kassel
university
press

Proceedings of the
Second Workshop on

Context Awareness for Proactive Systems CAPS 2006

Kassel, Germany, June 12-13, 2006

Editors:
Klaus David, Olaf Droegehorn
and Sandra Haseloff

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar

ISBN-10: 3-89958-210-1
ISBN-13: 978-3-89958-210-9
URN: urn:nbn:de:0002-2102

Umschlaggestaltung: Bettina Brand Grafikdesign, München
Druck und Verarbeitung: Unidruckerei der Universität Kassel
Printed in Germany

Preface

The volume at hand contains the tutorial description, keynotes, and papers of the Second Workshop on Context Awareness for Proactive Systems (CAPS 2006). After the successful start of the CAPS workshop series in Helsinki, Finland, in 2005 the aim of the CAPS 2006 workshop was to reinforce a platform for an exchange of research findings and ideas as well as for inspiring discussions concerning the manifold challenges context-aware proactive computing is still faced with.

CAPS 2006 continued to deal with the question of how to enable computer systems to become aware of and suitably adapt to their environment and the situation of their users. This fundamental challenge was addressed in four workshop sessions each of which was concerned with a particular aspect of context-aware proactive systems: the impact of context data on user needs, context data representation and management, a context-dependent, proactive supply of information and services, and architectures for context-aware proactive systems.

The CAPS 2006 workshop was held at the University of Kassel, Germany, on June 12th and 13th. It was organized by the University's Chair for Communication Technology (ComTec). The organizing committee wishes to thank the initiators of the CAPS workshop series, the members of the Program Committee as well as all volunteers who helped with the organization of the event for their support.

Kassel, June 2006

Sandra Haseloff
Klaus David
Olaf Drögehorn

Organization

CAPS 2006 was organized by the Chair for Communication Technology (ComTec) of the University of Kassel, Germany.

Organizing Committee

Klaus David, University of Kassel, Germany
Olaf Drögehorn, University of Kassel, Germany
Sandra Haseloff, University of Kassel, Germany
Holger Konhäusner, University of Kassel, Germany

Program Committee

Klaus David, University of Kassel, Germany (PC chair)
Heikki Ailisto, VTT, Finland
Guy Bernard, Institut National des Telecommunications, France
David Bonnefoy, Motorola, France
Olaf Drögehorn, University of Kassel, Germany
Patrik Floréen, Helsinki Institute for Information Technology HIIT, Finland
Stefan Gessler, NEC, Germany
Sandra Haseloff, University of Kassel, Germany
Heikki Helin, TeliaSonera, Finland
Theo Kanter, Ericsson, Sweden
Mika Klemettinen, Nokia, Finland
Herma van Kranenburg, Telematica, The Netherlands
Martti Mäntylä, Helsinki Institute for Information Technology HIIT, Finland
Bernd Mrohs, Fraunhofer FOKUS, Germany
Kimmo Raatikainen, University of Helsinki, Finland
Kurt Rothermel, University of Stuttgart, Germany
Matthias Wagner, DoCoMo Euro-Labs, Germany

Table of Contents

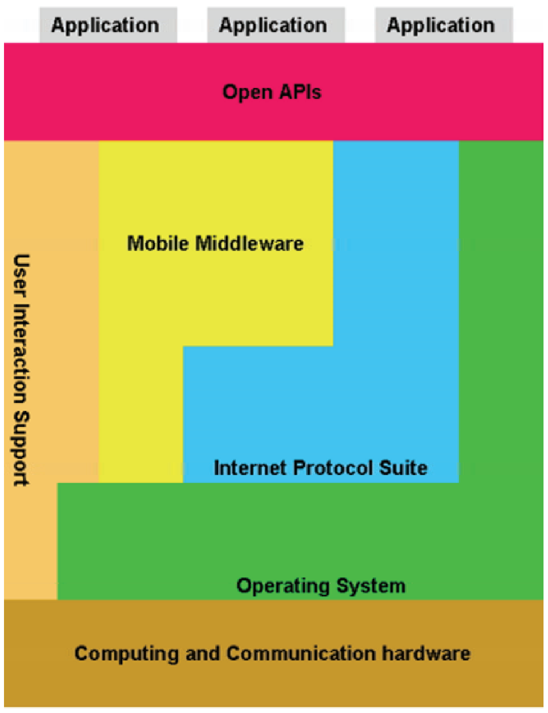
TUTORIAL

Recent Developments in Middleware Standardization for Mobile Computing

Kimmo Raatikainen

University of Helsinki, Department of Computer Science
P.O. Box 68, FI-00014 Finland
{firstname.lastname}@cs.helsinki.fi

Abstract. When a newcomer tries to find out relevant standardization forums of mobile middleware, he soon finds out to be in a jungle of forums. He faces a bunch of acronyms like IETF, 3GPP, 3GPP2, WWRF, OMG, OMA, W3C, WS-I, FIPA, JCP, DLNA, TCG, OSGi, UDDI, OASIS, UPnP Forum, Liberty Alliance. The tutorial gives a map of standardization activities relevant to mobile middleware.



The attached figure outlines the software architecture of a solution stack. The primary constitutes are Operating System, Internet Protocol Suite, Mobile Middleware, and User Interaction Support. An application obtains the services of these entities through common Application Programming Interfaces (APIs). If the applications would be stand-alone, then only Open APIs need to be standardized. However, most of the applications in the wireless world are networked and/or distributed. Therefore, we also need standard protocols that the applications use in their interactions. We also need standardization in mobile middleware, since generic service elements interact with generic service elements in other hosts. For us mobile middleware consists of an execution environment and a set of generic service elements like discovery and event notification.

Most of the standard protocols become from the ***Internet Engineering Tasks Force*** (IETF). The situation in generic service elements is much more confusing, since there are tens of forums that specify competing and complementing middleware services.

The IETF is a huge open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual. The actual technical work of the IETF is done in its working groups, which are organized by topic in several areas. The challenge is how the numerous Internet protocols are incorporated to the middleware.

In addition to IETF (http://www.ietf.org/) there are several forums that a mobile middleware researcher/developer needs to be aware. ***Third Generation Partnership Project*** (3GPP; http://www.3gpp.org/) and ***Third Generation Partnership Project 2*** (3GPP2; http://www.3gpp2.org/) are forums that specify 3rd generation mobile telephony systems. From middleware perspective they do no create anything new, which is a good piece of news, but select existing standardized solutions like a set of Internet protocols and ***Parlay*** (http://www.parlay.org/) Open Service Architecture. The ***Wireless World Research Forum*** (WWRF; http://www.wireless-world-research.org/) is a prestandardization forum addressing issues related to the 4G or Beyond 3G Systems.

Object Management Group (OMG; http://www.omg.org/) is the forum specifying CORBA and its extensions, and UML. In the CORBA realm, there are not very much going on related to mobile middleware. Recent specifications of Wireless CORBA (OMG document: formal/2003-03-64), Super Distributed Objects (OMG document: dtc/2003-04-02) and Smart Transducers (OMG document: formal/2003-01-01) are useful in wireless environments. The Model Driven Architecture (MDA; http://www.omg.org/mda/) may be the next silver bullet in software development, also for the wireless world. In MDA the target is to raise the abstraction level of software development: modelling instead of programming. The essence is code generation directly from the specification.

Open Mobile Alliance (OMA; http://www.openmobilealliance.org/) specifies service enablers for the mobile world. The core of OMA work is inherited from the forums (SyncML initiative, Wireless Village, Location Interoperability Forum - LIF, WAP Forum, Mobile Wireless Internet Forum - MWIF, and Mobile Gaming Interoperability Forum - MGIF) that merged into OMA.

World Wide Web Consortium (W3C; http://www.w3.org/) has several activities that are relevant to wireless world. These include Web Services Activity, Device Independence Activity, and Semantic Web Activity. In the Web Services Activity, there are working groups for Web Services Architecture, XML Protocol (SOAP), Web Services Description, and Web Services Choreography. The Device Independence Activity has recently published Device Independence Principles. There is also the document

CC/PP[1]: Structure and Vocabularies. In the Semantic Web Activity, there are RDF Core Working Group and Web Ontology Working Group.

Some aspects of service description are also addressed in ***Organization for the Advancement of Structured Information Standards*** (OASIS; http://www.oasis-open.org/), particularly in Universal Description, Discovery and Integration (UDDI; http://www.uddi.org/). ***Foundation of Intelligent Physical Agents*** (FIPA; http://www.fipa.org/) has Networking Ontology specification that provides the means to describe properties of connectivity. FIPA has also developed Device Ontology Specification.

Java Community Process (JCP; http://jcp.org/) has recently produced several JSRs increasing the functionality of Java 2 Micro Edition (J2ME). The primary targets of ***Open Services Gateway Initiative*** (OSGi; http://www.osgi.org/) specification are digital and analog set top boxes, service gateways, cable modems, consumer electronics, PC s, industrial computers, cars and more.

The ***Liberty Alliance Project*** (http://www.liberty-project.org/) has the mission of serving as the premier open alliance for federated network identity management and services. It ensures interoperability, supports privacy and promotes adoption of its specifications, guidelines and best practices. Liberty Alliance Version 1.1 Specification Suite incorporated Federated Authentication to enable seamless sign-on. Liberty Alliance Version 2.0 Specification Suite will feature Web Service Framework, authorizing e.g. a Service Provider to access your location information.

In addition, there are also ***UPnP™ Forum*** (http://www.upnp.org/) addressing discovery and autoconfiguration, ***Web Services Interoperability Organization*** (WS-I; http://www.ws-i.org/) promoting Web Services Interoperability across platforms, operating systems and programming languages, ***Trusted Computing Group*** (TCG; https://www.trustedcomputinggroup.org/home) specifying security in the network layer, and ***Digital Living Network Alliance*** (DLNA; http://www.dlna.org/) ensuring interoperability of consumer electronics, personal computers and mobile devices in the home.

To conclude, the standardization landscape is quite a mesh. The IETF covers the Internet Protocol Suite but has a vast number of working groups. The middleware services are standardized here and there including tens of forums. The major forums for wireless world seem to be Open Mobile Alliance, W3C, and Liberty Alliance.

[1] Composite Capabilities/ Preference Profile

KEYNOTE PAPERS

Context Awareness Targeting User Needs

Herma van Kranenburg, Johan Koolwaaij, Ingrid Mulder, and Henk Eertink

Telematica Instituut, Brouwerijstraat 1, Enschede, the Netherlands
{Herma.vanKranenburg, Johan.Koolwaaij, Ingrid.Mulder, Henk.Eertink}@Telin.nl

Abstract. Mobile services and applications need to be able to react on changes in the end-user's context, such as available resources, user preferences, user environment and situation. Context aware support functions assist in such tailoring and should preferably not only allow applications to adapt to (predicted) changes in the user context but also anticipate user intentions and goals. This advances pro-active systems into pragmatic systems. The paper gives an overview of this vision and examples of our context aware framework and applications.

