds to a transition with two or more input places. XOR-splits/XOR-joins correspond to places with multiple outgoing/ingoing arcs. Given the close relation between tasks and transitions we use the terms interchangeably.

Our process mining research aims at rediscovering WF-nets from event logs. However, not all places in sound WF-nets can be detected. For example places may be implicit which means that they do not affect the behavior of the process. These places remain undetected. Therefore, we limit our investigation to WF-nets without implicit places.

Definition 3.3. (Implicit place) Let $N = (P, T, F)$ be a Petri net with initial marking s. A place $p \in P$ is called implicit in (N, s) if and only if, for all reachable markings $s' \in [N, s\rangle$ and transitions $t \in p\bullet$, $s' \geq \bullet t \setminus \{p\} \Rightarrow s' \geq \bullet t$.[3]

Figure 1 contains no implicit places. However, adding a place p connecting transition A and D yields an implicit place. No mining algorithm is able to detect p since the addition of the place does not change the behavior of the net and therefore is not visible in the log.

[3] $[N, s\rangle$ is the set of reachable markings of net N when starting in marking s, $p\bullet$ is the set of output transitions of p, $\bullet t$ is the set of input places of t, and $\geq$ is the standard ordering relation on multisets.

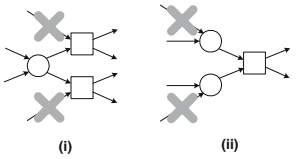


Fig. 3. Constructs not allowed in SWF-nets

For process mining it is very important that the structure of the WF-net clearly reflects its behavior. Therefore, we also rule out the constructs shown in Figure 3. The left construct illustrates the constraint that choice and synchronization should never meet. If two transitions share an input place, and therefore "fight" for the same token, they should not require synchronization. This means that choices (places with multiple output transitions) should not be mixed with synchronizations. The right-hand construct in Figure 3 illustrates the constraint that if there is a synchronization all preceding transitions should have fired, i.e., it is not allowed to have synchronizations directly preceded by an XOR-join. WF-nets which satisfy these requirements are named *structured workflow nets* and are defined as:

Definition 3.4. (SWF-net) A WF-net $N = (P, T, F)$ is an *SWF-net* (Structured workflow net) if and only if:

1. For all $p \in P$ and $t \in T$ with $(p, t) \in F$: $|p \bullet| > 1$ implies $|\bullet t| = 1$.
2. For all $p \in P$ and $t \in T$ with $(p, t) \in F$: $|\bullet t| > 1$ implies $|\bullet p| = 1$.
3. There are no implicit places.

This paper introduces the α^+-algorithm, which mines *all* SWF-nets. The α^+-algorithm is based on the α-algorithm, which correctly mines SWF-nets *without short loops*. In our solution, we first tackle length-two loops (see Section 4) and then also length-one loops (see Section 5). While tackling length-two loops only, we do not allow the nets to have length-one loops. That is why we introduce the definition of *one-loop-free workflow nets*.

Definition 3.5. (One-loop-free workflow nets) Let $N = (P, T, F)$ be a workflow net. N is a *one-loop-free* workflow net if and only if for any $t \in T$, $t \bullet \cap \bullet t = \emptyset$.

3.2 The α-Algorithm

The starting point for process mining is the event log. A log is a set of traces. Event traces and logs are defined as:

Definition 3.6. (Event trace, event log) Let T be a set of tasks. $\sigma \in T^*$ is an *event trace* and $W \in \mathcal{P}(T^*)$ is an *event log*.[4]

From an event log, ordering relations between tasks can be inferred. In the case of the α-algorithm, every two tasks in the event log must have one of the following four ordering relations: $>_W$ (follows), $\rightarrow_W$ (causal), $\|_W$ (parallel) and $\#_W$ (unrelated).

[4] T^* is the set of all sequences that are composed of zero of more tasks from T. $\mathcal{P}(T^*)$ is the powerset of T^*, i.e., $W \subseteq T^*$.

These ordering relations are extracted based on local information in the event traces. The ordering relations are defined as:

Definition 3.7. (Log-based ordering relations) Let W be an event log over T, i.e., $W \in \mathcal{P}(T^*)$. Let $a, b \in T$:

- $a >_W b$ if and only if there is a trace $\sigma = t_1 t_2 t_3 \ldots t_{n-1}$ and $i \in \{1, \ldots, n-2\}$ such that $\sigma \in W$ and $t_i = a$ and $t_{i+1} = b$,
- $a \rightarrow_W b$ if and only if $a >_W b$ and $b \not>_W a$,
- $a \#_W b$ if and only if $a \not>_W b$ and $b \not>_W a$, and
- $a \|_W b$ if and only if $a >_W b$ and $b >_W a$.

To ensure the event log contains the minimal amount of information necessary to mine the process, the notion of log completeness is defined as:

Definition 3.8. (Complete event log) Let $N = (P, T, F)$ be a sound WF-net, i.e., $N \in \mathcal{W}$. W is an *event log of* N if and only if $W \in \mathcal{P}(T^*)$ and every trace $\sigma \in W$ is a firing sequence of N starting in state $[i]$ and ending in state $[o]$, i.e., $(N, [i])[\sigma\rangle(N, [o])$. W is a *complete* event log of N if and only if (1) for any event log W' of N: $>_{W'} \subseteq >_W$, and (2) for any $t \in T$ there is a $\sigma \in W$ such that $t \in \sigma$.

For Figure 1, a possible complete event log W is: *abcd*, *acbd* and *ef*. From this complete log, the following ordering relations are inferred:

- (follows) $a >_W b$, $a >_W c$, $b >_W c$, $b >_W d$, $c >_W b$, $c >_W d$ and $e >_W f$.
- (causal) $a \rightarrow_W b$, $a \rightarrow_W c$, $b \rightarrow_W d$, $c \rightarrow_W d$ and $e \rightarrow_W f$.
- (parallel) $b \|_W c$ and $c \|_W b$.

Now we can give the formal definition of the α-algorithm followed by a more intuitive explanation.

Definition 3.9. (Mining algorithm α) Let W be an event log over T. The $\alpha(W)$ is defined as follows.

1. $T_W = \{t \in T \mid \exists_{\sigma \in W} t \in \sigma\}$,
2. $T_I = \{t \in T \mid \exists_{\sigma \in W} t = \mathit{first}(\sigma)\}$,
3. $T_O = \{t \in T \mid \exists_{\sigma \in W} t = \mathit{last}(\sigma)\}$,
4. $X_W = \{(A, B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall_{a \in A} \forall_{b \in B} a \rightarrow_W b \wedge \forall_{a_1, a_2 \in A} a_1 \#_W a_2 \wedge \forall_{b_1, b_2 \in B} b_1 \#_W b_2\}$,
5. $Y_W = \{(A, B) \in X_W \mid \forall_{(A', B') \in X_W} A \subseteq A' \wedge B \subseteq B' \Longrightarrow (A, B) = (A', B')\}$,
6. $P_W = \{p_{(A,B)} \mid (A, B) \in Y_W\} \cup \{i_W, o_W\}$,
7. $F_W = \{(a, p_{(A,B)}) \mid (A, B) \in Y_W \wedge a \in A\} \cup \{(p_{(A,B)}, b) \mid (A, B) \in Y_W \wedge b \in B\} \cup \{(i_W, t) \mid t \in T_I\} \cup \{(t, o_W) \mid t \in T_O\}$, and
8. $\alpha(W) = (P_W, T_W, F_W)$.

The α-algorithm works as follows. First, it examines the event traces and (Step 1) creates the set of transitions (T_W) in the workflow, (Step 2) the set of output transitions (T_I) of the source place , and (Step 3) the set of the input transitions (T_O) of the sink place[5]. In

[5] In a WF-net, the source place i has no input transitions and the sink place o has no output transitions.

steps 4 and 5, the α-algorithm creates sets (X_W and Y_W, respectively) used to define the places of the discovered WF-net. In Step 4, the α-algorithm discovers which transitions are causally related. Thus, for each tuple (A, B) in X_W, each transition in set A causally relates to *all* transitions in set B, and no transitions within A (or B) follow each other in some firing sequence. These constraints to the elements in sets A and B allow the correct mining of AND-split/join and XOR-split/join constructs. Note that the XOR-split/join requires the fusion of places. In Step 5, the α-algorithm refines set X_W by taking only the largest elements with respect to set inclusion. In fact, Step 5 establishes the exact amount of places the discovered net has (excluding the source place i_W and the sink place o_W). The places are created in Step 6 and connected to their respective input/output transitions in Step 7. The discovered WF-net is returned in Step 8.

Finally, we define what it means for a WF-net to be *rediscovered*.

Definition 3.10. (Ability to rediscover) Let $N = (P, T, F)$ be a sound WF-net, i.e., $N \in \mathcal{W}$, and let α be a mining algorithm which maps event logs of N onto sound WF-nets, i.e., $\alpha : \mathcal{P}(T^*) \rightarrow \mathcal{W}$. If for any complete event log W of N the mining algorithm returns N (modulo renaming of places), then α is able to *rediscover* N.

Note that no mining algorithm is able to find names of places. Therefore, we ignore place names, i.e., α is able to rediscover N if and only if $\alpha(W) = N$ modulo renaming of places.

4 Length-Two Loops

In this section we first show why a new notion of log completeness is necessary to capture length-two loops in SWF-nets and why the α-algorithm does not capture length-two loops in SWF-nets (even if the new notion of log completeness is used). Then a new definition of ordering relations is given, and finally we prove that this new definition of ordering relations is sufficient to tackle length-two loops with the α-algorithm.

Log completeness as defined in Definition 3.8 is insufficient to detect length-two loops in SWF-nets. As an example, consider the SWF-net in Figure 2 (left-hand side). This net can have the complete log: *ab, acdb, edcf, ef*. However, by looking at this log it is not clear whether transitions *c* and *d* are in parallel or belong to a length-two loop. Thus, to correctly detect length-two loops in SWF-nets, the following new definition of complete log is introduced.

Definition 4.1. (Loop-complete event log) Let $N = (P, T, F)$ be a SWF-net and W a log of N. W is a *loop-complete event log of* N if and only if W is complete and for all event logs W' of N: if there is a firing sequence $\sigma' \in W'$ with $\sigma' = t_1 t_2 t_3 \dots t_{n'}$ and $i' \in \{1, \dots, n' - 2\}$ such that $t_{i'} = t_{i'+2} = a$ and $t_{i'+1} = b$, for some $a, b \in T : a \neq b$, then there is a firing sequence $\sigma \in W$ with $\sigma = t_1 t_2 t_3 \dots t_n$ and $i \in \{1, \dots, n - 2\}$ such that $t_i = t_{i+2} = a$ and $t_{i+1} = b$.

Note that a *loop-complete event log* for the net in Figure 2 will contain one or more traces with the substrings "*cdc*" and "*dcd*". By definition, all loop-complete event logs are also complete event logs.

The new notion of a loop-complete event log is necessary but not sufficient to mine length-two loops. The main reason is that the tasks in the length-two loop are inferred to be in parallel. For example, for the net in Figure 2, any loop-complete event log will lead to $c\|_W d$ and $d\|_W c$. However, these transitions are not in parallel. In fact, they are connected by places that can only be correctly mined by the α-algorithm if at least $c \rightarrow_W d$ and $d \rightarrow_W c$. Using this insight, we redefine Definition 3.7, i.e., we provide the following new definitions for the basic ordering relations $\rightarrow_W$ and $\|_W$.

Definition 4.2. (Ordering relations capturing length-two loops) Let W be a loop-complete event log over T, i.e., $W \in \mathcal{P}(T^*)$. Let $a, b \in T$:

- $a \triangle_W b$ if and only if there is a trace $\sigma = t_1 t_2 t_3 \ldots t_n$ and $i \in \{1, \ldots, n-2\}$ such that $\sigma \in W$ and $t_i = t_{i+2} = a$ and $t_{i+1} = b$,
- $a \diamond_W b$ if and only if $a \triangle_W b$ and $b \triangle_W a$,
- $a >_W b$ if and only if there is a trace $\sigma = t_1 t_2 t_3 \ldots t_{n-1}$ and $i \in \{1, \ldots, n-2\}$ such that $\sigma \in W$ and $t_i = a$ and $t_{i+1} = b$,
- $a \rightarrow_W b$ if and only if $a >_W b$ and ($b \not>_W a$ or $a \diamond_W b$) ,
- $a \#_W b$ if and only if $a \not>_W b$ and $b \not>_W a$, and
- $a\|_W b$ if and only if $a >_W b$ and $b >_W a$ and $a \not\diamond_W b$.

Note that, in the new Definition 4.2, a and b are also in the $a \rightarrow_W b$ relation if $a >_W b$ and $b >_W a$ and the substrings aba and bab are contained in the event traces.

However, there is still a problem. Length-one loops in the net may also produce "cdc" and "dcd" patterns in the event traces, cf. Figure 4. Therefore, to prove that the α-algorithm can correctly mine length-two loops when using the new definitions of loop-complete event log and ordering relations, we assume that the net is a *sound one-loop-free SWF-net*.