Keywords: context awareness, user centric design, pro-active responsiveness, plausible user assistance

1 Introduction

Tailoring applications to the needs and situation of mobile end-users both increases user appreciation and is a potential business enhancer. Satisfied users are likely to accept and to pay for added-value personalized applications, tailored to their needs and circumstances, always choosing the optimal communication means and serving the user with appropriate assistance, while user disappointment eventually leads to not-using such badly tailored applications. This gives the motivation to research and develop functionality that enables applications to be 'aware' of the environment they are working with. In general such functionality is coined 'context awareness'.

We follow the definition by Dey [1] for context: being any information that can be used to characterize the situation of an entity (where an entity can be a person, place, physical or computational object that is considered relevant to the interaction between a user and an application, including the user and applications themselves). In addition we consider context from a user-centric perspective. Context-aware applications use context to provide task-relevant information and/or services to a user [1,2]. Hence a context aware system provides applications with the appropriate and relevant context information that enables them to adjust optimal to the user situation.

One step beyond are ubiquitous attentive systems and pro-active systems that even enable applications to timely react on upcoming changes in context hence on beforehand changes have occurred [3]. Examples of ubiquitous attentive system responses are predicting a train-traveller will arrive in his destination-city in 5 minutes, push the bus-time schedule to him as well as information on the bus departure platform. In many pro-active systems we witness the push of potentially interesting services and applications in combination with the predicted next user location, for instance based on user preferences or application usage of users from the same age, gender, etcetera. For truly user-centric tailored applications, the attentiveness or pro-activeness should be targeted at for the user plausible and usable results, recommendations, and contextual parameter-set predictions [4]. Such pragmatic systems support the user's intentions, his/her plausible needs and meant goals. Feedback from our context aware validators is and will be used to better understand and properly design context aware systems and in particular realize our vision of supporting users with plausible and usable results.

In this paper we will explain our vision – from reactive via pro-active towards pragmatic context aware system support. Our context management framework will be described and we will show examples – including lessons learned, from some of our context aware applications.

2 Vision

2.1 Evolution from Personal Service Environment to Ubiquitous Attentiveness

Our vision on tailored mobile applications and a user-centric system support started with the Personal Service Environment [5]. This vision, depicting a user surrounded by communication means, resources, terminals and applications nowadays is still applicable. For instance in a recently started FP6 IP [6] one of the principles is a distributed communication sphere, where a user with his terminal is considered in relation to the resources and other users within reach of him. Our PSE could be seen as a shell surrounding the end-user and taking care of discovery, negotiation, and adapting of resources and applications to the end-user and his dynamic situation. We extended the notion of a PSE into a concept coined ubiquitous attentiveness [3]. Ubiquitous attentiveness refers to the combination of 'ubiquitous computing' (i.e. lots of devices with communication capabilities), 'information systems that provide context information' and 'pro-active responsiveness' of applications. This pro-active responsiveness feature means that applications can timely react on upcoming changes in the context of the end-user, including the network and the context of any of the services that are used. A (wearable) system is ubiquitously attentive when it is cognizant and alert to meaningfully interpret the possibilities of the contemporary user space and intentionally induces some (behavioral/ system) action(s). Awareness of the contextual parameters that are relevant for the user, exchanging this information across and between (heterogeneous) domains and the pro-active responsiveness of the ubiquitous environment are thus important ingredients that are more elaborated on in [3]. An example of a ubiquitous attentive system response is predicting the next user location or next user-activity and pro-actively pushing route-assistance or entertainment services to him. An example on access network level is: while coming into reach of several Wi-Fi hotspots, instead of triggering a handover to the first sollicitated one, by taking into account additional context parameters, like travelling trajectory and predicted overlapping period with certain access network a handover to a more favorable upcoming access network will be initiated.

2.2 Pragmatic Systems Targeting Usable and Plausible Support

We envision that for truly user-centric tailored applications, the attentiveness or proactiveness should be targeted at for the user plausible and usable results that not only takes into account the (predicted) changes in the (mobile) end-user environment, but also anticipates end-user's intentions and goals [4]. We called such systems pragmatic. An example illustrating this is an end-user pointing at a tv-screen, asking "brighter please". A correct system response would be to make the screen brighter, however this would deteriorate the screen settings; while a usable system response would be to turn down the room lights or close the curtain, leading to a perceived brighter screen. Hence we expect in a pragmatic system that plausible and usable information and responses prevail over correct ones, and as such enhance and improve pro-active tailoring of applications and services.

To realize a pragmatic system, one needs to overcome many challenges. These both include understanding user intentions and goals and translating these into machine readable input as processing this input together with the contextual parameters. The processing functionality ('reasoners' in our context management framework, see section 3.1) in pragmatic systems will heavily rely on prediction techniques, reasoning techniques capable of processing vague and uncertain information, user behavior analysis modules, recommender and learner components [4]. Pragmatic reasoner components are needed that learn from user behavior and extend their knowledge base with learned rules for certain usage situations. Domain-specific knowledge bases are part of so-called expert systems; computer programs that provide solutions to search problems or give advice on intricate matters by making use of reasoning mechanisms. Expert systems can emulate human reasoning using appropriate knowledge representations, they can learn from past experiences by adjusting the reasoning process to follow promising tracks discovered on earlier occasions, and they can apply rules of thumb ('heuristics') to 'guess'. These systems can make use of logical rules, probabilistic reasoning techniques and mechanisms that can handle uncertainty, as well as use semantic web technology. Concepts like beliefs, goals, intentions, events and situations, often including the use of ontology make these methods suitable for pragmatic systems. Proper inter-working of the behaviour analysis modules, learning components and prediction engines is, of course, required.

3 Examples of Context-Aware Applications

In this section we will describe various context-aware applications we developed in different projects. We intend to use the feedback from our context aware validators to better understand and properly design context aware systems and in particular realize our vision of supporting users with plausible and usable results.

3.1 Context Management Framework

We designed and realized a context management framework [7] that we can extend in a flexible manner. Our system comprises components taking care of sensing, collecting and processing contextual information – the Context-Source components in Fig 1 – and discovering and providing access to sources of contextual information – the Context-Broker component.

In gathering contextual information we use wrapper components. For example a messenger wrapper, which connects to Messenger software and delivers information on the status of a user on his PC or laptop and a GPS location wrapper, which is a wrapper around a Bluetooth-enabled GPS receiver connected to a Symbian phone. The role of the Context-Broker is to provide access to sources of contextual information, for which it uses a Context-Source Registry. The function of the Context-Source is to provide for relevant context and triggers by monitoring the environment. Relevant context relates to context processing challenges such as the information should be at the right semantic level, delivered at the right moment, in the requested format, conflicts resolved, etcetera. A challenge in itself is to deduce by the system (hidden) user intentions and goals; what constitutes usable and plausible and how should the system interact in a non-obtrusive way appreciated by the user.

Often the requested information is not straightforward available, and hence available information needs to be reasoned on, interpreted, combined with information from other sources, and entailed information needs to be inferred. Such derived contextual information is viewed upon as a new Context-Source which is essential in our architecture: different layers of Context-Sources can be stacked on top of each other; calling on each others interface thereby constituting a dependency hierarchy or constellation of Context-Sources.

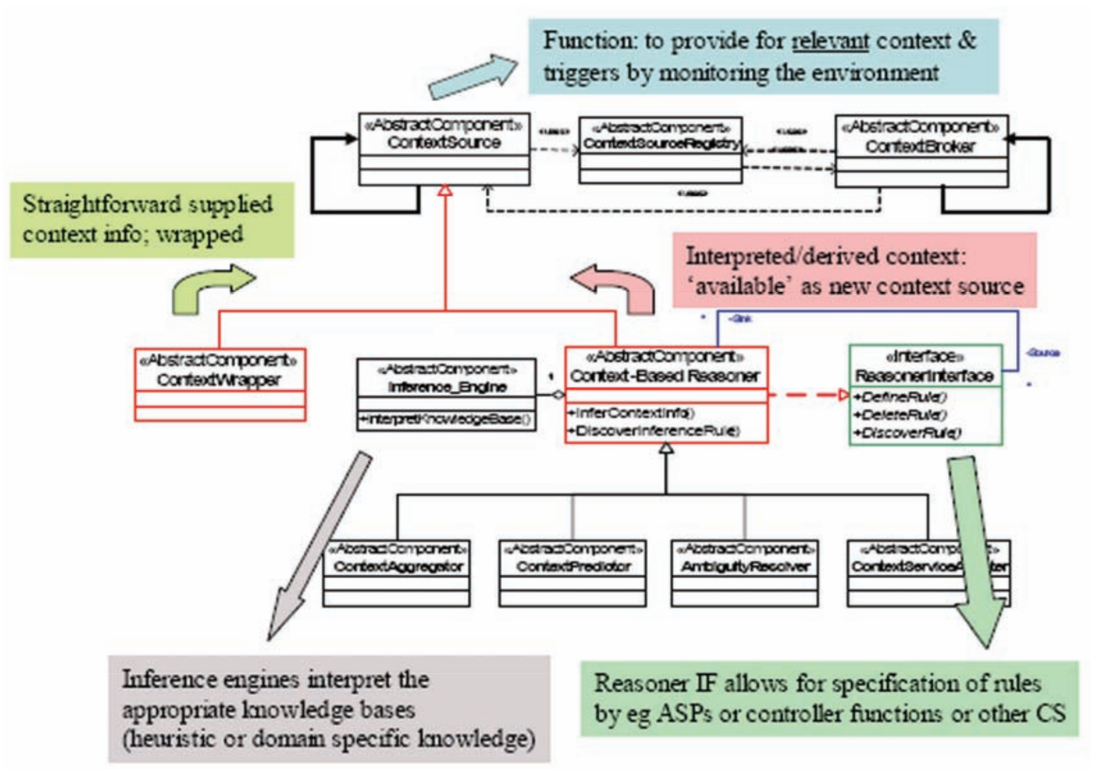

Function: to provide for relevant context & triggers by monitoring the environment
«AbstractComponent» ContextSource
«AbstractComponent» ContextSourceRegistry
«AbstractComponent» ContextBroker
Straightforward supplied context info; wrapped
Interpreted/derived context: 'available' as new context source
«AbstractComponent» ContextWrapper
«AbstractComponent» Inference_Engine
+interpretKnowledgeBase()
«AbstractComponent» Context-Based Reasoner
+InferContextInfo()
+DiscoverInferenceRule()
«Interface» ReasonerInterface
+DefineRule()
+DeleteRule()
+DiscoverRule()
«AbstractComponent» ContextAggregator
«AbstractComponent» ContextPredictor
«AbstractComponent» AmbiguityResolver
«AbstractComponent» ContextServiceAdapter
Inference engines interpret the appropriate knowledge bases (heuristic or domain specific knowledge)
Reasoner IF allows for specification of rules by eg ASPs or controller functions or other CS

Fig. 1. Functional architecture of our Context Management Framework

Reasoning bears many challenges in itself. In simple words, context reasoning is about deducing entailed contextual information from the various sources of context info, that is linked to real context data. This requires grounding; linking of your models and meta-models used in the inferring process to real context data. For the models, classical engineering methods can be used, but also several complex network modeling methods exist. The challenge is understanding which, why, when and how specific inference mechanisms compliant with multiple network contexts are used by and are valuable to people. In Fig 2 it is depicted how raw network data (e.g. from sensors) can form low-order context, while interpretations and combinations of lower-order contexts can yield higher-order contexts. Typical in low order e.g. Bayesian models used, while in higher order ontology reasoning, extended with rule-based inference can be used. In the project MobiLife (see section 3.4) this is, among others, researched upon.

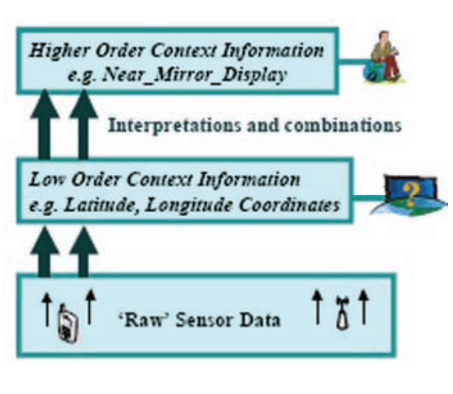

Higher Order Context Information
e.g. Near_Mirror_Display
Interpretations and combinations
Low Order Context Information
e.g. Latitude, Longitude Coordinates
'Raw' Sensor Data

Fig. 2. Reasoning schematics

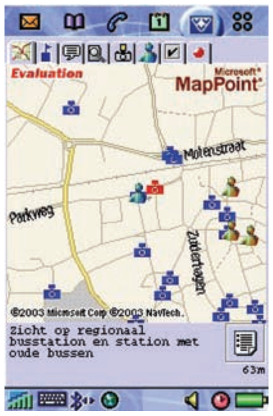

Evaluation
MapPoint
Molenstraat
Parkweg
Zuiderhagen
©2003 Microsoft Corp ©2003 Navtech.
zicht op regionaal busstation en station met oude bussen
63m

Fig. 3. Compass screenshot

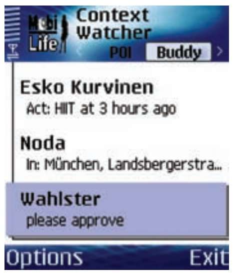


Fig. 4. Context-Watcher screenshot

3.2 Compass

One of the most well-known and well-tried application domains for context-aware applications is tourism. In this domain, we have run an early pilot with the Compass application in 2003. Compass stands for context-aware mobile personal assistant, and is essentially a digital city guide that harvests touristic information from a wide range of on-line information services, based on the current location and interest of an individual tourist.

This application was built upon a context-aware service toolkit, developed in the Freeband WASP project [8]. The toolkit is tailored towards re-use of components and developer convenience and all components are exposed as web services. The screenshot in Fig. 3 shows an example where historic pictures (1880-2003) coming from different sources (monument care, municipal archives, personal collections, ...) are uniformly displayed on a map on the mobile phone, pruned based on the user's location close to the railway station and his interest in architecture. Clicking an icon results in a real-time information request to one of the services. The Compass application runs on Symbian, but advanced features require a Sony Ericsson in the P800/P910 range.

Compass is tested in a small trial in Enschede with tourists, inhabitants and experts walking different routes in the city center in early 2004. Some teams wandered through the city without a specific goal, just like tourists tend to do, other teams were instructed to simulate a young couple with child interested in city architecture or to follow a pre-defined city walk in Compass. In general, the teams were impressed by the new angle of looking at the same city that Compass provided, but at the same time it was judged as too complex for the normal tourist. This complexity was mainly in working with a PDA style phone with a stylus for the first time, the small screen with the different tabs, and unexpected delays because of the GPRS connection, but also the context-aware abilities of the Compass application necessarily cause unpredictable behaviour, which was not always appreciated by the user. However, in general the users liked the surprise effect that the system points them at unexpected places or present them with new information about known places. They also value the aspect that it is at all times possible to request information about nearby places, be it monuments or functional places such as shops or nearby toilets, and also have an easy overview of where their friends are in the city. For local inhabitants, the ability to browse through the different periods in the history of their current location was an eye-opener, which was very much appreciated.

From a design and implementation point of view we found that a) most parties that have information about the points of interest in a city, like tourism offices and national heritage (potential service providers), do not have enough technical experience to expose their information assets via web services. Furthermore their information assets sometimes need heavy post-processing (like enriching data with latitude-longitude information or normalization of old maps) to make them suitable for application in location-based services, b) device independence is very hard to achieve. Although being our initial goal we added about 10% of native code to the generic JAVA application code to work with the phone camera and jog dial, which does not generalize to other devices, unfortunately, c) battery life becomes a limiting factor working with Bluetooth, lit screen, and GPRS continuously, and d) the platform offers functionality that is general enough to support other context-aware applications. For example we also developed a find-a-new-home application based on cell-id positioning techniques using the same platform as well as some other applications.

3.3 Abel

The experiences gained in the Compass trial were the basis for the development of Abel [9]. Abel is a digital guide for tourists who explore Twente (the region around Enschede) by bike, and it is commercially available as of April 2006. Abel basically guides tourists along predefined routes between hotels, and alerts them when points of their interest are nearby. This time we spend most of the development time on a) limiting (and not extending) the functionality for the tourist to what is really needed, b) management functionality for the commercial operator, c) making the application stable and fool proof, and d) practical issues like GPS fast fix, extending battery life time, mounting on the bikes, etcetera. Abel still has to prove itself in practice at the time of writing, but the early tests look promising. Of course, a point of concern stays the delivered content. One of the challenges of Abel is to have at least a basic but high quality content set ready before the tourists arrive in summer 2006.

3.4 Context Watcher

Context-aware life blogging is like writing your personal diaries in an automated fashion. It is no bother at all. A mobile application developed in the MobiLife project [10], named the Context Watcher automatically connects to available sensors, logs the information, detects patterns over time, and generates daily summaries about your location, activities and moods, and environmental conditions [11]. The Context Watcher is written in Python, and runs on Nokia Series 60 mobile phones (see screenshot in Fig 4). The aim of the Context Watcher is to make it easy for end-users to automatically record, store, and use context information. This can be done for personalization purposes, as input parameter for information services, or for sharing information with family, friends, and colleagues, or even just to log them for future use or to provide a mirror for the user to see his own behavior, e.g. how many times did I visit grandmother last year?

The vision behind context-aware life blogging is that man is a social being who likes to communicate about his or her experiences. The Context Watcher facilitates that process by enabling the user to submit pictures from his mobile device that are tagged with information about the context in which the picture was taken, so that title and description can be automatically generated, e.g. “I was on [business trip] together with [Henk] and [Bernd] in [Oulu] and I made this picture of the [Alexanderkatu] while traveling [with public transport] to the [summer school]”, where all the information between brackets is auto-generated. And pictures are easily discoverable by posing a simple context query like “all pictures in Amsterdam together with Bernd when it was snowing”. A second step is that the user can enable the Context Watcher to automatically generate daily summaries of his activities and send that to a (public or private) blog. Such a summary can be configured by the user and might contain an overview of all pictures of that day, the visited places, time spend there, the people you met and for how long, the weather at your location etcetera.

The Context Watcher is a thin client that handles the interaction between locally connected sensors and remote providers of context, such as for location, wellness, experiences, and photos, plus visualization and interaction in a user interface. Context reasoning and enhancement capabilities vary from enhancing e.g. bar code information to book description and personal interest profile, and reasoning to find e.g. someone’s frequently visited places based on historic location tracks and automatic tagging of the places.

The Context Watcher is running since March 2005, and has a user base of more than 150 persons of which about 50 are active each month. These users have formed many relationships, and in their hundreds of thousands context traces about 850 frequently visited places are detected. What the users liked most so far is 1) the possibility to know where the others are and to keep in touch without having to approach them directly, 2) easy access to services because the input parameters are automatically provided contextual parameters (e.g. local weather with one click, rich picture submission with one click, easy public transport info, etcetera), and 3) the sharing of context information across different (mobile) applications and web sites, including Flickr.com and their personal blogs. Sharing information via these channels with nonusers of the Context Watcher (e.g. parents) was perceived as added value.

The Context Watcher application is freely available from [12].

3.5 Smart Homes and Your Context

The Ambient Intelligence vision [13] focusses on embedded, distributed, computational power in combination with sensors and actuators in normal environments like homes, cars, and offices. This integration results in smart environments that are able to adapt their services to its current inhabitants. We are currently studying the applicability of our context management framework [see section 3.1] for these intelligent environments in the Amigo project [14]. We start from the observation that intelligent environments are able to do multiple things: trace the users, and observe the interactions between the users and the services offered by the environment (e.g. watching TV, switching on the light, listening to a particular radio station, making coffee). In an intelligent Amigo home all interactions between the users and the devices are through services, that can be automatically discovered, and have a common security model. Therefore the environment is able to monitor service usage and can derive 'regular patterns' in the activities of persons (e.g. grandma always goes to the bathroom first, then to the kitchen, switches on the radio, and makes some coffee). This knowledge may enable future services, such as switching on the light just before grandma enters the kitchen, or automated selection of her favorite radio channel.

In Amigo our approach is to combine intelligent environments with personal mobility [15]. For that, we defined the concept of Personal Amigo Device (PAD), see Fig. 5. This device is able to operate as a trust-broker between your home and your current Amigo-enabled location. This PAD is seen as a guest-device in the current environment, and uses (typically) a cellular connection to its home. This scenario makes it possible for the PAD to provide detailed information (e.g. temperature, location, nearby users) towards its home environment (privacy rules permitting). Furthermore, it not only conveys contextual information, but also information about the available services. This enables one to directly use devices and services between the two environments without involving the PAD (that is only involved in the creation of the association and configuring the firewalls of both homes). In this scenario, the current context of the user includes the services of its actual environment; in other words, it closely interacts with the service-discovery mechanisms of both homes. This PAD concept enables services based on buddy lists (e.g. gaming, watching TV together) to make use of devices in the environment, instead of only personal devices. So you are able to take (virtually) part in a game with family members when in an Amigo enabled hotel-room, using the hotel TV as your interface.