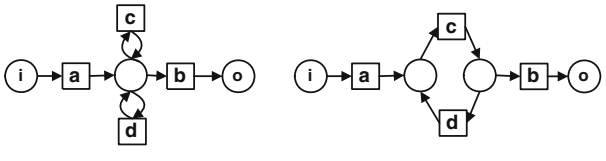


Fig. 4. Example illustrating why length-one loops are not allowed when mining length-two loops. Note that both nets have loop-complete event logs that contain traces with the substrings "cdc" or "dcd"

Theorem 4.3. Let $N = (P, T, F)$ be an one-loop-free sound SWF-net. Let W be a loop-complete event log of N. For any $a, b \in T$, such that $\bullet a \cap b\bullet \neq \emptyset$ and $a\bullet \cap \bullet b \neq \emptyset$, $a \triangle_W b$ implies $b \triangle_W a$.

Proof. For a detailed proof we refer to [14]. □

Using this result, it is possible to prove the following theorem which shows that the new ordeing relations solve the problem of length-two loops.

Theorem 4.4. Let $N = (P, T, F)$ be a sound one-loop-free SWF-net and let W be a loop-complete event log of N. Then $\alpha(W) = N$ modulo renaming of places.

Proof. See [14] for a detailed proof. □

In the next section, we show how to handle the general case (all sound SWF-nets).

5 Length-One Loops

In this section we first show some of the properties of length-one loops in sound SWF-nets. Then we present an algorithm (called α^+) that correctly mines all sound SWF-nets.

Length-one loops are connected to a single place in any sound SWF-net and this place cannot be the source or the sink place of the sound SWF-net, as is stated in the following theorem:

Theorem 5.1. Let $N = (P, T, F)$ be a sound SWF-net. For any $a \in T$, $a\bullet \cap \bullet a \neq \emptyset$ implies $a \notin i\bullet$, $a \notin \bullet o$, $a\bullet = \bullet a$ and $|\bullet a| = 1$.

Proof. See [14]. □

Property 5.2. Let $N = (P, T, F)$ be a sound SWF-net. Let W be a complete event log of N. For any $a \in T : \bullet a \cap a\bullet \neq \emptyset$ implies there are $b, c \in T : a \neq b$ and $b \neq c$ and $a \neq c$ and $b \rightarrow_W a$ and $a \rightarrow_W c$ and $b \rightarrow_W c$ and $\bullet c = \bullet a$.

Theorem 5.3. Let $N = (P, T, F)$ be a sound SWF-net. Let $N' = (P', T', F')$ be a one-loop-free PT-net such that $P' = P$, $T' = \{t \in T | \bullet t \cap t\bullet = \emptyset\}$, and $F' = F \cap (P' \times T' \cup T' \times P')$. Let W be a loop-complete event log of N and let W^{-L1L} be the log created by excluding the occurrences of length-one-loop transitions from every event trace in W. Then:

1. N' is a sound one-loop-free SWF-net,
2. $\alpha(W^{-L1L}) = N'$ modulo renaming of places.

Proof. See [14]. □

Theorem 5.3 states that the main net structure (called N' in the theorem) of any sound SWF-net can be correctly discovered by the α-algorithm whenever length-one-loop transitions are removed from the input log. Consequently, since length-one-loop transitions are always connected to a single place in sound SWF-net (Theorem 5.1), we can use the α-algorithm to mine the main net structure N' and then connect the length-one-loop transition to this net.

The solution to tackle length-one loops in sound SWF-nets focuses on the pre- and post-processing phases of process mining. The key idea is to identify the length-one-loop tasks and the single place to which each task should be connected. Any length-one-loop task t can be identified by searching a loop-complete event log for traces containing the substring tt. To determine the correct place p to which each t should be connected in the discovered net, we must check which transitions are directed followed by t but do not direct follow t (i.e. p is an output place of these transitions) and which transitions direct follow t but t does not direct follow them (i.e. p is the input place of these transitions). Figure 5 shows the basic idea by illustrating a pre- and post-processing phase.

The algorithm - called α^+ - to mine sound SWF-nets is formalized as follows. Note that the function $eliminateTask$ maps any event trace σ to a new one σ' without the occurrence of a certain transition t.

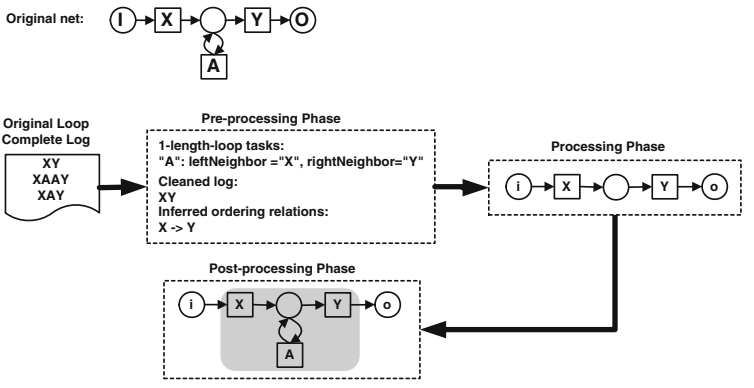


Fig. 5. Example of the approach to tackle length-one loops in sound SWF-net

Definition 5.4. (Mining algorithm α^+) Let W be a loop-complete event log over T, the α-algorithm as in Definition 3.9 and the ordering relations as in Definition 4.2.

1. $T_{log} = \{t \in T | \exists_{\sigma \in W}[t \in \sigma]\}$
2. $L1L = \{t \in T_{log} \mid \exists_{\sigma = t_1 t_2 \ldots t_n \in W; i \in \{1,2,\ldots,n\}}[t = t_{i-1} \wedge t = t_i]\}$
3. $T' = T_{log} \setminus L1L$
4. $F_{L1L} = \emptyset$
5. For each $t \in L1L$ do:
   (a) $A = \{a \in T' \mid a >_W t\}$
   (b) $B = \{b \in T' \mid t >_W a\}$
   (c) $F_{L1L} := F_{L1L} \cup \{(t, p_{(A \setminus B, B \setminus A)}), (p_{(A \setminus B, B \setminus A)}, t)\}$
6. $W^{-L1L} = \emptyset$
7. For each $\sigma \in W$ do:
   (a) $\sigma' = \sigma$
   (b) For each $t \in L1L$ do:
      i. $\sigma' := eliminateTask(\sigma', t)$
   (c) $W^{-L1L} := W^{-L1L} \cup \sigma'$
8. $(P_{W^{-L1L}}, T_{W^{-L1L}}, F_{W^{-L1L}}) = \alpha(W^{-L1L})$
9. $P_W = P_{W^{-L1L}}$
10. $T_W = T_{W^{-L1L}} \cup L1L$
11. $F_W = F_{W^{-L1L}} \cup F_{L1L}$
12. $\alpha^+ = (P_W, T_W, F_W)$

The α^+ works as follows. First, it examines the event traces (Step 1) and identifies the length-one-loop transitions (Step 2). In steps 3 to 5, the places to which each length-one-loop transition should be connected to are identified and the respective arcs are included in F_{L1L}. Then, all length-one-loop transitions are removed from the input log W^{-L1L} to be processed by the α-algorithm (steps 6 and 7). In Step 8, the α-algorithm discovers

a workflow net based on the loop-complete event log W^{-L1L} and the ordering relations as defined in Definition 4.2. In steps 9 to 11, the length-one-loop transitions and their respective input and output arcs are added to the net discovered by the α-algorithm. The workflow net with the added length-one loops is returned in Step 12.

Theorem 5.5. Let $N = (P, T, F)$ be a sound SWF-net and let W be a loop-complete event log of N. Using the ordering relations as in Definition 4.2, $\alpha^+(W) = N$ modulo renaming of places.

Proof. See [14]. □

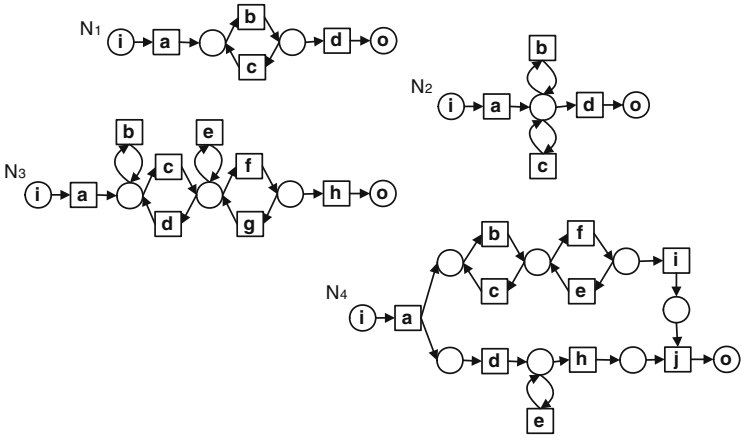


Fig. 6. Examples of sound SWF-nets that the α^+-algorithm correctly mines

The original net in Figure 2 and the nets N_{1-4} in Figure 6 satisfy the requirements stated in Theorem 5.5. Therefore, they are all correctly discovered by the α^+-algorithm. In fact, the α^+ can be extended to correctly discover nets beyond the class of sound SWF-net (cf. [14]).

6 Related Work

The idea of process mining is not new [2, 5, 7, 8, 9, 11, 12, 13, 15, 17, 18] and most techniques aim at the control-flow perspective. However, process mining is not limited to the control-flow perspective. For example, in [4] we use process mining techniques to construct a social network. For more information on process mining we refer to a special issue of Computers in Industry on process mining [6] and a survey paper [5]. In this paper, unfortunately, it is impossible to do justice to the work done in this area.

The focus of this paper is on an extension of the α-algorithm. For more information on the α-algorithm, we refer to [2, 7, 13, 18]. This paper tackles one of the problems raised in [13] and should be considered as an extension of [7]. For more detailed proofs we refer to a technical report [14].

To support our mining efforts we have developed a set of tools including EMiT [2], Thumb [18], and MinSoN [4]. These tools share a common XML format. For more details we refer to www.processmining.org.

7 Conclusion

New technologies such as RFID, Bluetooth, WLAN, etc. and the omnipresence of small computing devices (e.g., mobile phones and PDAs) enable UMSs that can be used to record human behavior and business processes in detail. The combination of flexibility (i.e., the ability to deviate from standard procedures) and the automated capturing of events, suggests that UMSs form an interesting application domain for process mining. In this paper, we did not elaborate on the application of process mining in this domain. Instead, we focused on an extension of the α-algorithm such that it can mine all sound SWF-nets. The new algorithm is called α^+. The α-algorithm is proven to correctly discover sound SWF-nets without length-one or length-two loops. The extension involves changes in the pre- and post-processing phases. First, length-two loops are tackled by redefining the notion of log completeness and the possible ordering relations among tasks in the process. This solution is addressed in the pre-processing phase only. Then, the solution to tackle length-two loops is extended to tackle also length-one loops. The key property is that length-one-loop tasks are connected to single places in sound SWF-nets. Therefore, the α^+-algorithm (1) removes all occurrences of length-one loops from the input log, (2) feeds in the α-algorithm with this log and the new defined ordering relations, and (3) reconnects all the length-one loop tasks to their respective place in the net the α-algorithm produced. In this paper and a technical report [14], we provide a formal proof that the α^+-algorithm correctly mines nets in the *whole* class of sound SWF-nets.

Through our website, www.processmining.org, we provide the tool EMiT that implements the α^+-algorithm. Note that EMiT uses a generic XML-based input format. In addition, EMiT provides adapters for tools like Staffware, InConcert, ARIS PPM, etc. We invite people developing UMSs to log information in our XML format (cf. www.processmining.org). Examples of possible applications we envision include:

- Patients and medical equipment are tagged while the medical staff is equipped with small computing devices. This way health-care processes can be monitored to audit medical protocols, support the medical staff (e.g, alerts), and suggest improvements.
- Service engineers are equipped with PDAs, mobile phones (for internet connection), and GPS (for location). This way service processes can be analyzed.
- Students use ubiquitous/mobile technologies to study anytime/anywhere. To monitor progress and act accordingly, process mining is used to analyze the learning process.

Note that in each of these applications there will be repetitions. Therefore, it is essential that the α^+-algorithm is able to tackle short loops.

Acknowledgements

The authors would like to thank Minseok Song, Laura Maruster, Eric Verbeek, Monique Jansen-Vullers, Hajo Reijers, Michael Rosemann, and Peter van den Brand for their joint efforts on process mining techniques and tools at Eindhoven University of Technology.

References

1. W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
2. W.M.P. van der Aalst and B.F. van Dongen. Discovering Workflow Performance Models from Timed Logs. In Y. Han, S. Tai, and D. Wikarski, editors, *International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002)*, volume 2480 of *Lecture Notes in Computer Science*, pages 45–63. Springer-Verlag, Berlin, 2002.
3. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
4. W.M.P. van der Aalst and M. Song. Mining Social Networks: Uncovering interaction patterns in business processes. In M. Weske, B. Pernici, and J. Desel, editors, *International Conference on Business Process Management (BPM 2004)*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2004.
5. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
6. W.M.P. van der Aalst and A.J.M.M. Weijters, editors. *Process Mining*, Special Issue of Computers in Industry, Volume 53, Number 3. Elsevier Science Publishers, Amsterdam, 2004.
7. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. QUT Technical report, FIT-TR-2003-03, Queensland University of Technology, Brisbane, 2003. (Accepted for publication in IEEE Transactions on Knowledge and Data Engineering.).
8. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.
9. J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.
10. S. Dustdar, H. Gall, and R. Schmidt. Web Services For Groupware in Distributed and Mobile Collaboration. In P. Cremonesi, editor, *Proceeding of the 12th IEEE Euromicro Conference on Parallel, Distributed and Network based Processing (PDP 2004)*, pages 241–247. IEEE Computer Society, Los Alamitos, CA, 2004.
11. J. Herbst. A Machine Learning Approach to Workflow Management. In *Proceedings 11th European Conference on Machine Learning*, volume 1810 of *Lecture Notes in Computer Science*, pages 183–194. Springer-Verlag, Berlin, 2000.
12. IDS Scheer. ARIS Process Performance Manager (ARIS PPM). `http://www.ids-scheer.com`, 2002.
13. A.K.A. de Medeiros, W.M.P. van der Aalst, and A.J.M.M. Weijters. Workflow Mining: Current Status and Future Directions. In R. Meersman, Z. Tari, and D.C. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 389–406. Springer-Verlag, Berlin, 2003.
14. A.K.A. de Medeiros, B.F. van Dongen, W.M.P. van der Aalst, and A.J.M.M. Weijters. Process Mining: Extending the α-algorithm to Mine Short Loops. BETA Working Paper Series, WP 113, Eindhoven University of Technology, Eindhoven, 2004.