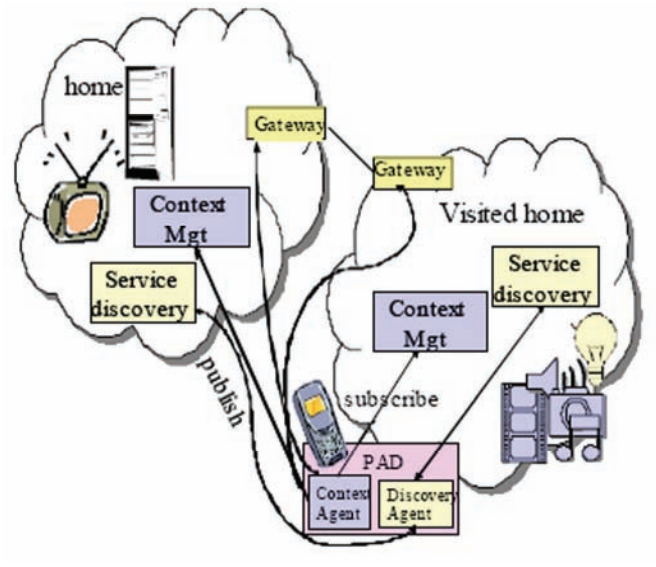


Fig. 5. Personal Amigo Device scenario

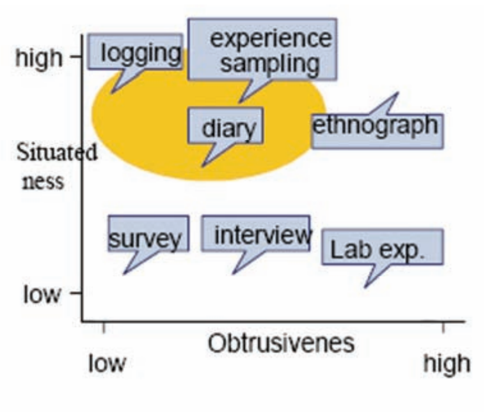


Fig. 6. SocioXensor compared to other methods [18]

3.6 SocioXensor

Current research into context aware applications stresses the relevance of using context information in applications in order to improve desirable properties such as social translucence (see e.g. [16,17]). Despite occasional design successes such as Presence and Instant Messaging applications, researchers are still lacking a systematic understanding which context information is relevant in what kind of situations and which kind of applications. At the same time, designers of context-aware applications face design issues like: selecting which context information should be conveyed or aggregated to other human users (who then interpret that information), and selecting which context information is predictive enough such that it can be interpreted by applications. Although many methods exist to study social phenomena, including interviews, focus groups, surveys, laboratory experiments, ethnography, diary studies, logging and experience sampling, obtaining the right answers to design context aware applications proves to be rather complicated [18]. With our SocioXensor approach we intend to complement existing data collecting methods, compare Fig. 6. SocioXensor is a research instrument for field trials in experience and application research. It aims to strengthen logging and experience sampling by combining them with contemporary mobile and wearable devices such as smartphones and PDAs. Such devices are personal in nature, stay and travel with one person most of the time and consequently enter various contexts of that person (e.g., home, work, and mobile context).

SocioXensor is a toolkit that makes use of the hardware and software functionalities built into mobile devices for which we plan to use our context management framework. The goal is to collect information about how users experience a setting or applications. This includes aspects of functionality and usability, and emotional aspects. SocioXensor collects: objective information of human behaviour within a specific context, such as where and how communication takes place; usage information of applications, like duration of use and keystrokes usage; and subjective data, reflecting the mood of the user, like being stressed, happy, sad.

SocioXensor allows scientists to gain a much deeper and dynamic insight into the relations between user experiences, human behaviour, context, and application usage. It's results can be applied both for formative and summative evaluation, which is further elaborated on in [19].

4 Future Work and Conclusion

Context awareness can add value to applications as perceived by end-users. We consider the following key aspects of context aware pragmatic systems: awareness of the contextual parameters that are relevant for the end-user; providing the information space for modelling, storing and managing the relevant contextual parameters; exchanging the relevant contextual information across and between (heterogeneous) domains; pro-active responding and adapting of the ubiquitous environment (including the system behaviour, applications, but also sensors in the environment) on the dynamic changing context; and predicting the changes in the (mobile) end-user environment, and anticipating end-user's intentions and goals.

Lessons learned from our context aware applications include that end-users unfamiliarity with mobile terminals and mobile applications indicate that user appreciation of plausible and usable context aware support is something for the (far) future. Buddy awareness, indirect staying in touch and easy service access are appreciated by users. We continue our research and prototype validating and will use upcoming SocioXensor – and other – results for feeding our truly pro-active user support systems. Rightful understanding the end-user with his - partly hidden, likely not formulated explicitly, let alone in a machine/system readable format – intentions and goals input is likely to be a crucial factor. SocioXensor's combination of quantitative data tracked by the system (e.g., location, proximity, communication) with qualitative data provided by the users (e.g., availability, feeling, experience sampling) is expected to contribute to the acquisition of a deeper and more holistic insight of people's context, answering questions like: when designing context aware systems we make inferences on users' needs, based on contextual data implicitly sensed by the system: on which criteria should we draw a meaning out of these data? And how should the system adapt to that? Given that quantitative data are more exact, to which extent are they suitable for the understanding of the emotional aspects of experiences? Given the subjectivity of experience, what are the parameters to be assessed for different users?

Acknowledgments.
This work is partly done under the Freeband projects WASP (http://wasp.freeband.nl), AWARENESS (http://awareness.freeband.nl) and FRUX (http://frux.freeband.nl), sponsored by the Dutch government and the FP6 projects MobiLife (IST-511607), Amigo (IST-004182), and SPICE (IST-027617), sponsored by the EU.

References

1. Dey, A.K.D., Abowd, G.D.: Towards a better understanding of context and context awareness. Workshop on The What, Who, Where, When, and How of Context Awareness, affiliated with the 2000 ACM Conference on Human Factors in Computer Systems (CHI 2000), The Hague, Netherlands (April 2000)
2. Eertink, H., van Kranenburg, H., Hesselman, C.: Context-aware content distribution and adaptation, Proceedings WPMC '03, Masao Nakagawa (eds) ISSN 1347-6890 (2003)
3. van Kranenburg, H., Salden, A., Eertink, H., van Eijk, R., de Heer, J.: Ubiquitous attentiveness: enabling context-aware mobile applications and services, In: Lecture Notes in Computer Science, LNCS 2875, Ambient Intelligence, ISSN 0302-9743, Springer-Verlag, Berlin Heidelberg New York (2003) 76-87
4. van Kranenburg, H., Snoeck, N., Mulder, I.: Context aware support targeting plausible and usable results - from reactive, via pro-active to pragmatic systems, WWRF#16, Shanghai (2006)
5. Biemans, M.C.M., van Kranenburg, H., Lankhorst, M.M.: User evaluations to guide the design of an extended personal service environment for mobile services, Fifth International Symposium on Wearable Computers 2001, Zurich, Switzerland (2001)
6. http://www.ist-spice.org
7. van Kranenburg, H., Eertink, H.: Processing heterogeneous context sources, Proc. SAINT 2005, Next Generation IP-based Service Platforms for Future

Mobile Systems workshop (Trento, Italy), ISBN 0-7695-2263-7 (IEEE) (2005) 140-143

8. Koolwaaij, J.W., Strating, P.: Service Frameworks for Mobile Context-aware Applications, eChallenges workshop Future Workplaces: Supporting Mobile User and Worker (2003)
9. http://www.uitmetabel.nl
10. http://www.ist-mobilife.org/
11. Koolwaaij, J.W., et al: ContextWatcher — Sharing context information in everyday life, Int. Conference on Web Technologies Applications and Services, Calgary (2006)
12. http://www.lab.telin.nl/~koolwaaij/showcase/crf/cw.html
13. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J-C.;Advisory Group to the European Community's Information Society Technology Programme (ISTAG): Scenarios for Ambient Intelligence in 2010 (2001)
14. http://www.amigo-project.org/
15. Ramparany, F., Euzenat, J., Broens, T., Pierson, J., Bottaro, A., Poortinga, R.: Context Management and Semantic Modelling for Ambient Intelligence, 1st International EASST-EU Workshop on Future Research Challenges for Software and Services, associated to ETAPS'06, Vienna (2006)
16. Erickson, T.; Kellogg, W.A.: Social translucence: an approach to designing systems that support social processes. Trans. on Computer-Human Interaction (TOCHI) 7 (2000) 59-83
17. Schilit, B.N.; Hilbert, D.M.; Trevor, J.: Context-aware communication, In IEEE Wireless Communications, 9(5) (2002) 46-54
18. Mulder, I., Steen, M., Mulder, I., Steen, M., ter Hofte, G.H., Kort, J.: Mixed emotions, mixed methods -- An investigation of how to study experience of we-centric context-aware adaptive mobile services [FRUX Deliverable 1.5] Enschede, Netherlands (2004)
19. Mulder, I., ter Hofte, G.H., Kort, J.: SocioXensor: Measuring user behavior and user eXperience in conteXt with mobile devices. In Noldus, L.P.J.J. et al. (eds.). Proceedings of Measuring Behavior 2005, Wageningen, Netherlands (2005) 355-358

Context-Awareness in Mobile Games

Jari Porras[1], Kari Heikkinen[1], Kimmo Koskinen[1] and Riku Suomela[2]

[1]Lappeenranta University of Technology, P.O.Box 20, 53851 Lappeenranta, Finland
[2]Nokia Research Center, P.O. Box 100, 33721 Tampere, Finland
{Jari.Porras, Kari.Heikkinen}@lut.fi, Kimmo.m.koskinen@iki.fi, Riku.Suomela@nokia.com

Abstract. Context awareness is one of the trends of current application development. Ever since location based applications got into the market the context awareness has attracted developers as well as users with the promise of intelligent user centric possibilities. In this paper a platform for rapid context aware application development is used for the creation of context aware games. The platform itself offers several sources of context data through context producers. Three games created on top of this platform are presented and evaluated from the context awareness point of view. Some details of heuristics developed for evaluating mobile games and context awareness are presented

Keywords: MUPE platform, context awareness, mobile games

1 Introduction

The importance of gathering and disseminating context data for various applications is widely recognized. The use of context information evolved with the use of location information in different applications. Location-aware applications enable services like navigation and service positioning. However, location awareness is only a small part of wider topic, namely context awareness. Dey [2] has given a widely referenced definition for the context: "Context is 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 the application themselves". Context information can be used for several different purposes as the classification of ePerSpace [3] shows us. ePerSpace has categorized context information into the following groups:

- Environmental context (e.g. temperature)
- Personal context (e.g. heart beat)
- Task context (e.g. events)
- Social context (e.g. social network)
- Spatio-temporal context (e.g. location, time)
- Device context (e.g. battery level)

- Service context (e.g. service specific data)
- Access context (e.g. network capabilities)

These categories give a nice overview of the possibilities that the context awareness enables. These categories include both user (client end) contexts (environment, personal, social, and device) as well as remote (server end) contexts (environment, task, service and access). Some of the contexts are very personal (personal, social) whereas some are general and very neutral contexts (environment, task, spatio-temporal, service, access). This classification is just a way to emphasize different flavors and possibilities of the use of context awareness. By knowing that the context information is available the main issue here is the meaningful use of this information. So far mainly location based services are considered meaningful for the users. This is easy to believe as location is self-evident for users as well as for the developers.