15. M. zur Mühlen and M. Rosemann. Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues. In R. Sprague, editor, *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33)*, pages 1–10. IEEE Computer Society Press, Los Alamitos, California, 2000.
16. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
17. M. Sayal, F. Casati, and M.C. Shan U. Dayal. Business Process Cockpit. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 880–883. Morgan Kaufmann, 2002.
18. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
19. M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.

Activity-Based Support for Mobility and Collaboration in Ubiquitous Computing

Jakob E. Bardram

Centre for Pervasive Healthcare,
Department of Computer Science, University of Aarhus,
Aabogade 34, 8200 Aarhus N, Denmark
bardram@daimi.au.dk

Abstract. This paper presents the design philosophy of *activity-based computing* (ABC), which addresses mobility and cooperation in human work activities. Furthermore, it presents the ABC Framework, which is a ubiquitous computing infrastructure supporting activity-based computing. The idea of activity-based computing and the aim of the ABC framework is to: (i) support human activity by managing its collection of work tasks on a computer, (ii) support mobility by porting activities across heterogeneous computing environments, (iii) support asynchronous collaboration by allowing several people to participate in an activity, and (iv) support synchronous, real-time collaboration by enabling 'desktop conferencing' by sharing the activity across several clients. During a period of two years our framework has been co-designed and evaluated in close cooperation with a range of clinicians in a hospital.

1 Introduction

One of the core ideas in ubiquitous computing is to take the computer 'away from the desktop' and have it pervasively available in the environment [20, 7, 9]. This move away from the office setting and into a wide variety of work and domestic settings, bring the issues of mobility and cooperation into front [12, 4, 3].

In this paper we present the concept of Activity-Based Computing (ABC) as a way of thinking about supporting human activities in a ubiquitous computing environment [6]. Furthermore, we present the ABC framework, which is an implementation of the activity-based computing concepts. The framework enables mobile users to continue their activities in a ubiquitous computing environment, while shielding them from managing the low-level details of migrating applications across a heterogeneous and dynamic execution environment. User activities become first class entities that are represented explicitly, and activities are inherently collaborative, treating single-user activities as collaborative activities that just happen to have only one participant. This paper focuses on the mobility and collaborative aspects of activity-based computing, and does not discuss how heterogeneous devices are handled. Evaluations and studies of clinicians using the ABC framework indicate that the idea of organizing computer support in terms of activities has turned out to offer simple, light-weight, and versatile mechanisms for solving some of the traditional hard problems in collaborative systems. For example, the activity and

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 166–180, 2004.

its list of participants becomes a natural conceptual and technical mechanisms for managing collaborative sessions, including session creation, user entrance and departing, and floor-control.

The paper starts by presenting the concepts and principles of Activity-Based Computing and illustrates them in two hospital scenarios, focusing on activity-based support for mobility and cooperation. Sections 3 presents the architecture of the ABC runtime infrastructure. Section 4 and 5 presents the core technical mechanisms in the ABC Framework responsible for supporting mobility and collaboration. Section 6 presents the current implementation and evaluation status of the framework, and section 7 concludes the paper.

2 Activity-Based Computing

The background for developing the activity-based computing concepts is studies of healthcare practices in hospitals [1, 2, 5, 3]. A range of challenging properties in medical work makes it fundamentally different from typical office work; extreme mobility, ad-hoc collaboration, interruptions, and a high degree of communication. This makes healthcare an interesting application area for the design of pervasive computing technology.

Healthcare has a long tradition of using computer-based systems, and clinicians are today faced with many different systems and with a wide range of functionality within each of these. Thus, carrying out a single activity typically involves a lot of different systems and a lot of specific functionality and data presentation within each system. This is illustrated in figure 1. If you ask the doctor what he is doing, he would answer "I'm prescribing medicine for Mr. Hansen". If you instead view it from the computational level, the doctor is actually handling several distinct services or applications, as well as viewing and manipulating a substantial amount of medical data: reviewing the medical history, looking over the medicine schema, studying the results of blood tests, viewing X-ray images, etc. Thus, we can identify at least three levels of abstraction, namely the

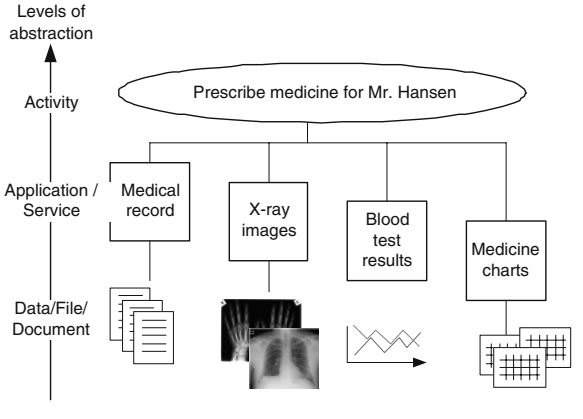


Fig. 1. A single activity involves many services

higher level of human activities (the 'activity level'), the lower level of computational services or applications being used (the 'application level'), and the lowest level of data, files, or material being manipulated (the 'data level'). Our key argument is that contemporary computing systems do not support the activity level, only the application or the data level. Our aim is therefore to explore how to support the activity level directly in the computing system; explore what the concept of "activity" is in this context, and to evaluate how activities may help clinicians in their daily work.

2.1 Scenarios – Activity-Based Computing in a Hospital

Below are two scenarios, which illustrate how activity-based computing supports mobility and cooperation in a hospital. The scenarios are direct extrapolations of real-world scenarios taken from our field-studies of work in a large metropolitan hospital (see e.g. [3]). The first scenario describes a 'morning ward round' for a physician and a nurse at a medical department. It illustrates how activity-based computing supports mobility and asynchronous collaboration.

> *The nurse and the physician meet in the ward's office to prepare for the ward round. The physician comes directly from the medical conference and he has been discussing Mrs. Pedersen's case during this conference. Mrs. Pedersen suffers from leukemia but her treatment is progressing very well. The physician logs in on the wall-based display in the office, and the activity on Mrs. Pedersen used during the medical conference is resumed. All relevant medical data for Mrs. Pedersen is displayed, with her latest blood test results in focus. After a short discussion, the nurse and physician decide that she can be sent home, if she feels comfortable with her medication. The nurse and physician walk to Mrs. Pedersen's bedside, and use the display built into the bed to show her how her treatment is progressing. By logging into the bed computer, the exact same information is restored on the display. The nurse goes to the medicine room to prepare some medicine for Mrs. Pedersen to take home. When she logs into the computer in the pharmacy, the activity is restored, and she can look at the medicine the physician has prescribed. Later in the afternoon, the physician restores the 'Mrs. Pedersen' activity on his TablePC. He notices that the patient has taken her medicine and has been discharged from the ward. He archives the activity, because he does not expect to use it again.*

The second scenario describes how activity-based computing can be used for synchronous, real-time collaboration between a group of physicians and a radiologist.

> *During the ward round the physician receives an invitation from the radiology department to participate in another activity – this is indicated in the 'Activity Bar' (see figure 2) as a blinking icon. A while later, when he has finished talking to Mrs. Pedersen, he checks his list of incoming activities on his PDA. He activates the 'Radiology Conference' activity and walks up to the wall-based display in the ward's office. The display changes into showing a large number of X-ray images and a live video link to the radiologist. The radiologist describes his analysis for each patient, using his pointing device to mark important items*

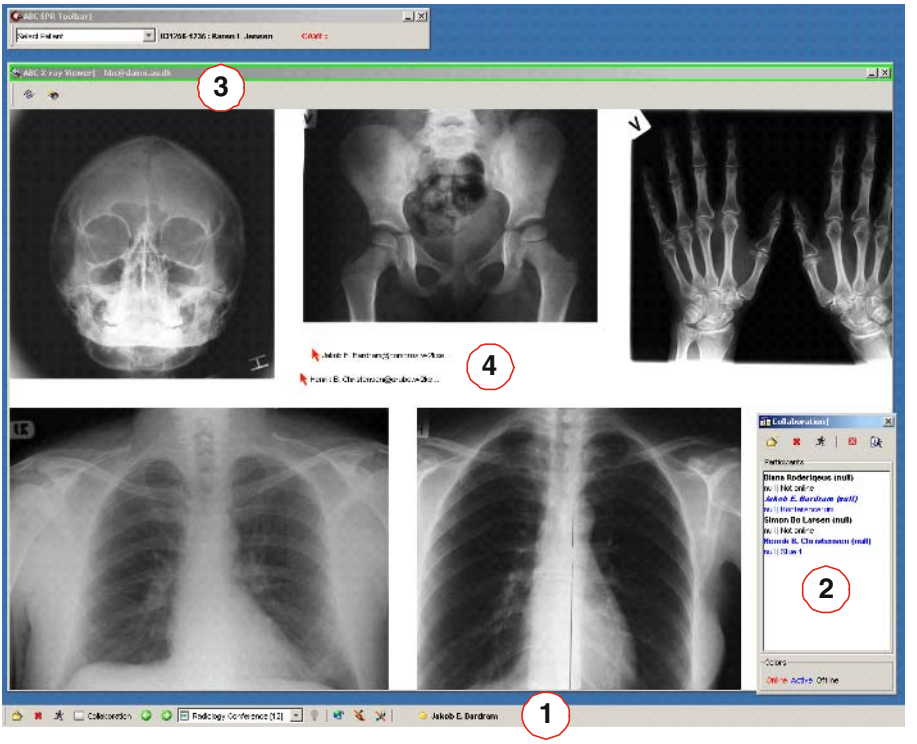


Fig. 2. The ABC user-interface with (1) the Activity Bar, (2) the Collaboration Frame, (3) two ABC Applications implementing the 'EPR' and the 'Radiology Image' service, (4) Telepointers. This picture shows a typical activity for a radiology conference

on the X-ray images. All the users' gestures and annotations on the images are reflected on all the screens participating in the activity.

2.2 Activity-Based Computing Principles

The scenarios above illustrate several central points in our view on activity-based computing and hence the support provided from the ABC framework. Activities can be broken down along three dimensions: (i) time/space, (ii) task, and (iii) users [1]:

– **Time/Space** – In Activity-Based Computing, 'computational activities' (or just activities) become first class entities that are represented explicitly in the framework.

[1] By using the terms 'Activity' and 'Task', Activity-Based Computing might sound like a workflow system. However, in ABC an activity is *light-weight* in the sense that it does not model nor control real-world human activities. A computational activity can be created and modified according to the desire of the user and does not come out of models of work activities. Activity-Based Computing is hence not to be mistaken for a workflow system.

This means that activities are managed and persistently saved over time, are distributed across computational devices that can handle them, and are directly accessible to users in the user-interface. A user has a range of activities which s/he alternates between. Activities are *stateful* by maintaining and storing state information about the unfolding of the activity in time. An activity has a life cycle and can be in four different states; it can be initiated, paused, resumed, and finalized. The activity concept thus supports *interruptions* and *multi-tasking* in work by allowing an activity to be paused and later resumed. Combining this statefulness with the distributed nature of activities implies that an activity can be paused at one host and later resumed and its state restored at another host. Hence, an activity supports *mobility*.

- **Task** – As illustrated in figure 1, an activity is a collection of services. For the activity to become stateful, each service needs to be stateful, i.e. be able to hand over state information to the activity. Services and applications are decoupled. Thus, a tight coupling does not necessarily exist between a service described in an activity and an application running on a host device that can handle this service as part of executing an activity. In the scenario above, X-ray images are shown on the wall-sized display but when resuming this activity on a PDA, proper applications for image rendering might not be available, or might be replaced by a low-fidelity version. Thus, this decoupling between services and applications supports the usage of different applications for e.g. image manipulation on different platforms.
- **Users** – Activities are inherently shared and thereby *collaborative*. An activity can have several participants as illustrated in the scenario above. The list of participants works as an access control list to the activity – only users on the participant list can resume the activity. The sharing of activities also provides an awareness of the state of the activity. A user who resumes an activity will receive the latest state of it – hence s/he will see changes made by other participants. Furthermore, if two or more participants activate an activity simultaneously, they will engage in synchronous real-time 'desktop' conferencing by seeing what each of them is doing.

3 The ABC Framework for Activity-Based Computing

The *ABC Framework* is our current implementation of the above stated activity-based computing principles. The main goal of the ABC framework is to provide a runtime and programming platform for the *development* and *deployment* of activity-based computing applications. This is achieved by having on the one hand a *runtime infrastructure*, which handles the computational complexities of managing distributed and collaborative activities by adapting to the available services or resources in a specific environment. And on the other hand by having a *programming framework* (API), which helps the programmer to create ABC-aware applications that can be deployed in the runtime infrastructure.