Development of context aware applications is a tedious task without a proper architecture and tools. At the moment several tools are available. Context Toolkit [8] allows developers to focus on the application. ARToolkit [1] uses augmented reality for enabling context aware applications. The platform for context aware application development used in this paper is MUPE (Multi User Publishing System) [7] [9]. MUPE platform is an open-source effort and it allows the use of different context sources for the application developers. MUPE platform can be used for developing different types of context aware applications. In our studies the emphasis has been with the context aware games. Context aware games utilize context information with mixing the gameplay with real world information.

2 Context-aware Application Platform: MUPE

MUPE (Multi User Publishing Environment) is a client-server application platform for creating multi-user context aware applications for mobile phones. It strongly favors context usage in mobile applications. MUPE is a Java based implementation designed to support as many standard devices as possible, both in the client and server side. The overview of the MUPE application platform is presented in Figure 1.

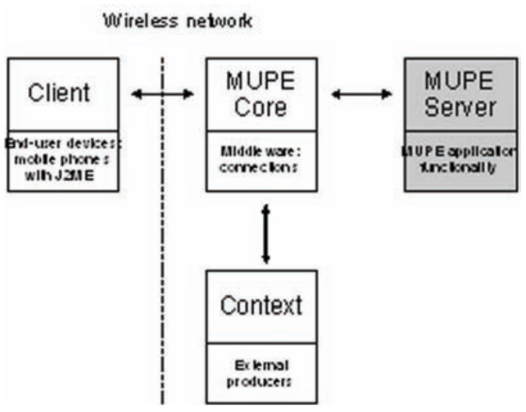


Fig. 1. MUPE application platform overview

MUPE application development is aimed to be rapid which is achieved by using several techniques including code and script reuse, and programming the application server only. Developing applications for mobile phones can be a cumbersome task, since the development is done in PC, which is a different device from the target platform. MUPE uses a scripted client that allows all the development to be focused on the server, which offers the biggest advance in development speed, since no mobile programming is needed. The server uploads the UI and functionality to the clients allowing rapid development of applications.

Context support is built into MUPE in many ways, and the context information can originate either from the connected user clients (client side contexts), or any source in the Internet (server side contexts). Since MUPE works in different clients, all of the clients do not have the same context-aware technologies in their use. Some clients may have GPS, whereas others may have motion sensors. Context information originating from the Internet, however, is the same for all devices. This information is not personal at all, though.

Adding context-awareness into applications is made easy in MUPE. Any client may use a new context by adding this functionality to the UI scripts. All non-standard functionality in MUPE client is written as plugins, and if a particular client does not have the functionality, the plugin loading fails. Context-awareness on the server side is as easy, but done differently. External context information must originate from a MUPE context producer, which feeds information to MUPE. First, the information is sent to a context manager that filters incoming information and forwards it to the MUPE server. Inside MUPE server, there is a corresponding context manager that handles all context information directly in Java.

MUPE is being actively developed constantly in many fronts, since the platform is open source. The platform development is focusing on many areas in MoMUPE project. The platform is being developed to accommodate the new technologies in mobile phones, and to expand the client scripts to better support development. Development tools for the Eclipse IDE are being developed, and they focus on supporting the special features of the MUPE classes. New context-aware technologies and respective tools are also in development. MoMUPE project aims at making MUPE an open-source platform for developing context-aware applications. So far, the code camps and other events have proved that the platform has a lot of potential, and the biggest issues should be dealt with the developer tools.

3 Context Information Mediation in MUPE

For every context data source a context producer is implemented to support the context data provisioning in MUPE. Figure 2 shows the data provisioning path starting from the originating context information source and ending into the MUPE application. The basic idea is that a Context Producer is used to gather and wrap the original context data into CEP (Context Exchange Protocol) [6] format. The MUPE servers can then use the CEP data in a universal way. It is enough to write a single context producer to serve all MUPE applications. Each context producer feeds information to all MUPE applications that subscribe for the data.

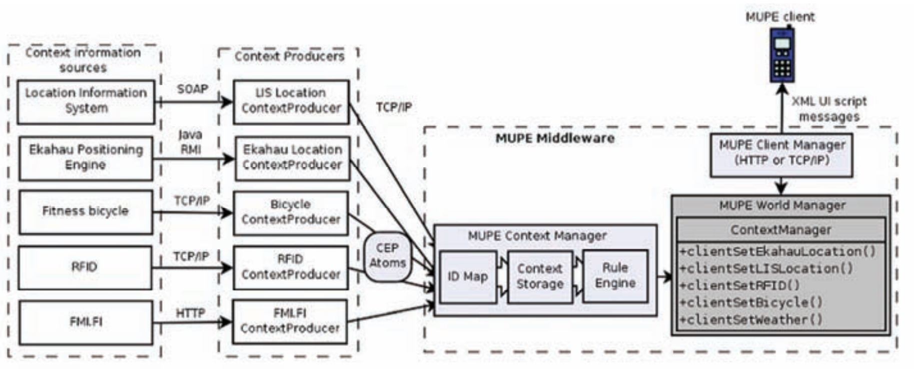


Fig. 2. Context information in MUPE

The following context producers are already implemented to the MUPE platform:

- Location information system (LIS) [5] (server, spatio-temporal)
  LIS gathers and provides cell-based location information from a WLAN network. An interface for making queries and tracking the location of WLAN devices or their users is provided. The context producer first polls for changes in the location for a defined set of devices and then wrap the LIS location data into a CEP atom which is forwarded to the MUPE servers.
- Ekahau Positioning Engine (EPE) [10] (server, spatio-temporal)
  EPE provides map/x/y and area based location information by using signal 'fingerprinting'. The location of a device is calculated by using a

probabilistic method to compare current signal measurements to a calibrated signal fingerprint of an area. Context producer allows a push -like continuous receiving of location estimates for devices.

- Global Positioning System (GPS) (client, spatio-temporal)
  GPS provides x/y/z coordinates for the client connected to this context producer. GPS is an interesting case for MUPE, since each MUPE client can use a GPS receiver directly, or the GPS can act as a context producer feeding information directly to the context manager. When the client connects to the GPS, the functionality can be uploaded to the phone with a single script, but the GPS information is fed to MUPE only when the user is actively using the application. If the GPS feeds the information to the context manager, the information flow is not dependant on the user using the application, but additional software is needed.
- RFID/barcode (server/client , spatio-temporal, task, service)
  Passive RFID tags can be used for representing events in the environment. The RFID reader was interfaced with a dedicated program that read tag sightings and relayed them over the context producer program and finally as CEP atoms to MUPE servers. If used only as an ID for some event then barcode could be used as well.
- BluetoothID (client, social, task)
  BluetoothID provides a nice proximity based approach for social networking. It is easy to determine which other players are at close proximity. Bluetooth address space can also be used like barcodes or RFID tags.
- Environment context (server, environment, service)
  Environmental context information is provided from the web service of the Finnish Meterological Institute (FMI, http://www.fmi.fi). The service provides weather measurements from different locations in Finland. The weather data contains information about temperature, humidity, wind speed and direction, air pressure and cloudiness. The data is updated every hour, since weather is not changing too rapidly from a human perspective. The operation of the weather context producer was simple: fetch the web page containing the data with HTTP, parse the information and form a CEP atom which is forwarded to the MUPE context manager. This kind of context information requires the use of location information in order to be meaningful.
- Fitness bicycle (server, personal, task, service)
  Personal context data can be provided through a Tunturi Recumbent. This exercise bike can measure the current speed of the bike and the heart rate of the user. This data is read from the bike using a serial connection with a dedicated program. This is a very interesting case in the context producers since this device is location-static, that is, the device is in a fixed location.

In case users would use this, they would have to tie their application to locations where the bike would be.

- Stock market data (server, environment, social)
  Stock market data is very promising context information as the stock market data contains several different aspects. Stock markets can be divided according to the business area or the location of the stock markets. Use of the daily rates of natural resources enable real context for trade or production type of games.

4 Evaluating Mobile Context Aware Games

Systematic development of mobile context aware games requires ways to analyze the contents of a game. In our current project MoMUPE we use mobile game heuristics developed in Nokia in addition with the new heuristics that are in development for context-awareness. The heuristics designed for evaluating context awareness is built from several viewpoints. Firstly, it follows Mobile Game Playability Heuristics [4] and secondly adds some modules to it (e.g. public screen heuristics). The heuristics can be used to evaluate any application that uses context in some integral manner in its application logic. The heuristics could be used both as a tool for expert analysis and as a design guideline for application designer.

This paper introduces a sample heuristic in each class. Currently context-awareness class has fifteen (15) heuristics, divided into application oriented context awareness and public screen issues. These heuristics are in development, so they will most likely still evolve. The other classes follow the heuristics presented in [4]. The evaluation of [4] is changed so that to the grading system (major, medium, minimal) another level was added (none).

Table 1. Sample heuristics for the game evaluation

Heuristic for Evaluation	Description
GU4: Indicators are available [4]	The user sees the information that is needed to use the application. Also frequently used and application critical information should be available. The device indicators, e.g. battery level, should be visible
GP4: User is in control [4]	The application should provide at least an illusion that the user is in control of what happens in the application
MP1: The application supports communication [4]	If there is a need in application logic to communicate with other users, the

	application should contain a direct or indirect interfaces or communication channels
CA1: Adapt the interface of the application to the context of the user	The UI of the application changes if the context of the user changes so that application logic will be changed or needs input from the user.
CA15: Provide the possible audience with adequate information of the application events	The public screen available in some location (e.g. railway station) presents application related information to a known or unknown audience at the location.

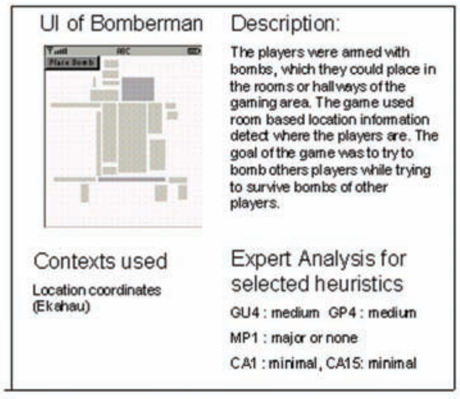
5 Implementations of Context-Aware Games on MUPE

MUPE platform has been used in several occasions for the development of mobile context-aware games. Lappeenranta University of Technology has arranged two code camps as well as participated in two separate projects where MUPE has been the main platform and context aware applications as the main emphasis.

Code camp is a collaborative learning effort where participants learn together. The „Location Aware MUPE games“ code camp was organized as a part of the 2nd Workshop on Applications of Wireless Communications in 2003. This code camp lasted 24 hours and 6 games were implemented by groups of 2-3 participants with no prior knowledge of MUPE platform. The Bomberman game was implemented during this event. Another week long „Context Aware Games“ code camp was arranged as an optional course for the students of Lappeenranta University of Technology. 90 students participated in this intensive course. As this course had both game patterns theory and implementation parts, only half of the students participated to the actual coding part. As a result 14 context aware games were implemented this time. The snow war game was implemented in this event. At the moment we are working on the ongoing MoMUPE project that enhances the MUPE platform by providing new context sources, tools as well as new game designs and implementations. Collect Me game is designed in this project.

5.1 Bomberman

This game is a very good example of a game using location information. The game requires user to move around the game area, and location information is used as a very direct input in the game. The user sees most of the information (albeit limited in text) needed to play the game and thus only minimal changes to the UI would be required (such as e.g. UI does not have indicators to support avoiding the bombs). Also, it is not clear for beginner players what is the grey area (own bomb and own location or own bombs). The user definitely is in control as he/she needs to place the bomb. However, e.g. for tactical reasons it should be possible to defuse the bomb. So the user does not really control the game, the bombs control the game. These aspects are, however, minor tweaks to the game. The UI does not have any communication channel visible, and for this reason if the communication is supported, the UI needs to be redesigned. The UI does not provide enough data to support context-awareness and so further UIs (prior and after location change) should be available for evaluation. The UI is informative as it clearly shows a map and so only some clarifying text (for the audience) in needed to support showing this game in a public screen.


UI of Bomberman

Description:

The players were armed with bombs, which they could place in the rooms or hallways of the gaming area. The game used room based location information detect where the players are. The goal of the game was to try to bomb others players while trying to survive bombs of other players.

Contexts used

Location coordinates (Ekahau)

Expert Analysis for selected heuristics

GU4 : medium GP4 : medium

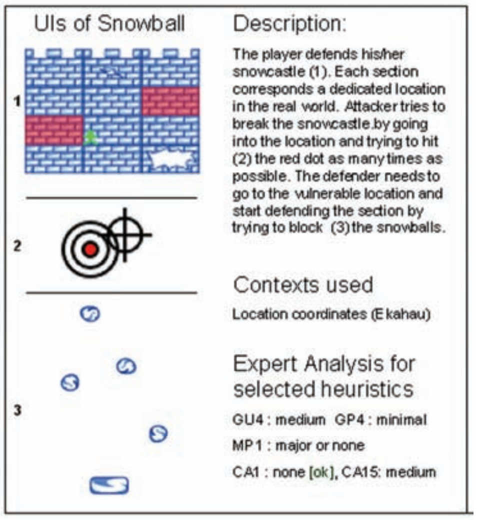
MP1 : major or none

CA1 : minimal, CA15: minimal

Fig. 3. Context aware Bomberman game

5.2 Snow War

The Snow war game extended the use of location information with minigames. Location information was used to trigger the minigames, instead of using it as a more direct game input. Still, the minigames were small enough so that they required active moving in the real world. The user sees some of the information (very limited in text) needed to play the game and thus some changes to the UI would be required; such as e.g. the game itself does not support the player location in real world (e.g. 3rd floor east wing) and it takes time for players to adapt to the real location unless the location are somehow else described. The mini-games do not tell to the player what to do and what happens if the player misses the target or is there a chance to fix the broken part of the castle. The attacker must feel that he is in control of the game (as initiator), for the defender the game might be time to time very frustrating. For playability reasons the defender should have more options. The game does not have communication channel, and the interfaces need to be redesigned if such is required. The UI shows the location of the players and it nicely adapts to the location of both the attacker and the defender. The UIs needs changes if the game is presented in the public screen. Not only UIs require some clarifying text (for the audience), it would be beneficial to have a map-like area (not snow castle) to show the flow of the game at a public screen.



Description:

The player defends his/her snowcastle (1). Each section corresponds a dedicated location in the real world. Attacker tries to break the snowcastle by going into the location and trying to hit (2) the red dot as many times as possible. The defender needs to go to the vulnerable location and start defending the section by trying to block (3) the snowballs.

Contexts used

Location coordinates (Ekahau)

Expert Analysis for selected heuristics

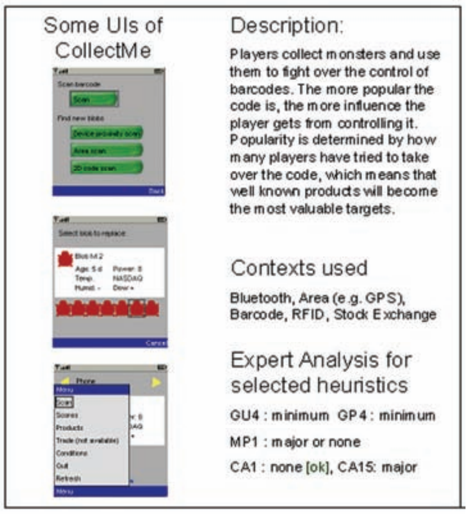
GU4 : medium GP4 : minimal

MP1 : major or none

CA1 : none [ok], CA15: medium

Fig. 4. Context aware Snow war game

5.3 Collect Me



Description:

Players collect monsters and use them to fight over the control of barcodes. The more popular the code is, the more influence the player gets from controlling it. Popularity is determined by how many players have tried to take over the code, which means that well known products will become the most valuable targets.

Contexts used

Bluetooth, Area (e.g. GPS), Barcode, RFID, Stock Exchange

Expert Analysis for selected heuristics

GU4 : minimum GP4 : minimum

MP1 : major or none

CA1 : none [ok], CA15: major

Fig. 5. Context aware Collect Me game

The user almost all the information that is needed to play the game and thus only minimal if any changes to the UI would be required. The graphical items are mostly supported by clarifying text. The player definitely has a feeling of control as nothing happens unless he/she has triggered some action in the game. The game currently does not have a communication channel and the

UIs need to redesigned or chat need to be added to the menu. The UI changes mostly by the actions carried out by the user and context is rather tool for the user. The user could use Bluetooth to scan the neighborhood, GPS to scan the selected area and physical object tragger to analyze the bar codes of the products. Thus the context is handled a little bit differently as in the two other games. Nevertheless, the context information supports the game logic. If the game is portrayed in a public screen, the UIs need to be redesigned very much. Even though the text clarifies nicely the actions, the flow of the game is quite a slow and the game server requires a lot of players so that there would happened something in the public screen. For example, the battle of two players competing of one particular item could be presented as an advertisement in the public screen.

6 Evaluation of the Context Usage in Mobile Games

A questionnaire of some 60 questions concerning the development of mobile games as well as context awareness was given to the participants of the week long “Context-Aware Games” code camp. This code camp offered four types of context information, i.e. location, fitness bicycle, weather and RFID, but the location information was by far the most used as 73 % of the groups used location information as their context source. As the offered location information was indoor -type of information (floor, room, x/y), this type of context information was, in a sense, "closest" for the participants (which spent their time mostly indoors, coding away...). 18 % of the groups used bicycle data (speed and heart rate) and 9% used weather data. The state of the outdoor weather didn't have as rapidly changing characteristics (we only received one update per hour) as the other contexts. The fitness bicycle could have been a very interesting context because of its rapid-changing and session based characteristics. Still it was used by only few groups.

To us, the most interesting characteristic in the game implementations would have been the use of multiple contexts in one application. Only one group used the bicycle and location contexts together, but most of the groups focused on one context source. Based on this, the usage of multiple contexts in one application is a demanding task. Also, one context source can provide multiple sub-types of context information (e.g. temperature, speed and direction of wind for environmental context; heart-rate and physical stress for physiological context; different precision for location context), which broadens the use of the actual context information.

7 Conclusions

In this paper we have presented the usage of context aware information in mobile games. With the presented MUPE platform it is easy to produce meaningful games that use different types of context information. Since anyone can create a mobile multiplayer context-aware game, there should be ways for the developers to pay attention to the playability of the games. The developers are in many cases students, or small groups who do not have the skills necessary for such evaluations, so the game playability heuristics provide help in this. Context-awareness, a key feature of future games, add another layer of complexity, that further poses new challenges for the playability. Once finished, the heuristics for context-aware games will provide the developers even further help in the development. While developing mobile and even possibly context aware games, the developer should remember the following rules:

1. Game is on all the time. It is good, that the game is running even though you are not actively playing.
2. Short and Long play sessions. It should be possible to have both short, and long duration play sessions.
3. Adapt to player schedule. Allow players to play the game when they have the time, not when the game requires it.
4. Hide the technology. Do not talk about Bluetooth - talk about social proximity, proximity, or other. It is more important to discuss what the software does, not how or what does it.

References

1. AR Toolkit, http://www.hitl.washington.edu/artoolkit/, 2005.
2. Dey, A., K.: Providing Architectural Support for Building Context-Aware Applications. PhD thesis, College of Computing, Georgia Institute of Technology, 2000
3. EU-project ePerSpace, http://www.ist-eperspace.org/, accessed last time 4.5.2006
4. Koivisto, E., Korhonen, H.: Mobile Game Playability Heuristics. Forum Nokia, http://www.forum.nokia.com/info/sw.nokia.com/id/5ed5c7a3-73f3-48ab-8e1e-631286fd26bf/Mobile_Game_Playability_Heuristics_v1_0_en.pdf.html
5. Koskinen, K., Spacil, R., Ikonen, J.: Location Information System – a description of positioning middleware system, Proceedings of ECEC Euromedia 2005 international conference, (2005)
6. Lakkala, H.: Context exchange protocol specification. Technical report, Nokia, 2003
7. MUPE platform site, http://www.mupe.net

8. Salber, D., Dey, A., K., Abowd, G., D.: The context toolkit: aiding the development of context-enabled applications. Proceedings of the SIGCHI conference on Human factors in computing systems, (1999), 434–441
9. Suomela, R., Räsänen, E., Koivisto, A., Mattila, J.: Open-Source Game Development with the Multi-User Publishing Environment (MUPE) Application Platform. Proceedings of the Third International Conference on Entertainment Computing 2004, 308-320, Lecture Notes in Computer Science 3166 Springer 2004
10. Suomela, R., Koskinen, K., Heikkinen, K.: Rapid Prototyping of Location-Based Games with the Multi-User publishing Environment Application Platform. Proceedings of the IEE International Workshop on Intelligent Environments, (2005), 143-151

CONTRIBUTED PAPERS

Goal-Based Requirements Modelling as a Basis for Adaptivity to the Service Context

Klaus Schmid

University of Hildesheim, Institute for Computer Science
Marienburger Platz 22, 31141 Hildesheim, Germany
schmid@sse.uni-hildesheim.de

Abstract. The demand for flexible software systems is constantly increasing. A promising approach is to enable systems to become self-adaptive up to a point where they are able to compose themselves flexibly based on available services. Despite this flexibility, behavior consistent with the underlying requirements of the system is still expected. This is a challenge to current software engineering approaches, which are built on the vision of a stable system. In this paper, we propose a specific approach to software engineering and in particular requirements engineering to support the systematic development of adaptive systems. We focus in particular on adaptivity to the service environment.