This section describes the actual runtime infrastructure that lies beneath the ABC framework. Its responsibilities as for activities are to manage, store, activate, and distribute activities, manage and distribute shared state information, ensuring synchronization methods on collaborative activities, and collaborative session management. Figure 3

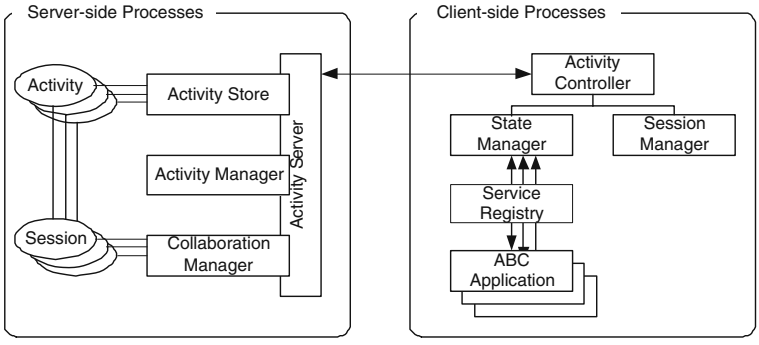


Fig. 3. The ABC Runtime Infrastructure illustrating both server-side and client-side processes

illustrates the ABC runtime infrastructure. It consists of a range of server processes running on one or more servers and a range of client processes supporting the execution of the ABC applications.

- The *Activity Store* handles the persistence of activities by providing an interface to create, delete, and get activities and templates for new activities by reference or query. The store keeps track of the usage history for a user, enabling stepping forward and backward in the list of activities.
- The *Activity Manager* manages the runtime behavior of an Activity by enabling activities to be created, initialized, paused, resumed and finalized by clients.
- The *Collaboration Manager* handles the real-time requirements for synchronous collaboration among active participants within an activity. To do this it manages a *Session* object for each ongoing collaborative activity currently activated by one or more users at different host machines, including the same user on several hosts. Basically, a Session notifies its active participants if the Session or its associated Activity changes. Typical changes are entrance, movement and departing of users in a session and changes to the state of an activity. Parties interested in listening to changes to a Session can add a *Session Listener* to the Session. A central listener on Session objects is the client side *Session Manager* described below.
- The *Activity Controller* is the link between the client and the server. A client's Activity Controller registers at one or more Activity Managers and maintain a link to the *Activity Bar*, which is the user-interface to the ABC infrastructure (see figure 2). The Activity Bar contains a list of non-finalized activities that the user participate in. It provides functionality to create and delete activities, and for launching ABC Applications. The Activity Controller can also be remotely controlled by the Activity Manager, which can force the client to e.g. change activity. The Activity Controller is also notified about relevant events from the server processes. For example, if the current user is invited to participate in another activity, the Activity Controller is notified and an appropriate signal can be made to the user via the bar. When an Activity Controller activates an activity, the local *State Manager* is notified, which in turn uses the *Service Registry* to look up appropriate *ABC Applications*, which can handle the services collected in the activity.

4 State Management and Stateful Applications

A key design invariant in the ABC framework is that applications are stateful and hence can hand over and restore its own state. The runtime infrastructure collects, manages, distributes and synchronizes this state information across the movement of users between physical machines and in the participation in synchronous collaborative sessions (see section 5). The collection of state information from all applications running in an environment is saved in the activity and hence, the activity can be assumed to always contain the shared state. The State Manager is the core process in upholding this invariant. The State Manager is a singleton process running on a client in the ABC infrastructure. It creates the link between the Activity Controller (accessing the server-side Activity Manager) and the client-side applications.

Figure 4 illustrates the interaction between the ABC processes when resuming and pausing activities. When an ABC-aware application is launched, it registers itself at the Service Registry, telling what kind of service it can handle. This is done using RMI, and the ABC application and the Activity Controller can thus run in separate Java virtual machines, potentially on different physical machines. Hence, the Activity Controller running on one device can remotely control applications running on another device. When an Activity is activated via the Activity Controller, the State Manager is notified and extracts the application-specific state information from the activity's state object, and hands it over to each of the applications. This also includes launching and tearing down applications. The application itself is now responsible for restoring its state. The application is basically given back the state information it handed over previously. Similarly, when an activity is deactivated the State Manager asks for state information from all the running applications, which is then handed over to the Activity for storage and distribution.

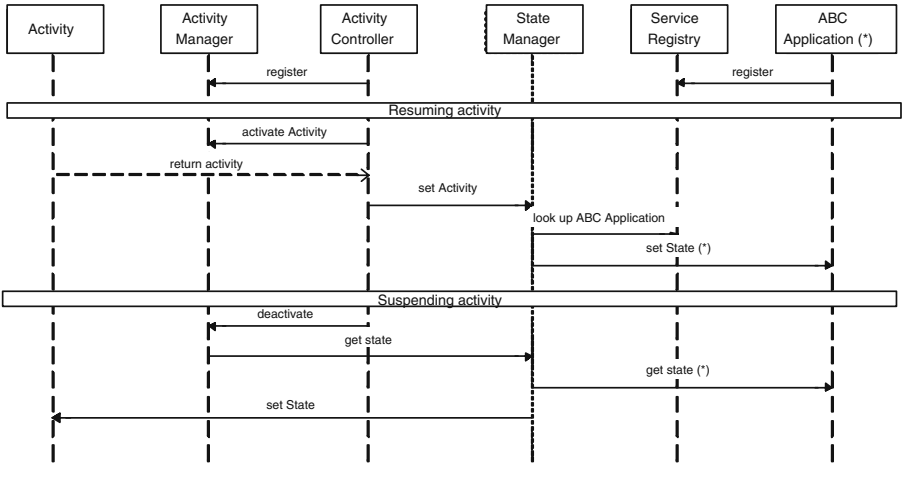


Fig. 4. Interaction diagram showing the collaboration among ABC processes when restoring and saving state information. Multiple instances and calls are marked with (*)

The UML diagram for an Activity is illustrated in figure 5. An activity holds a list of `Participant` objects and a `State` object. This state object is an aggregation of `ApplicationState` objects, holding the state for each service in the activity. When an activity is resumed and the `setState()` method is called, the ABC Application receives an `ApplicationState` object. Similarly, when the `getState()` method is called, the ABC Application must return an `ApplicationState` object. The ABC framework thus requires an 'ABC-aware' application to hand over enough state information to the State Manager for it to restore its current state later, when asked to do so. This puts a burden on the application programmer. To accommodate this, each application state is an aggregation of a set of `ComponentState` objects, holding state information for each component in a service. The ABC API provides several pre-made UI components, which wraps common Swing UI widgets made 'ABC-aware'. For example, ABC versions of JFrame, JScrollpane, JEditorPane, JTextField, and JComboBox exist in the ABC Framework. Each of these ABC Swing UI components can set and get `ComponentState` information when asked by the State Manager. For example, the ABC version of JFrame automatically manages state information about its size, visibility, position, etc. If the programmer uses the ABC Swing library, UI state is managed automatically. A similar approach is used in DistView [14]. The ABC Swing UI components have the same names as their corresponding Swing components and are located in a package called `dk.pervasive.abc.swing`. Hence, a programmer can replace the traditional Swing components with their corresponding ABC Swing components by just changing the `import` statements in a Java application.

Figure 6 shows a simple ABC chat program, which can be used to chat with participants within the same activity. The UI consists of a JFrame (1), a JEditorPane (2) embedded in a JScrollPane (3), a JTextField (4), and a JButton (5). The JFrame and the JScrollPane are ABC-aware components. Figure 7 shows the code for state management in this chat application. The application extends the generic `ABCFrame` class, which extend the ABC version of JFrame. The `ABCFrame` class contains convenient ABC-specific functionality for e.g. getting hold of the State Manager and registering the application. The methods `getABCState()` and `setABCState()` are called in the state management cycle shown in figure 4 and enable the application to save and restore application specific state information. In the chat application the `ChatState` object maintains a copy of the chat text. In combination; when the chat application is suspended and resumed, the following component state information is saved:

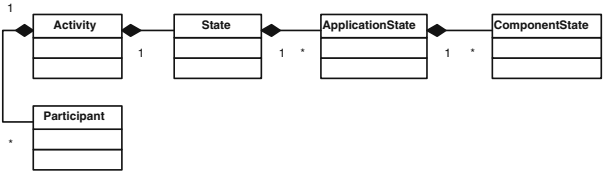


Fig. 5. The UML diagram for an `Activity` and its associated State objects

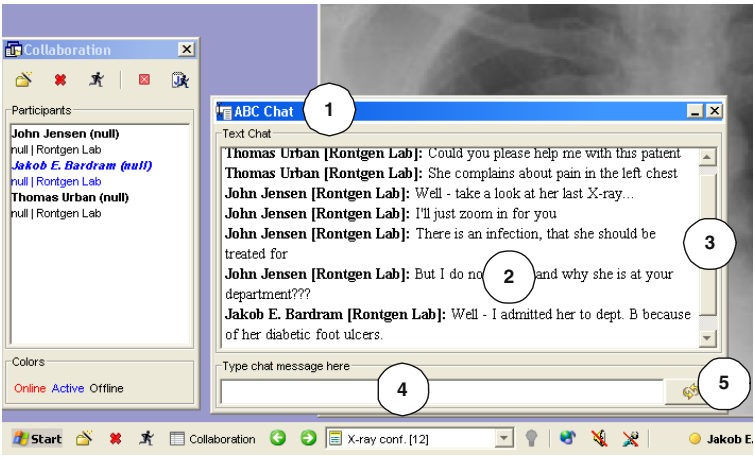


Fig. 6. The ABC Chat application

```
import dk.pervasive.abc.*;

public class ChatApplication extends ABCFrame
                             implements ChatListener {
  ...

  public void setABCState(ComponentState state) {
    if(state instanceof ChatState) {
      ChatState s = (ChatState)state;
      getChatPanel().setChatText(s.getDocument());
    }
  }

  public ComponentState getABCState() {
    ChatState state = new ChatState();
    state.setDocument(getChatPanel().getChatText());

    return state;
  }

  public void chatChanged() {
    fireABCStateChanged();
  }
}
```

Fig. 7. Code listing of state management in a simple chat application. The primary methods are the `getABCState()` and `setABCState()` methods. The `fireABCStateChanged()` method is used in real-time collaboration, as discussed in section 5

- `JFrameState` – Holds the size, visibility, and position of the JFrame.
- `JScrollPaneState` – Holds the position of the vertical and horizontal scrollbars of the JScrollPane.
- `ChatState` – Holds the text in the chat text box.

When a chat application is suspended on one host and resumed on another, the above mentioned state is restored. Any text typed in the JTextField (4) is not saved and restored, because the chat application is using an ordinary (i.e. stateless) JTextField Swing component.

5 Collaboration Management in Real-Time Activity Sharing

An important part of the Activity-Based Computing concept is the support for collaboration – synchronously as well as asynchronously. The state management mechanisms described above support asynchronous collaboration by allowing cooperating users taking turns in activating and working on a shared activity (i.e. one they all participated participated in). In this section we will described the synchronous, real-time collaboration aspects of the ABC framework, which we term *real-time activity sharing*. If two or more participants activates the same activity simultaneously on different clients, the they engage in real-time activity sharing. This means that they are collaborating within this activity and have a synchronized view on the activity. Within CSCW this is normally called desktop conferencing, collaborative applications, or application sharing. Central to real-time activity sharing is the concept of session management, real-time state management in a distributed setup, and the support for direct collaboration 'widgets', like telepointers and audio/video connections.

5.1 Session Management via Sessions and Session Listeners

On runtime the server-side Session class handles the collaborative session in real-time activity sharing. This is illustrated in figure 8. There is a one-to-one relationship between an Activity and a Session. The Session is *persistent*, i.e. it exists even though there are no active users. By simply activating an activity, the user joins the session. Hence, session management is implicit to the user, avoiding the need for explicit session creation, naming, and browsing found in other application sharing systems (e.g. GroupKit [16]). This activity-based session management policy in the ABC framework is a light-weight session management policy similar to the one used in Rendezvous [8], except that in the ABC Framework, it works for a whole set of applications.

Processes can register as Session Listeners to a specific session, thereby receiving notifications about changes to this session. Session listeners receive events when a session changes, for example when a user joins or leaves a session, or when a session's associated activity changes. A central session listener is the Session Manager running on the different clients. The primary responsibility of the session manager is to maintain an up-to-date set of peer-to-peer communication channels to session managers running on the other clients participating in this real-time activity sharing. Another example of a session listener is the Collaboration Frame shown in figure 2. The frame is notified when participants enter or leave the session and the user-interface is updated accordingly.

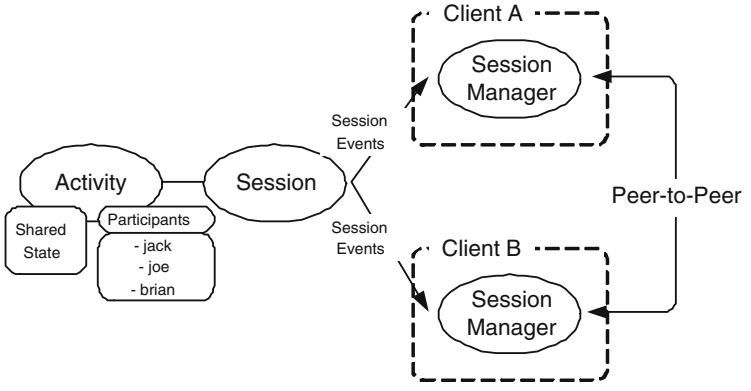


Fig. 8. The interaction between the Activity, the Session, and the Session Manager at each client. The clients are labeled C_A and C_B

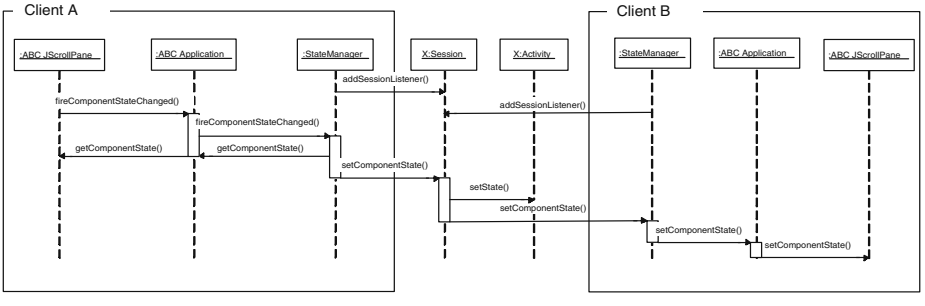


Fig. 9. The UML Sequence Diagram for Real-time Activity Sharing. The diagram shows the interaction between two clients, C_A and C_B

5.2 State Management in Real-Time Activity Sharing

In real-time activity sharing there is a need for passing around the events of the users. For example, if a user scrolls down in a browser window, then all users in this session should see the same. To accomplish this, the ABC framework utilizes the shared state mechanisms described above. Figure 9 illustrates the UML sequence diagram for state management in real-time activity sharing. When Client A (C_A) activates the Activity (X), then C_A's State Manager is added as a Session Listener to the Activity's Session ($Session_X$). Similarly when Client B (C_B) activates X. When the ABC JScrollPane Swing component registers a change (e.g. the user scrolling down), it notifies the State Manager using a cascading sequence of `fireComponentStateChanged()` methods. The State Manager subsequently collects the new state information using the `getComponentState()`method.This new state information is propagated to $Session_X$, which subsequently notifies all Session Listeners about changes to the component state. These listeners include X, as well as C_B. C_B's State Manager is notified and it subsequently notifies the corresponding ABC Application on C_B, which then sets the component state on the correct ABC JScrollPane. Components in an application have unique IDs.

In the chat application in figure 6 and 7 the JFrame and the JScrollPane are stateful. Hence, if one user scrolls down in the text, or moves the window, all active participants will see these changes. The `fireABCStateChanged()` is used to notify the State Manager about application-specific state changes. Every time the chat text is changed, the `chatChanged()` method is called, which again calls the `fireABCStateChanged()` method. In this way, the chat text is replicated between any active participants in the current session, and hence functions as a 'real' real-time chat application among people in this activity. The JTextField used for typing in the chat text is not collaborative (i.e. stateful). Thus, text input in this field is not distributed to other clients in the session – this would also be quite annoying, as users would be editing in the same text line.

5.3 Shared State Synchronization and Latecomers

As described above, the ABC Framework uses the same state management mechanisms for both activity suspending and resuming, as well as for real-time activity sharing across distributed hosts. Therefore, two levels of state granularity in an ABC Application exist. First, the application should be able to handle (get and set) state information for the whole application. This is used in activity suspending and resuming. Second, the application should be able to handle (get and set) state information for each of its components. This is used in real-time activity sharing.

Managing shared state in collaborative systems requires some kind of state synchronization policy. When 2+ users are using an application, concurrency conflicts might arise – e.g. one user scrolling up in a window and another trying to scroll down. The state management policy in ABC is to have *memoryless* states, i.e. an update to a shared state replaces the old one, and it applies *optimistic state synchronization*, like e.g. the Corona server [11, 17]. The decision is based on observations of medical conferences, where concurrency conflicts are rare because people mediate interaction themselves. There are clearly potential race conditions and inconsistency problems when using such an implementation [19], but these conditions are solved by the clinicians. A similar observation is made by Greenberg and Marwood [10], who in their design of the GroupKit toolkit argue that: "The point is that, in many groupware applications, concurrency conflicts may be rare because people mediate interactions themselves." (p. 211). One should, however, note that we are only using the concurrent state management policy for UI state management. All medical data and records are stored in a database server with traditional lock and commit mechanisms.

The shared state mechanisms in ABC facilitates handling *latecomers* to a collaborative session in a straightforward manner. When a client joins a session by activating its activity, the infrastructure simply transfers the current shared state of the activity to the new participant without disrupting the work of the existing members (see [17] for a similar session synchronization strategy).

5.4 Peer-to-Peer Collaboration

As a listener to the collaborative session, each client's Session Manager maintains a complete list of participating host machines in a session, including details of network addresses and their configuration of communication port usage. This list is used to establish, maintain, and destroy direct peer-to-peer communication links to the other

participating clients in the session. The peer-to-peer links are used to create 'collaboration widgets' that provide mutual awareness and information between the participating users. Two examples of such peer-to-peer collaboration widgets developed on top of this peer-to-peer communication mechanism in the ACB framework are *telepointers* and *direct voice links*. When two or more hosts activate the same activity, telepointers and voice links are automatically established. Hence, if a person activates an activity that another person already has activated, both of them will see each other's cursor movements as well as hear each other.

6 Implementation and Evaluation

The framework is currently in a version 3.1. Its core parts are built in Java, using Java RMI for remote communication between distributed hosts. UDP is used in the peer-to-peer communication. The core ABC framework consists of 100+ classes, which contain classes to handle the processes illustrated in figure 3. In addition, we have created ABC Swing classes for the most used Swing components, like JFrame, JScrollPane, JTabbedPane, and JComboBox. We have created a number of ABC-aware applications, like a chat program, a text editor, and a browser, as well as a number of applications for electronic patient records, including a medicine schema, a medical record, an X-ray viewer, and a chart application for measuring and displaying biomedical data such as blood pressure (see figure 2). It often takes a few lines of code to make such applications ABC-aware, most of it having to do with handing over and receiving state information (a similar observation is made in DistView [14]).

The ABC framework is fully functional and implements e.g. the scenarios described in section 2.1, except that the video link is replaced with an audio link only. It has been used in a range of design and evaluation workshops with clinicians from the University Hospital of Aarhus. In total we have conducted more that 15 workshops ranging 6-9 hours each in our test facilities. It is beyond the scope of this paper to discuss these findings in details, but they indicate that the 'activity' as a concept is a rather robust approach to ubiquitous computing. It has unified a range of usability-wise and technical concepts. For example, from a user's perspective, the movement of activities from one host to another seems natural, as well as the concept of collaborative activity sharing. As for the more technical side of it, the activity concept, as implemented in the ABC framework, helps maintain shared state, which is managed and distributed by the framework; the activity concept encompasses a lightweight strategy for session management using the activity's list of participants; and it provides a versatile basis for collaboration awareness by providing a platform to establish a wide range of peer-to-peer type of widgets, like the telepointer and the voice link described above.

The implementation and evaluation of the ABC Framework has, however, also raised several challenges, which we are currently addressing in our research. On a conceptual level, the discrete style of an 'activity' as something you either participate in or not is a subject for further investigation. We are investigating how to support 'peripheral awareness' on your activities and how to 'merge' and more 'gracefully alternate' between activities. On a more technical level we are developing methods for supporting heterogeneous devices in a ubiquitous computing environment. It is, however, not at all

trivial how to support users to suspend an activity on a wall-display and to resume it on a PDA, and vice-versa. And supporting real-time activity sharing amongst heterogeneous devices is even worse – how would that look between a wall-sized display and a PDA? Finally, we are designing a general-purpose ABC protocol that enables non-Java applications to participate as services in the framework.

7 Conclusion

In this paper we have described the *Activity-Based Computing* (ABC) principles, which advocate computational support for human activities across *time/space*, *tasks*, and *users*. An activity is a collection of tasks, having multiple users as participants, and can be suspended and resumed over time and space. ABC allows users to preserve continuity in their work when moving between different computing environments. The key advantage of this approach over more traditional approaches (e.g. thin clients or portable equipment like laptops or PDAs) is that it allows the system to adapt the user's task to the computational resources in the environment, making it possible to use several of these resources concurrently. Further, the framework at its core attacks the problem of handling collaboration, especially session creation and management. As users resume and pause activities, they implicitly engage in a collaborative session.

The ABC Framework is our current implementation of the ABC principles. The key ingredients of the ABC framework are its runtime infrastructure and its programming API, used to develop activity-aware applications as well as to tailor the behavior of the infrastructure. We have described and discussed the concepts of *stateful applications* and *state management* in the ABC Framework. The core point to make here is that this basic state management mechanism facilitates support for both mobility and collaboration in ubiquitous computing environment. State management helps users move between physical machines while preserving their working context; it helps asynchronous collaboration by supporting users in taking turns in working on an activity; it helps collaborative multi-tasking by enabling users to alternate between different activities; and it supports synchronous real-time collaboration through real-time activity sharing.

The framework has been implemented in Java, permitting activity handling for services written within the framework's API. While this implementation is only a first step, it already demonstrates that activity migration, distribution, and collaboration can be supported by the ABC framework. Evaluation of the end-users' view on the framework has been conducted and design recommendations have been incorporated in the framework.

Acknowledgments

The concepts of Activity-Based Computing have been developed in cooperation with Henrik B. Christensen and Claus Bossen, as well as a team of highly engaged physicians and nurses from the University Hospital of Aarhus. The Danish Centre for IT Research (CIT) and ISIS Katrinebjerg funded this research.

References

1. J. E. Bardram. Plans as Situated Action: An Activity Theory Approach to Workflow Systems. In Rodden et al. [15], pages 17–32.
2. J. E. Bardram. Designing for the Dynamics of Cooperative Work Activities. In S. Poltrock and J. Grudin, editors, *Proceedings of the 1998 ACM conference on Computer Supported Cooperative Work*, pages 89–98. ACM Press, 1998.
3. J. E. Bardram and C. Bossen. Moving to get aHead: Local Mobility and Collaborative Work. In K. Kuutti, E. H. Karsten, G. Fitzpatrick, P. Dourish, and K. Schmidt, editors, *Proceedings of the Eighth European Conference on Computer Supported Cooperative Work*, pages 355–374, Helsinki, Finland, Sept. 2003. Kluwer Academic Publishers.
4. V. Bellotti and S. Bly. Walking Away from the Desktop Computer: Distributed Collaboration and Mobility in a Product Design Team. In Olson et al. [13], pages 209–218.
5. C. Bossen. The parameters of common information spaces: The heterogeneity of cooperative work at a hospital ward. In *Proceedings of the 2002 ACM conference on Computer Supported Cooperative Work*, pages 176–185. ACM Press, 2002.
6. H. B. Christensen and J. E. Bardram. Supporting Human Activities – Exploring Activity-Centered Computing. In G. Borriello and L. E. Holmquist, editors, *Proceedings of Ubicomp 2002: Ubiquitous Computing*, volume 2498 of *Lecture Notes in Computer Science*, pages 107–116, Göteborg, Sweden, Sept. 2002. Springer Verlag.
7. M. L. Dertouzos. The future of computing. *Scientific American*, July 1999.
8. K. Edwards. Session Management for Collaborative Applications. In Smith et al. [18], pages 323–330.
9. D. Garlan, D. P. Siewiorek, A. Smailagic, and P. Steenkiste. Project Aura: Toward Distraction-Free Pervasive Computing. *IEEE Pervasive Computing*, 1(2):22–31, Apr. 2002.
10. S. Greenberg and D. Marwood. Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface. In Smith et al. [18], pages 207–217.
11. R. W. Hall, A. Mathur, F. Jahanian, A. Prakash, and C. Rasmussen. Corona: A Communication Serice for Scalable, Reliable Group Communication Systems. In Olson et al. [13], pages 140–149.
12. P. Luff and C. Heath. Mobility in collaboration. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 305–314. ACM Press, 1998.
13. G. Olson, J. Olson, and M. S. Ackerman, editors. *Proceedings of CSCW1996*. ACM Press, 1996.
14. A. Prakash and H. S. Shim. DistView: Support for Building Efficient Collaborative Applications using Replicated Objects. In Smith et al. [18], pages 153–164.
15. T. Rodden, J. Hughes, and K. Schmidtk, editors. *Proceedings of ECSCW97*, Lancaster, UK, Sept. 1997. Kluwer Academic Publishers.
16. M. Roseman and S. Greenberg. Building real-time groupware with groupkit, a groupware toolkit. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 3(1):66–106, 1996.
17. H. S. Shim, R. W. Hall, A. Prakash, and F. Jahanian. Providing Flexible Services for Managing Shared State in Collaborative Systems. In Rodden et al. [15], pages 237–252.
18. J. B. Smith, F. D. Smith, and T. W. Malone, editors. *Proceedings of CSCW1994*. ACM Press, 1994.
19. C. Sun and D. Chen. Consistency maintenance in real-time collaborative graphics editing systems. *ACM Transactions on Computer-Human Interaction (ToCHI)*, 9(1):1–41, 2002.
20. M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, September 1991.