1 Motivation

In recent years software systems have become ubiquitous in our environment. This development provided the basis for a paradigm-shift, described by terms like Ubiquitous Computing [Wei91], Pervasive Computing [Sat01], or Ambient Intelligence [Aarts04]. These paradigms have in common:

- Future systems are expected to consist of a number of individual devices that participate in providing services to the end-user. Thus, end-user services are generated through runtime provisioning. No single development organization is able to individually define and provide the complete service.
- The service provision will be increasingly adaptive and proactive, i.e., systems will react to changes of the operational context and pursue the user's goal on his behalf.

In this paper, we will focus on the implications of this category of applications on software engineering approaches. In particular, we will focus on a requirements and knowledge modelling point of view, based on our assumption that for adaptive systems in general – and for context-aware applications in particular – those two must be strongly intertwined [SEG05]. The approach we propose is driven from the idea that a system that aims to be proactive and context-aware must be able to pursue user goals despite all (self-)adaptations. This is derived from the concept that user and system goals provide the limits for adequate self-adaptation of the system.

Any adaptation that violates a goal is counter-productive and should be thus avoided. If an adaptation that could keep intact the user goals is not performed, this is counter-productive as well. Thus, it becomes the role of the application (agent) to evaluate the context – in particular the available services – and use this to pursue the user's goals on his behalf. The system context that is defined by the available services is also called the service context.

We will also discuss how such a goal-based modelling approach can be used as a basis for deriving both: static requirements as well as descriptions of context and background information. The approach complements existing work on technical aspects of context-aware systems like modelling of context, processing of context, and existing methodological work on later stages (e.g., architectures) [Prz05, ST05, ML+05a].

The paper is structured as follows. In the next section, we will discuss some related work. In Section 3, we will discuss our vision of a development approach for these kinds of systems that relies on a three-lifecycle model. In Section 4, we will discuss our approach for the integration of user demands and system proactivity. Finally, in Section 5 we will conclude.

2 Related Work

The approach we present here relies on existing work in the requirements engineering community for goal-oriented requirements engineering (GORE) like the ones by Lamsweerde, Yu, Anton, Rolland and others [Lam03, RSA98, YM98, Ant96]. Here, we imply a hierarchical goal decomposition, similar to [Lam03]. The major difference is that we do not expect a full decomposition, but rather some goals may be left unrefined in order to satisfy them dynamically during runtime.

The basic approach of preparing infrastructures and exploiting their flexibility can be regarded as similar to approaches like product line engineering (PLE) [CN01, BF+99, PBL05]. While in PLE an infrastructure is prepared for reuse and its flexibility is exploited during development time, we focus here on the exploitation of the flexibility during runtime. In addition, we regard the development of the underlying services and the final applications as being done by independent organizations. This leads to a shift from the two-lifecycle model common in product line engineering [PBL05] to a three-lifecycle model for adaptive systems as described here.

Self-adapting and context-aware systems have received rapidly increasing interest over the last few years. This is true both for systems that adapt their behavior to external circumstances like [PS+04] as well as for systems that monitor their own performance and adapt their behavior in order to keep up their current performance level [DEAS05]. We see context-awareness as a key capability for successful adaptivity [CAPS05].

The set of available services is often regarded either as fixed or as varying within easily describable boundaries. Most work that focuses on environments where the available services change dynamically is focused on the context of web services and in particular on semantic web services. Among those efforts are the OWL-S effort [DAML06] and the Adaptive Services Grid project (ASG) [ASG06].

Finally, our work relies on existing work in knowledge engineering. While usually knowledge engineering is regarded as an independent effort from requirements engineering, we tightly integrate the two. In this paper, we do not rely on a specific language for knowledge representation. Typical languages for ontologies are for example RDF [W306], DAML+OIL [DAML06], F-Logic, OWL [DAML06] or WSMO [WSMO06]. The examples we present use a pseudo-language for simplicity. This language is closest to OWL.

3 Development Structure

While current work on context-aware systems is often performed by a single organization which implements most or all system parts, starting from simple core services like information storage to the final user interface, we believe this approach will not scale, but will be replaced by a multi-group approach where applications, infrastructure and services will be provided by different organizations. This has ramifications for the system architecture and the development process.

3.1 System Structure

We can distinguish three major concerns in the development of adaptive systems. As a result, we assume them to provide the basis for a corresponding system architecture. These layers are shown in Figure 1b.

Infrastructure – the infrastructure layer provides the glue among the applications and individual base services. For a specific environment the infrastructure is probably the most stable part, if compared to the other two layers. Besides base functionality like capabilities for matching services and applications, the infrastructure also contains the domain model. This domain model provides a common ontology for services and applications.

Applications – the applications run on top of the infrastructure layer. They enable the final user to achieve his or her ultimate goals. In our model applications consist of both: fully implemented functionality (e.g., like classical desktop applications) and only partially implemented functionality were open goals remain which are satisfied at runtime with the help of services. In the dynamic integration of services the systems rely on appropriate services.

Services – finally services provide the underlying execution capability for functionality. This may range from complex services like ordering theater tickets to basic services like sensor readings. In our model, meta-information that the services provide about themselves forms the basis to enable the applications to determine at run-time which service to employ.

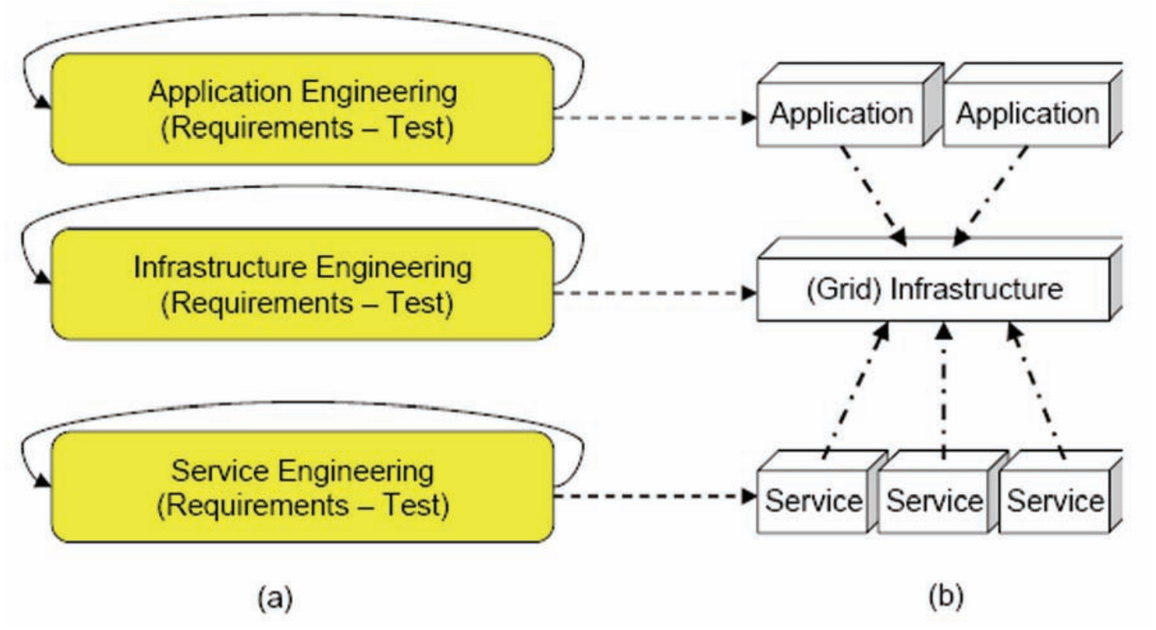


Fig. 1. (a) Three-lifecycle-model (b) Three-layer-architecture for flexible systems

3.2 Development Approach

The system structure is reflected in the development model as well. For this reason, we also speak about a three-lifecycle model. As the development and organizational context of each layer can be completely different, the development process can be different as well. Due to differences in terms of the stability of the different parts, the relevant project sizes and so forth, system development for the different layers should actually follow different development approaches. Thus, we distinguish:

- *Application engineering* aims at developing the final, proactive applications. This is done by mixing goal models that are interpreted at run-time with goal models that are fully encoded at development time.

- *Infrastructure engineering* aims at setting up an integration basis that also provides the underlying reasoning capabilities to enable context-sensitive and proactive behavior.
- *Service engineering* aims at developing individual services that can be dynamically or statically used by the applications in order to provide their underlying behavior.

In the following section, we will focus in particular on the requirements engineering aspects of these engineering activities and will discuss how this leads to a tight integration of requirements engineering and knowledge engineering activities.

The resulting approach integrates the derivation of the core functionality with the derivation of the necessary ontological and context information.

4 User Demands for Adaptive Systems

In this paper, we build on top of an existing approach, which we developed initially in order to address system adaptivity – in particular in the context of adaptive service grids [SEG05]. This approach relies on the notion of goal-based requirements engineering as proposed by Lamsweerde [Lam03].

We will briefly summarize the general approach to goal-oriented requirements engineering as defined by Lamsweerde in Section 4.1. In Section 4.2, we will describe our approach to the general case of adaptive systems. This is then extended in section 4.3 in order to address specific aspects of context-awareness.

4.1 Goal-Oriented Requirements Engineering

Goal-Oriented Requirements Engineering (GORE) in general and in particular the approach by Lamsweerde [Lam03] rely on the notion of system goals. Any system, which is built, is built in order to serve a certain purpose. This is the top-level goal, which can be further refined into lower-level goals. For instance, a system might have the top-level goal “provide comfort” (e.g., in an environment like a living room). This goal can be decomposed into sub-goals like: “identify situation”, “adjust temperature”, and “establish atmosphere”, where the last can be further sub-divided into goals “establish lighting”, “establish music”, etc. This is shown in Figure 2.

This model can be further refined into sub-goals. For example we can decompose the `establish music` goal into `determine music` and `play music`. This is a milestone decomposition [Lam04]. This decomposition effectively describes a sequence.

A goal-model can be systematically refined until all relevant objects and their actions are revealed. This leads to a level of description, which provides the basis for implementing the final system. At this point all actions of the system and the responses to events in the environment are identified. As a consequence, this kind of approach to system development restricts context-aware adaptivity of the system to a minimum.