Component-Based Development of Web-Enabled eHome Services

Michael Kirchhof and Sebastian Linz

Department of Computer Science III, Aachen University of Technology,
Ahornstr. 55, 52074 Aachen, Germany
{kirchhof, linz}@i3.informatik.rwth-aachen.de

Abstract. The scope of Web-enabled eHome services covers both automated homes and automated industry facilities. Web-enabled eHome services provide a fully integrated view onto distributed systems comprising automated homes, back-end systems of providers, communication protocols, and services which make distribution aspects transparent. Future platforms should make the development and deployment as easy as achievable. Access should be possible by the way of all communication devices (e.g. desktop computers, PDAs, mobile phones) and all communication networks. Also, all appliances, ubiquitous devices, and their networking protocols have to be supported.

Generations of Web-enabled eHome services have been developed based on proprietary hard- and software. Today, an extensible and modular platform is required for forward-looking design and implementation of such services. One of the main requirements is, that the developed system is maintenance-free and the system brings itself in an operable condition. For setup tasks, both the end-user and a remote operator should be able to execute necessary steps.

It can be observed that services build up hierarchies. We propose a 3-layer system structure, which can be taken to account in system design. Software components grouped by service layers can then be realized in order to implement concrete services. Based on the OSGi platform, we have developed sample services. Gained experience is used for verification of our assumptions. Summarizing, we propose a cookbook for convenient development and deployment of services of the described nature.

1 Introduction

In this paper we will take a look at the interior of connected homes, which build up complex IT systems. The building blocks of such systems are electronic devices, networks, and *services*, which empower the user to interact with his environment. Objects in the environment can be the room itself, the building, other persons, and information from outside. Talking about services, we mean any piece of software, which is executed in a *networked* environment, making usage and administration of pervasive appliances easier.

The scenario is illustrated in figure 1: The connected home on the right-hand side of the drawing is equipped with a *residential gateway*, a hardware device, which provides access to connecting infrastructure (e.g., X.10, EHS, Lon, Jini) and acts as runtime

L. Baresi et al. (Eds.): UMICS 2004, LNCS 3272, pp. 181–196, 2004.

environment for the *service gateway*. The service gateway manages and runs certain software components. These services realize *Web-enabled eHome services*, which are these visible to the users. In our work, we focus on the domains of Security, Consumption, and Infotainment. The services in these domains are based on certain equipment, as some examples are shown in the figure: an alarm system depends on cameras, motion detectors, and lamps or sirens. Monitoring and optimization of energy consumption can be realized by the use of ammeters, photo sensors, thermometers, the heating systems and so on. Elements of the infotainment domain can be incorporated for audio-based and video-based interaction. The communication backbone is an IP-based platform, including a distributed extension. With this extension, internal and external communication can be handled equivalently. Beside direct interaction with devices in the house, interaction with the eHome system based on personal computers, PDAs, and mobile phones is realized. Service providers are connected to the systems via the distributed IP-based platform to provide digital content, applications, and services.

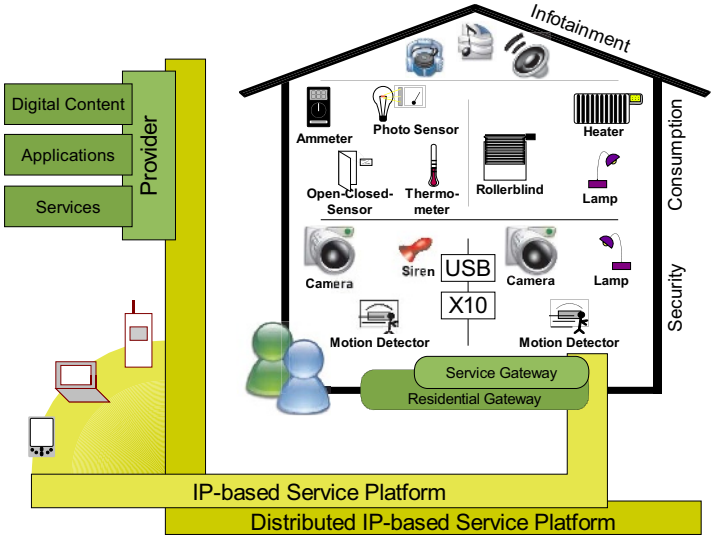


Fig. 1. Scenario

The extension of eHome systems to be connected to the outside world, is crucial for reasonable solutions. It has been shown, that many services profit from knowledge base systems that are maintained by service providers and offer immaterial goods [1]. These providers do not get directly into contact with customers. They merely communicate with one dedicated company in the role of the one and only service provider. Other companies interact and provide services, management data, and specific databases. They act as a virtual enterprise [2]. This stands in contrast to many other solutions: Though they

provide network access, the supported functionalities is most often restricted to remote controlling single devices.

In this scenario, we make use of several technologies: First, we use communication networks for remote access. Second, we use a gateway to access devices *across* protocol boundaries, as we think of many devices working together in a networked environment. One often mentioned problem is the existence of many established home automation standards. While several papers address this topic, we use a standardized gateway as the nerve center of our solution, namely the standard defined by the Open Service Gateway Initiative (OSGI, [3]). The specification introduces a component-oriented approach, which becomes manifest in bundles. The usage of the open service gateway enables an abstract, almost protocol-independent view onto the different home automation protocols used. We are more interested in the mechanisms of *services*, rather than in the mechanisms of *serving*.

In this paper we will describe a new view on component-based development of Web-enabled eHome services. We target the following problem: There is an adequate framework for the development of state-of-the-art Web-enabled eHome services, but there is no knowledge about the system and service structure and its architecture in detail. We will propose an enhanced framework, which incorporates several established design ideas and show how this *cookbook* makes system architects' and developers' life easier.

At Department of Computer Science III at Aachen University of Technology, tools for supporting development processes have been built for software engineering [4], mechanical engineering, chemical engineering, process control, telecommunication systems, authoring support, and eHome systems. We derive tools automatically from specifications, using the PROGRES system [5] for specification development, a code generator for producing code out of the specification, and the UPGRADE visual framework environment [6] into which the code is embedded. The resulting tools are efficient demonstrators for proof of concept purposes. The basis for this work is our research on software architectures in the above application domains.

The paper is structured as follows: in section 2 we will discuss the general requirements for useful services that are accepted by customers and are therefore successful in market. Then related work is classified and the current state-of-the-art is shown. Section 4 gives an in-depth description of our approach. We will give insights of our service development, which is based on a project carried out within research and industry cooperation. The results of the different development cycles are discussed in section 5. In the last section, we summarize our results and give an outlook for future work.

2 General Requirements

Home automation promises new comfort and useful services in everyday life. These services will take place where today we find ubiquitous appliances. From users' point of view, services should be at least as easy in use. From developers' point of view, different *appliances and technologies* exist, which have to be integrated. This imposes specific requirements on the development of services in the area of home automation.

Home automation aims at mass markets. Therefore, technical background can not be expected from users. Mainly users will expect trivial *plug-and-play* from home automation appliances as they do from their legacy pendants. This leads to the demand for *self-configuring* and *maintenance-free* systems, which seems to be an infeasible step. All details of a home, where each device is connected to each other, can not be specified in advance. So, some configuration and setup will always be a manual task done. On the other hand, home automation can not require technicians come to the user's home to integrate any kind of devices to home networks.

Talking about Web-enabled eHome services, we are talking about homes or facilities connected to external communication and data networks (at least temporarily). We recommend eHome services designed to be *remote-configurable*. This seems to be fairly more realistic, where configuration tasks can be left after installation. If a service does not install and configure itself automatically and a user can not do it on her own, he can call her service provider to remote access her eHome system and do necessary setup. Of course, this implies special mechanisms to keep Web-enabled eHome service's *security and integrity*. User should be able to determine and monitor who may virtually access her eHome system. Privacy of eHome must not be compromised by new comfort.

Connecting home area networks with communication and data networks provides potential for many service ideas. Up to now, service remote configuration is the most popular. Furthermore, user may access her eHome using different communication and data networks. Appliances can be controlled from any place (e.g., the office) using a browser and the Internet. The owner of a eHome may determine and change state of his alarm equipment with the Wireless Application Protocol (WAP [7]) and mobile communication networks using his mobile phone. Or in case of an alarm, the alarm equipment sends a multimedia message (e.g., MMS [8] or eMail) to the owner of an eHome, who will receive the message with his mobile phone wherever he is. But Web-enabled eHome services may also interact on their behalf with other data and communication network services making value-added services possible.

Summarizing, developing Web-enabled eHome services does not only mean to explore and realize consumers' needs and wishes. Realizing eHome services, special requirements have to be met. First, our eHome system should be connected with different communication and data networks. Second, as new needs arise by users, new appliances or new services emerge, our system should be extensible with services, respectively. Third, eHome appliances need a way to communicate. At present, home automation standards and technologies have been extremely fragmented. To cope with that, either, a standard has to be accepted, which encompasses all existing technologies, or we need a modular and extensible system to integrate existing technologies. The latter would facilitate developing applications, which use modules implementing existing protocols and appliances, which overcome the differences on syntactical and semantical level. Later on we will discuss this topic in more detail.

Decoupling of services from underlying infrastructure is based on abstraction from the infrastructure itself. A Web-enabled eHome service should rely on types of devices, instead of specific devices and their proprietary implementation. For example, an alarm system should be able to integrate any motion detector or device which is able to detect motion (e.g., cameras), instead of being tightly bound to a vendor-specific motion de-

tector. Taking the back-end systems into account, a Web-enabled eHome service is not only a piece of software executed on the service gateway. A *Web-enabled eHome service* is the *wholeness of services a user experiences*. The solution has to ease the realization of distributed systems, which do not impose any further burdens to users.

Following the approach of a modular and extensible system, we find useful concepts in *component-based software development*. Current component technologies rely on an infrastructure, which allows communication between registered components [9]. Applied to eHome service development, each component implements an appliance or network protocol. These components act as proxies, representing an appliance or a network protocol, respectively. This leads to the integration of existing and upcoming standards and technologies at a higher level, where appliances communicate by means of the component technology's infrastructure not using a specific protocol.

Beyond, component-based software development has advantages, which can be turned to account developing eHome services. We expect software engineering benefits and economic benefits from component-based software engineering mainly through software reuse. Previous paragraph described an example with a proxy, which would be a Web-server component. Other components could register their static or dynamic resources using the HTTP-proxy interface. Rigorous separation of interface and implementation and the non-existence of side-effects is a paradigm inherent in component technologies. Thus, components providing necessary implementation of an appliance or protocol from third-parties may be used reducing time-to-market of new eHome services.

Furthermore, new devices and protocols can easily be integrated. We can not figure out to which technology electric and electronic systems will converge. Most probably, actual situation will continue and different technologies and standards will coexist. But as new appliances or home automation technologies emerge, they may be incorporated using the approach of a modular and extensible system architecture. Either, a component has to be developed or a component can be used off-the-shelf, implementing the technology or appliance functionality, respectively. As new user needs and wishes will be explored, services satisfying these may be developed simply by *composing* existing service components.

Building eHome services, we use components' functionality through the interfaces provided. Any component captures and hides details of its knowledge about a special device or technology within its implementation. Thus, a component-based approach developing eHome services permits to concentrate on service logic, keeping details of used appliances or protocols unconsidered.

Based on the concept of residential gateway, the *open service platform* specified by OSGi [3] describes an approach, which supports extensible development of Web-enabled eHome services integrating existing and upcoming technologies. The open services gateway comes with an infrastructure that lets registered components retrieve other components and invoke their services. This infrastructure, sometimes referred to as a component model, is common among component infrastructure technologies (e.g., OMG CORBA [10], Sun EJB [11], Microsoft DCOM [12]). It builds the foundation for component-based software engineering. Thus, an approach based on the OSGi service platform enables component-based development of Web-enabled eHome services.

3 Related Work

A wide variety of different technologies competing for the emerging home networking market has been developed and evolved. While we are looking for a solution that enables communication of appliances and devices, to date, home networking technologies have been fragmented addressing particular application domains. Attempts exist to provide a generic framework that permits integration of different appliances and network protocols. We will give a brief overview in this section.

Different home networking technologies exist, e.g. Bluetooth, CEBus, EHS, LonWorks, HAVi [13, 14, 15, 16, 17]. They describe higher-level protocols built on top of a physical media and its lower-level protocol. Each addresses a specific domain. Using different physical media and protocols, it is obvious that devices may not inter-operate across technology boundaries. Thus, stand-alone solutions will not be successful.

Other technologies attempt to be more generic and independent of any physical media defining a common framework enabling all compliant devices to inter-operate. *Jini* [18] uses Java APIs to enable a device to join a Jini federation and propagate its service without any setup. Other devices in the Jini community can look up and discover the service. *UPnP* [19] uses IP networks to let devices discover, utilize and control other devices. The services a device provides and how they are controlled is described by XML-encoded messages. *AutoHAN* [20] has much in common with UPnP. Through the use of an XML-based service registry, it is not bound to any one API. These approaches impose particular hardware requirements to its devices, to run software modules of specific technologies. Some devices can not satisfy these particular hardware requirements, why a concept of proxy objects is introduced. Each resource-constrained device is thereafter connected with a performant device that runs the proxy object.

The *CableHome* specification [21] proposes an extension to cable modems to enable the execution of services with respect to IP-based communication, security aspects, and quality of service (QoS). While the extension of legacy installations (e.g., cable modems, telephone systems) is reasonable, the efforts of the specification are in the field of IP routing and ensurance of QoS. The integration of device drivers and applications is not tackled. CableHome technology could be used as the communication backbone for the distributed IP-based platform.

The open services gateway initiative (OSGi) specifies a set of open-standard software application interfaces (APIs) for building open-service gateways on top of *residential gateways*. The residential gateway describes an universal appliance that interfaces with internal home networks and external communication and data networks. Similar to a data network gateway, a residential gateway is equipped with interfaces to different physical media and provides conversion between protocols. It represents a concept, which lets different networks communicate transparently. This leads to a wide support of various standardized technologies, as depicted in figure 2.

The OSGi gateway (see figure 3) resides within a Java runtime environment, which offers the well-known features of Java. The core component is specified as the *OSGi service framework*, which acts as a container for service implementations. This environment includes a Java runtime with life cycle management, persistent data storage, version management and a service registry. Services are Java objects implementing a concisely defined interface. Services are packaged within *bundles*, which register zero, one or more

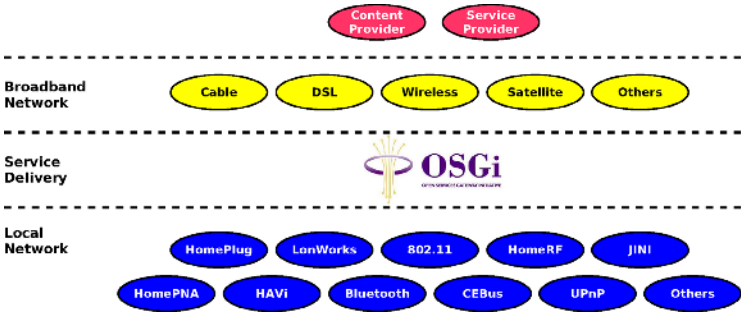


Fig. 2. OSGi and Related Standards

services within the framework's service registry. Bundles contain services implemented in the Java programming language, a manifest file describing import and export aspects, and additional implementation-specific libraries. Bundles can be deployed and undeployed during runtime, while the information in the manifest file ensures the integrity of the system. Security aspects are handled as well. As the figure shows, a bundle is not restricted to rely on functionality offered by the framework, it can profit from every layer below, i.e. native functionality offered by the operating system and the hardware. This stands in contrast to layered approaches in software engineering, but allows the realization of bundles for any protocol. To ensure a minimal common set of functionality, certain bundles are standardized (e.g., the `Log` bundle for logging and the `HTTP` bundle for user interface purposes).

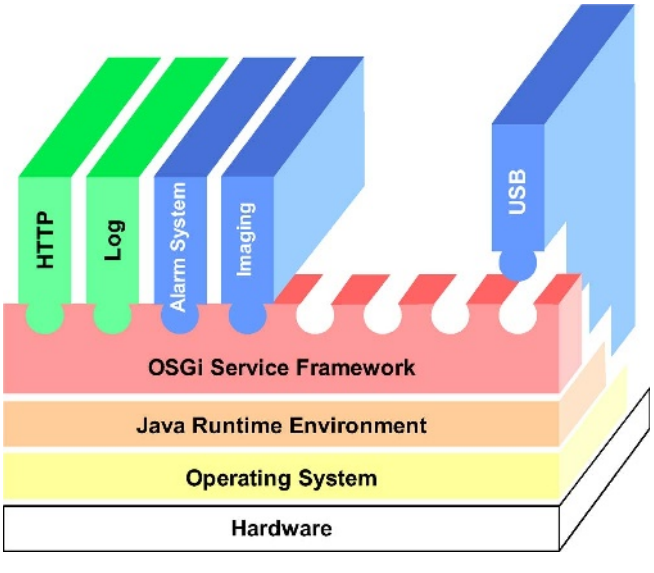


Fig. 3. OSGi system structure

Central to the OSGi specification [3] is the *service framework* with a service registry. Components register their services at the service registry where other applications may retrieve and use them. Based on the concept of residential gateway, the open services gateway specification describes an approach that permits coexistence of and integration with multiple network and device access technologies. In addition, components may be added implementing new technologies as these emerge. Interaction is enabled via Java interfaces, without relying on proxy objects. While other systems -like Jini- are decentralized, the OSGi approach is a centralized system, which simplifies the maintenance of the system.

The specification provides a concise and consistent programming model for Java bundle development, simplifying the development and deployment of services by decoupling the service specification (i.e., the Java interface) from its implementation. Therefore, developers of applications and services only need to know about the services' interfaces. This approach is well known in component-based software development. By separating a component's interface from its implementation, we follow the goal of loosely coupled service components. Service implementations may be modified without taking effect on client applications as long as they obey to their interface specification.

As we have seen, to date, there is no adequate home automation standard, which meets our requirements (cp. sec. 2). Each proposed standard is focused on one application domain or dependent of a proprietary infrastructure. Not rarely, approaches impose certain hardware requirements on devices, which not always are able to meet that. Thus, an approach incorporating a central processing unit seems feasible. The features of the open service platform come quite close to our requirements. For this reason, we choose the open service gateway platform (OSGi) as a basis for our approach.

4 Basic Approach

This section deals with our proposed approach to overcome the posed problems. As stated, we use an OSGi-conform service gateway as a basis for our solution. This gives a quite complete component model of basic infrastructure services and makes an additional component-based development easy and naturally. Furthermore, the device access mechanism provided by OSGi allows effective usage of the service registry, because devices are represented by services, too, and are discovered and represented within the framework automatically.

Following the idea of OSGi, each and every Web-enabled eHome service resides within a bundle not crossing the bundle boundaries. The deployment is fail-safe, because each bundle comes with a manifest describing the dependencies and requirements, restrict to the direct neighbors (1-context). Also, the mechanisms of the service gateway can be used for deployment during operation. This does not interfere with executed services (services are hot-pluggable). After installation, new services are announced via the service gateway and its message subsystem.

Each bundle may provide its own configuration user interface (UI) utilizing the (specified) HTTP-service. Besides that, for configuration data get- and set-methods are provided in order to be direct accessible from other bundles and components. The configuration data is of course isolated within the bundle. A third way for configuration

tasks can be realized: by implementing the interface ManagedService, bundles can be remotely configured by the module Configuration Admin, which is specified by OSGi, too. Configuration data are encapsulated in Property objects. Configuration Admin acts as a mediator. With that, basic configuration in terms of adjustments of properties can be realized.

Relevant for service and component development are two questions. Talking about component-based development, it is a design prerequisite to choose between (a) specialized components, which can not be used in other contexts, and (b) generic components, which can be used several times, but always with modifications necessary [22]. Applied to service development based on OSGi, the task is to strike the balance between insulation and sharing among bundles, which means that there is no reuse of basic components [23].

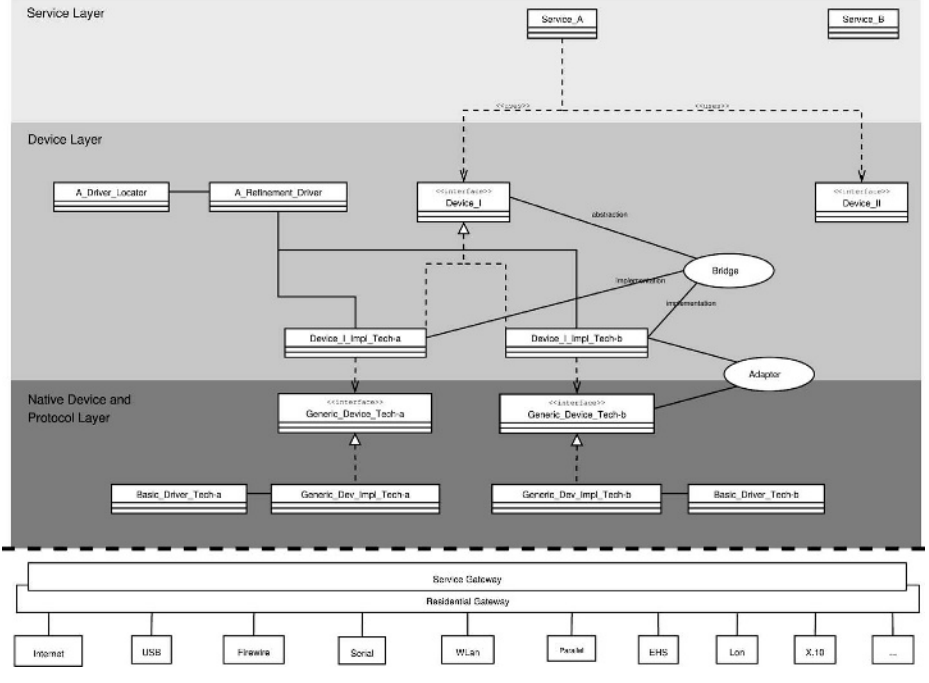


Fig. 4. Layered Service Architecture

In our dedicated approach for Web-enabled eHome service development, we propose a *3-layer system structure*, shown in figure 4. The lowest level comprises components mirroring functionality of native devices and network protocols. We named this layer *native device and protocol layer*. The middle layer is called the *device layer*. There the representations of device and protocols as *services* reside. These components wrap details visible at native device and protocol layer. This is also known as an structural design pattern called *facade* [24]. From an outside point of view, we have now an abstract view on the basis, the device infrastructure. OSGi's device access technology is integrated into both lower and middle layer, enabling to reflect the dynamic aspects of the eHome

system. Seamless integration of device access and protocol drivers provide means to reflects the *dynamic of the system*.

To meet the requirement of focusing on functionality instead of devices, we abstract from vendor-specific and device-specific implementations and used network infrastructure. We introduce interfaces for types of devices, which reflect a well-understood common functionality of devices. To incorporate existing technologies and devices, the specific interface has to be adjusted to the abstract interface. This enables the easy introduction of arbitrary devices and their implementations into the proposed system architecture.

The highest level provides room for sophisticated Web-enabled eHome services. These are the components developed by programmers, who now do not have to deal with device infrastructure details. They can concentrate on the abstract services, their functionality *and* requirements. Worth mentioning is, that services can not only be developed by turning lower levels to account, they can be build by composing high-level services, too. This brings developers in a quite comfortable position.

As an important design guideline a basic approach from component-based software engineering must be applied. Designing the architecture of eHome services must observe that dependencies between components may be described as a directed, acyclic graph. This enables services to be realized incrementally. Otherwise every part has to be realized to make it run. Beyond, the structure follows a logical design and becomes more easy to maintain.

In our approach, there is no need for generic bundles, which act as a container for traditional object-based realizations. Though there are approaches in this direction like Java Dynamic Management Kit [25], we think that this would be unnatural and makes the component-based basis nearly senseless. The separation of concerns is expressed by the bundle boundaries. Only basic services are shared among bundles, specialized tasks are captured within bundles.

The proposed layered system architecture describes a model for partitioning and integration of Web-enabled eHome services. With this model, a structure is introduced, which separates the application logic of Web-enabled eHome services from infrastructure details.

5 The Real World

Foregone we had a development cycle of Web-enabled eHome services based on a proprietary platform. Gathered experiences rose needs for a modular and extensible system. Therefore, extensibility had been identified as a must to integrate other home automation technologies and incorporate upcoming devices and services. Starting from the OSGi platform, the goal has been to gather experiences with a component-based approach redesigning services that had been developed based on the proprietary platform. Because of few experiences with component model, framework, and specified services of the OSGi service platform, a prototyping process has been chosen. The prototyping process has been applied to two development cycles. Within first development cycle, the service *PowerImage* has been developed. PowerImage is an imaging service, which provides access to imaging devices and offers additional functionality (e.g., an image

archive). Within second development cycle *PowerSafe* has been developed. PowerSafe utilizes PowerImage, sensors (e.g., motion detectors), actors (e.g., lamps and sirens), and an eMail service to realize an alarm system. With two development cycles, information gathered from the first can be applied to the second. Possible design mistakes can be adjusted.

From external view, *PowerImage* and *PowerSafe* appear as two services. Following our basic approach, these two services consist of three components. Figure 5 illustrates the identified structure. The service *PowerImage* is described by a PowerImageDevice interface. This interface builds the facade to different picture grabbing devices, i.e. webcams from different manufacturers and different device access technologies. There is no component PowerImage as might be expected. PowerImageDevice provides all functionality of the service *PowerImage*. The representation completes the service implementation.

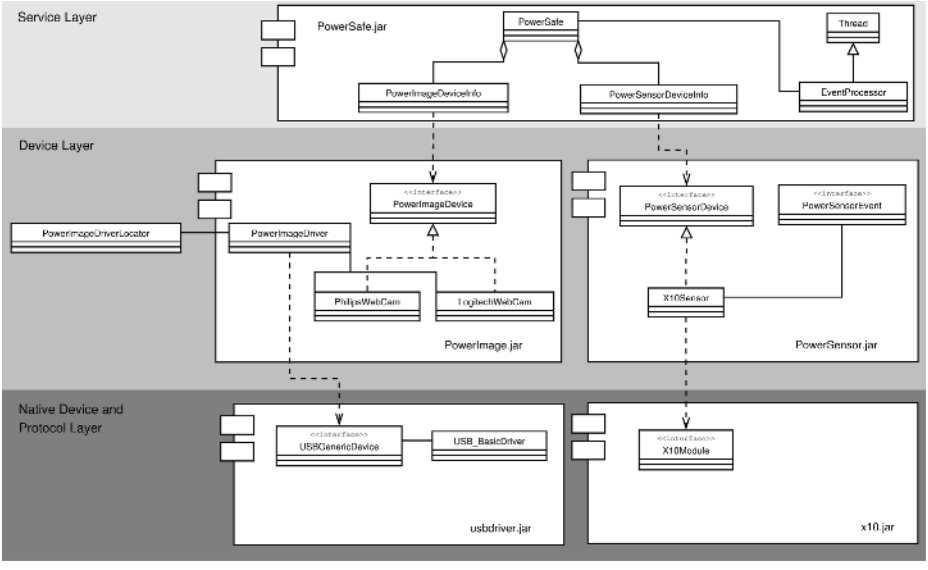


Fig. 5. Structure of PowerImage and PowerSafe according to Basic Approach

PowerSafe is the second component, that represents an alarm system. Different cameras can be connected to the alarm system. The component PowerSafe is placed at the service layer. It makes use of a third component called PowerSensorDevice. PowerSensorDevice is the second device facade at the device layer. It hides details of underlying sensors (e.g., motion detectors, cullet sensors, and fire detectors). Different sensors may be connected to the alarm system and send notifications to it in case an event occurs.

The separation of PowerSafe and PowerSensorDevice reflects our approach of distinguishing device layer and service layer. PowerSafe represents an alarm system composed of different devices. In contrast, PowerSensorDevice represents a generic interface for different sensor devices. They may be used in other contexts than alarm systems, e.g. Ser-

vice A (see figure 5) could be another service using PowerSensorDevice to switch on lights.

For development of PowerImage and PowerSafe a test bed was needed. USB web-cams have been chosen as PowerImageDevices. X.10 motion detectors have been chosen as PowerSensorDevices. A personal computer running Linux is used as residential gateway running an implementation of the OSGi service platform. The web-cams are directly connected at the USB-ports. X.10 devices are accessed through a X.10 module, that is plugged in a power socket and connected with the residential gateway's serial port.

For a deeper understanding of the three layers, we will describe them in detail in the next subsections.

5.1 Native Device and Protocol Layer

The introduction of device layers follows concepts of the OSGi platform. The OSGi *Device Access* specifies a common concept for management and automatic detection of devices in an OSGi environment. By the use of Device Access, a device service is automatically registered at or removed from the service registry as a device is attached to or disconnected from the residential gateway.

Base drivers provide the lowest-level representation of physical devices. Therefore, we associate them to the native layer. They detect if a device is connected to or removed from the residential gateway. Respectively, they register a generic device of the corresponding device access technology.

Device services registered by base drivers encapsulate implementation details of specific devices. They come with an API to access the functionality they provide.

Beyond device services, network services are located at the native device and protocol layer. Like device services, they provide an API hiding protocol details from their clients. For example, figure 5 shows USB as a device access technology and Lon, X.10, and EHS as home networking protocols at the native device and protocol layer.

5.2 Device Layer

OSGi Device Access specifies another category of drivers, the so called *refining drivers*. After a generic device service for a newly detected device has been registered, the OSGi environment searches for refining drivers. Each refining driver found is questioned, if it provides a refined service representation for the generic device. If the refining driver can match the generic device, it registers a refining device service in the OSGi environment. This refinement process is repeated until no other matching driver is found. To determine the process, a ranking is used to identify the best match.

Following the basic approach, refining drivers are located at the device layer. Base drivers are packaged within components of native device and protocol layer. Refining drivers are packaged together with interfaces and implementations they know from device layer (e.g., PowerImageDriver is bundled with the interface PowerImageDevice and the implementing class PhilipsWebCam).

We used a USB web-cam for development of *PowerImage*. Normally, a USB base driver is a component that could be used off-the-shelf. But no USB base driver bundle could be found that was applicable. Therefore, a USB base driver had been developed,

wrapping an available Java USB API. This driver registers generic USB devices within the OSGi environment as new USB devices are plugged in at the residential gateway. After that, the refining drivers are invoked by Device Access, i.e. PowerImageDriver is invoked. This driver has knowledge about the class PhilipsWebCam, which is an implementation of the interface PowerImageDevice *and* at the same time is a refined view of a generic USB device. This way a PowerImageDevice is registered at the OSGi environment without requiring additional configuration.

The OSGi environment provides notifications to inform interested components about occurring environment events. OSGi specifies three kinds of events: FrameworkEvent, BundleEvent and ServiceEvent. The OSGi environment propagates ServiceEvents, when services are registered at and unregistered from the OSGi service registry and when service properties have changed. Base drivers remove the generic device, if the corresponding device is removed from the OSGi environment. Therefore, the event mechanism can be used to unregister refining devices, if the corresponding generic device is unregistered. PowerImageDriver listens for service events, so it can unregister PhilipsWebCam objects it registered before.

The component PowerSensor developed within the second development cycle uses X.10 motion detectors. We used an OSGi-based X.10 component that implements the X.10 protocol. X.10 devices do not provide a discovery mechanism. Therefore, we choose an alternative proposed by the OSGi specification: we developed a Web-frontend for management and configuration of X.10 sensors. Through the use of Web-based user interfaces, X.10 motion detectors can be registered at, configured for, and removed from the OSGi environment.

5.3 Service Layer

The service PowerSafe is located at the service layer. First, because it represents an abstract view of an alarm system and does not represent a device connected with the residential gateway. Second, it is a service build by composing components from device layer. With our realization of an alarm system, different PowerImageDevices and PowerSensorDevices can be connected.

In contrast to traditional library-based software environments, the OSGi service platform is rather a dynamic runtime environment [23]. During run-time, services used by other services may come and go. Especially, as PowerSafe is composed of services from device layer, these may appear and disappear during runtime, as devices are connected or disconnected. Therefore, the pattern *ServiceMonitor* [23] has been used and has been further extended.

ServiceMonitors encapsulate dynamics of an OSGi environment using the event mechanism. Clients within a component do not use services from other components directly. They use external services through a ServiceMonitor, which manages the corresponding service reference. Beyond, the utilization of ServiceMonitors has been extended to manage tasks within a component when specific events occur. For example, servlets generating the UI are automatically registered after the HTTP-service becomes available.

5.4 Lessons Learned

Basically, the OSGi service platform meets our basic requirements for development of Web-enabled eHome services. Beyond, its framework provides services useful for cooperating components. Developing PowerImage and PowerSafe, we gathered experiences with development of Web-enabled eHome services.

While some technologies provide a discovery mechanism, some do not. The latter does not allow to detect connected devices automatically. Thus, maintenance-free systems are rather unrealistic, if legacy home automation technologies should be integrated. We identified a feasible alternative by the use of browser-based HTML-user-interfaces. If user does not cope with setup, operators may access the system remotely and configure *PowerSensorsDevices* for example.

The *Configuration Admin Service* specified from OSGi describes how operators can set configuration data of deployed components. Configuration data are stored persistently so they can be retrieved after the OSGi runtime-environment is restarted. Unfortunately, it allows only character strings to be used. If necessary, this can be drawn aside by using Java serialization [26]. But the trade off between design advantages and an approved configuration admin service implementation has to be evaluated.

Regardless, the OSGi environment meets our requirements for development of Web-enabled eHome services. Beyond it, the OSGi service platform provides mechanisms that support our basic approach, while the layered architecture reaches software engineering goals.

6 Summary and Conclusion

We have shown, that by using our approach the required design efforts can be split into handy tasks. Once infrastructure questions are answered, the design and development process can go one step further. The abstract view of the component model and basic services gives not only room for free thinking, it contributes to the tasks of saving investments by making reliable, approved, and maintainable solutions possible. The described development of the services PowerImage and PowerSafe reaches the described goals.

As other approaches (e.g., Jini) identified the update and maintenance tasks as a severe problem, we do feel confident, that the proposed 3-layer system structure reflects reasonable levels of abstraction. These levels can be associated with different roles, e.g. the native device and protocol layer can be maintained by manufactures or organizations for standardization. Components at the device layer will usually be provided by service gateway providers. Components at the service layer will be provided and marketed by service providers. Each layer has a clear-cut interface, so every component of a certain layer can be exchanged and new ones added without destroying the operable condition of the service gateway. The proposed architecture focuses on the a-priori design phase and the reegineering of Web-enabled eHome services. Third party services can be incorporated by the means of OSGi or the distributed IP-based service platform [27].

Thus, the described solution brings together the benefits of the OSGi open services gateway and of component-based software engineering. We proposed guidelines, how sophisticated systems of the described nature can be build. Following this idea, we

expect that new abstract services can be composed by other service's implementations. Through the deployment features of the service gateway, installation and setup is nearly as easy as plugging in an appliance. We expect an eased realization of Web-enabled eHome services, which are (a) remote-accessible, (b) remote-configurable, (c) remote-maintainable, (d) safe and secure, (e) fail-safe, (f) affordable, and (g) incorporating existing devices and infrastructures.

Based on the current development, several areas of future work have been identified. First, basic services may be identified to provide designers and developers a complete set of instruments. It has be to clarified, which of them should be proposed as an enhancement of the OSGi specification. Second, there are security issues. Monitoring of access and associated evaluation, placement and extent of personal data and configuration data are topics of interest crucial for the success of such systems. Furthermore, concurrency and synchronization of data has to be researched. A more sophisticated approach for dealing with arbitrary data in such systems is currently being developed. Third, the dependencies between gateways and back-end systems have to be examined. Furthermore, it has to be evaluated, in which other domains the results gathered can be applied.

Acknowledgements

The work presented here is part of a project carried out at the Computer Science III group at the Aachen University of Technology, Germany. We are especially indebted to Mr. Peters (RWE AG, Essen, Germany) and Dr. Pritsch (Booz, Allen, and Hamilton, Düsseldorf, Germany) for their engagement in this project.

References

1. Becker, S., Kirchhof, M., Nagl, M., Schleicher, A.: EAI, Web und eBusiness: Echte Anwendungsintegration macht Aufwand! In: Proceedings of Online '02. Congress VI (2002) C630.01–C630.27
2. L. M. C. Matos, ed.: Collaborative Business Ecosystems and Virtual Enterprises . Volume 213 of IFIP International Federation for Information Processing. Kluwer Academic (2002)
3. Open Services Gateway Initiative: OSGi Service Platform. (`http://www.osgi.org` (13.11.2003))
4. Nagl, M., ed.: Building Tightly Integrated Software Development Environments: The IPSEN Approach. LNCS 1170. Springer, Heidelberg (1996) ISBN 3-540-61985-2.
5. Schürr, A.: Operationales Spezifizieren mit programmierten Graphersetzungssystemen. PhD thesis, RWTH Aachen (1991)
6. Böhlen, B., Jäger, D., Schleicher, A., Westfechtel, B.: UPGRADE: Building Interactive Tools for Visual Languages. [30] 17–22
7. WAP-Forum: Wireless Application Protocol. (`http://www.wapforum.org`)
8. 3GPP: Multimedia Messaging Service, TS 22.140. `http://www.3gpp.org` (2002)
9. Brown, A.W., Wallmann, K.C.: The current state of CBSE. IEEE Software (1998) 37–46
10. Object Management Group, Inc.: The Common Object Request Broker: Architecture and Specification, Revision 2.6.1. (2002) `http://www.omg.org` (14.6.2002).
11. DeMichiel, L., Yalcinalp, L., Krishnan, S.: Enterprise Java Beans Specification, Version 2.0. (2001) Sun Microsystems, Inc.

12. Brown, N., Kindel, C.: Distributed Component Object Model Protocol DCOM. Microsoft Corporation (1998)
13. Bluetooth SIG, Inc.: Specification of the Bluetooth System. (2003) https://www.bluetooth.org/foundry/adopters/document/Bluetooth_Core_S%pecification_v1.2 (16.5.2004).
14. CEBus Industry Council: EIA 600 Specification. http://www.cebus.org/ (1996)
15. EHSA: Home Systems Specification (EHS). http://www.ehsa.org (2000) Release 1.3a.
16. Echelon Corporation: Cea-709.1-b: Control network protocol specification (2002)
17. HAVi Inc.: HAVi 1.1 Specification of the Home Audio/Video Interoperability Architecture (2001)
18. Waldo, J.: The Jini Architecture for Network-Centric Computing. Communications of the ACM **42** (1999) 76–82
19. UPnP Forum: UPnP Specification Documents. (2004) http://www.upnp.org/standardizeddcps/default.asp (16.5.2004).
20. Saif, U., Gordon, D., Greaves, D.J.: Internet Access to a Home Area Network. IEEE Internet Computing (2001) 54 – 63
21. Cable Television Laboratories, Inc.: CableHome 1.1 Specification. http://www.cablelabs.com/projects/cablehome/downloads/specs/CH-SP-CH1.1%-I03-040129.pdf (2004)
22. Szyperski, C.: Component Software. 2 edn. Addison Wesley (2002) ISBN 0-201-74572-0.
23. Chen, K., Gong, L.: Programming Open Service Gateways with Java Embedded Server Technology. Sun Microsystems (2001)
24. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
25. Sun Microsystems, Inc.: Java Dynamic Management Kit White Paper. http://www.sun.com/products-n-solutions/nep/software/java-dynamic/wp_jdmk40.pdf (2000)
26. Sun Microsystems, Inc.: Java Object Serialization Specification. http://java.sun.com (1999)
27. Kirchhof, M.: Distributed and Heterogeneous eHome Systems in Volatile Environments (2004)
28. Heineman, G.T., Councill, W.T.: Component-Based Software Engineering. Addison-Wesley (2001)
29. Gruhn, V., Thiel, A.: Komponentenmodelle DCOM, Javabeans, Enterprise Java Beans, CORBA. Addison-Wesley (2000) ISBN 3-827-31724-X.
30. Callaos, N., Hernandez-Encinas, L., Yetim, F., eds.: Proceedings of the 6^{th} World Multiconference on Systemics, Cybernetics, and Informatics (SCI02). Volume I (Information Systems Development I)., Orlando, Florida, USA, IIIS (2002)

Author Index