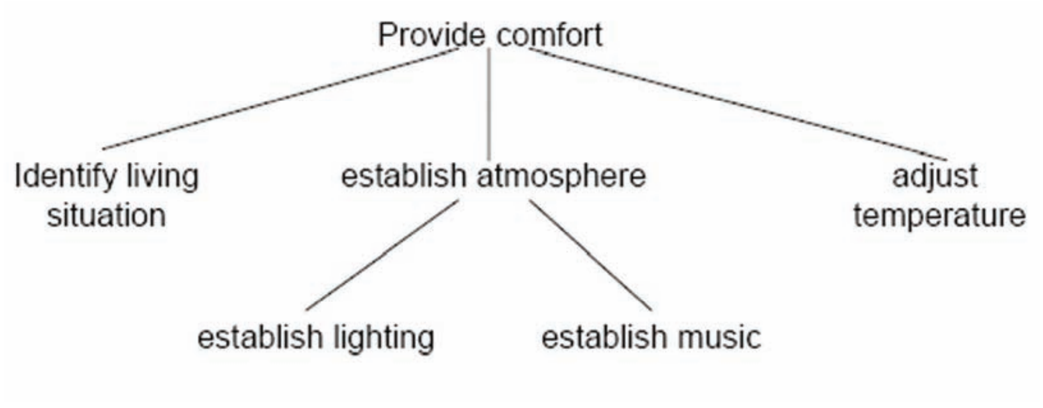


Fig. 2. Goal Decomposition for Goal "provide comfort"

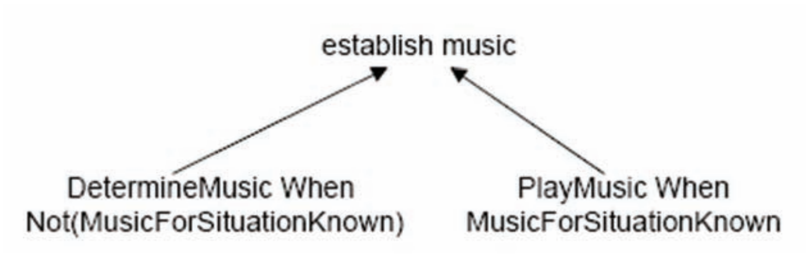

establish music
DetermineMusic When Not(MusicForSituationKnown)
PlayMusic When MusicForSituationKnown

Fig. 3. Goal Decomposition for Goal "establish music"

4.2 A Goal-Based Approach to Support System Adaptation

The goal-based approach described in the previous section aims at providing a guideline for the goal-oriented development of traditional (i.e., non-adaptive) systems. The derivation of the final system functionality is done completely manually. However, as we pointed out in Section 1, goals can provide a good basis for supporting adaptivity. The prerequisite is that the goals are defined to a level of detail that enables the system to dynamically identify services that can be used to achieve a goal. The approach is described in more detail in [SEG05]. This is shown in Figure 4.

It is important to note that here, based on an engineering process the decision is made explicit where adaptation should occur. This ensures a controlled and reliable form of adaptivity of the application.

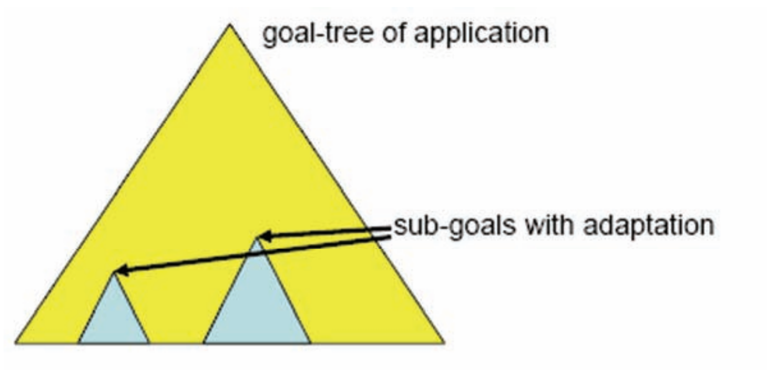


Fig. 4. Embedded Adaptation in a Goal-Tree

Any goals that are not resolved during development time must be resolved at runtime. This is mainly the task of the infrastructure layer. It must match the open goals to available services and identify potential services that are able to satisfy the goal. As a consequence, the infrastructure needs the following kind of information: (in parentheses the information source is given)

- Goal description (application) – a formal definition of the goal and any constraints must be given in order to enable the run-time identification of appropriate services. Similar to the establish music goal a decomposition of the goal can be made into `determine lighting` and `set lighting`. Some form of mode needs to be defined and a spatial context (close to the person) must be defined. In our example the application will probably use forms of mode definition different from the definitions in the infrastructure. Here additional assertions are required to translate the application light modes into the infrastructure definitions.
- Service description (service) – a formal definition of the available services and any constraints on their usage. Again this service might require parameters different from the infrastructure. An accompanying knowledge module is required to provide the translation.
- Domain model (infrastructure) – the domain model provides a common upper ontology. Both application and services are defined in relation to this. The ontology strongly depends on the specific domain for which it is established. For example, in the context of our ambient living scenario, we need to provide base information on rooms, light (or visual) and audio services. This is a prerequisite so that services and applications can be defined on top of it.

The differentiation into an underlying domain model and the goal description and service description on top of this does not only simplify the matching, it also means that the individual service descriptions and goal descriptions can be developed independent of each other. In particular, this isolates the infrastructure from changes to ontology definitions. Here, additional assertions are required to translate the application goals into the infrastructure definitions. The following considerations provide further information on the specific definitions and properties that need to be developed.

The goal model of an application is developed in a people-based process. Thus, we do not require a formal proof of its correctness (although, of course, a proof would increase our acceptance for the final result). Thus, we expect in general for the goal decomposition:

$$\{G_1, \ldots , G_n, \mathrm{Dom}\} \models G \qquad (1)$$

where Dom denotes the characteristics of the domain. This is a property of the software development expect (and can verify if needed).

For matching service and goal a formal proof is required, as this must be performed automatically. As a consequence, if we have a formal representation of G_i and can show that for a service S holds (where spec(S) denotes the formal specification of S) Equation (2), we know S will lead to a working application:

$$\{\mathrm{spec}(S), \mathrm{Dom}\} \vdash G_i \qquad (2)$$

This second property needs to be guaranteed at runtime.

4.3 Proactive and Context-Aware Applications

So far, we discussed the basics of the goal-driven engineering approach. Our main focus so far was on the impact of the *service-context*, i.e., how do we deal with the fact that services will not be available all the time, but the applications shall exploit existing services in an optimal manner.

Especially in an ambient context, the location of services will play an additional role. For example, for someone who is in the kitchen and wants to switch on the lights, a service manipulating the lights in the living room is of no interest. This is taken into account in the conceptual model shown in Figure 5.

Thus, we need to further refine our goal representation with context information. This can be done, for example, with using additional attributes, e.g., for describing a specific location for which a service, respectively a goal, is defined. This approach is illustrated in Example 1.

First we need to define the goal (on the application side) which needs to be achieved. This is given by "set_light" in the example below. In order to further describe the semantics of the goal "set_light", we need to provide additional information. In our example, we introduce the additional concept "Room_Light" and define further constraints on it. The set_room_light function now defines a service for switching on the light in the room. In order to have a semantic grounding for these terms, some basic terminology needs to be available. This must be defined in the infrastructure ontology. In our example we assume that the terms: room_light(.value), and position, as well as person are already available.

```
Goal set_light(person, mode):
Room_Light SubClassOf Light
- Room_Light restriction OnProperty: position =
position( person )
set_room_light(mode) SubClassOf Achieve
- set_room_light resultOf: room_light.value = mode
```

Example 1. Goal formalization

One should note, that in addition we make the implicit assumption that the values used for determining the light level are on the same scale as the values used for setting lights in general. This approach could be extended to take care of such a recoding and could also be used to take into account additional factors besides the position of the light.

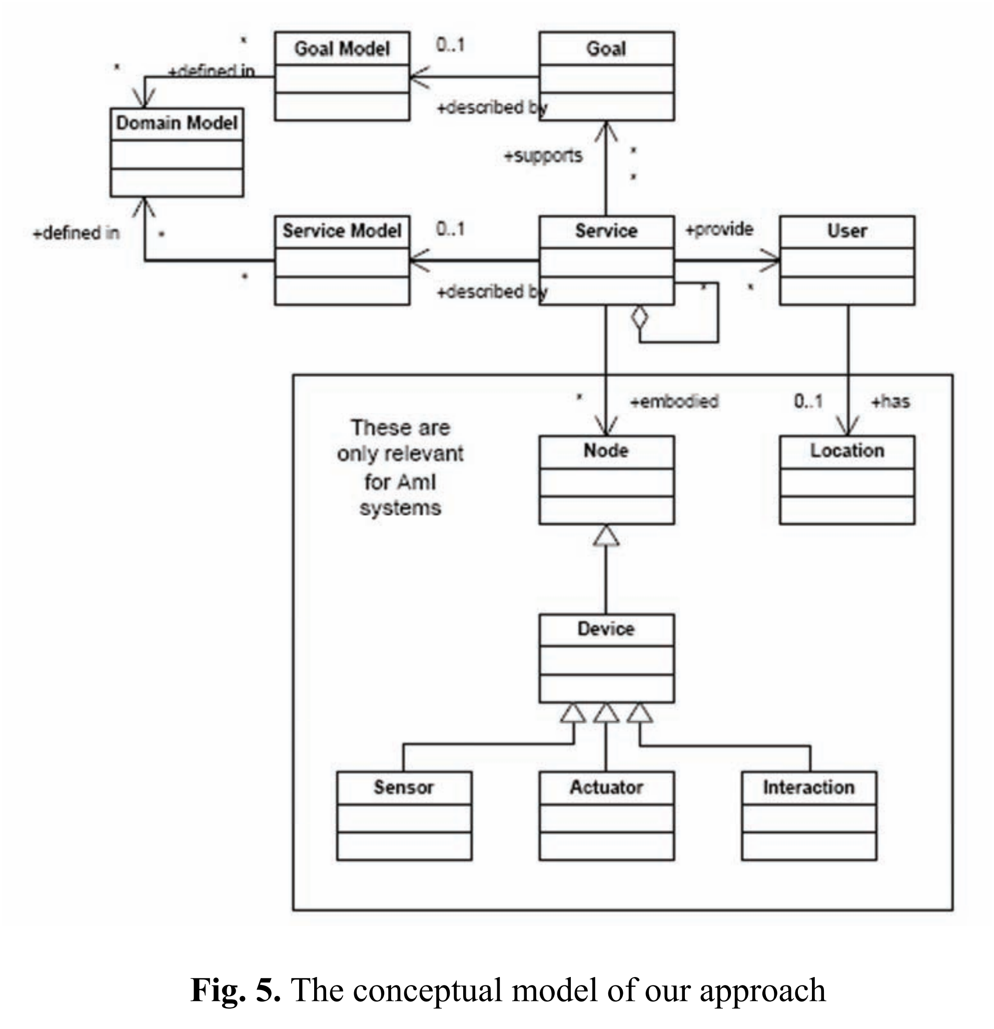


Fig. 5. The conceptual model of our approach

After defining the goal, we also need to define adequate services that are able to satisfy the goal. It is important to note however, that these services – according to our basic assumptions – are defined by a different organization.

As a result concepts like “room_light” will not be available – or even worse, be used with different definitions.

5 Conclusions

As the demand for adaptive and self-assembling systems increases, the need for adequate engineering approaches to these problems increases as well. While most work in the area of context-aware and flexible systems focuses on technical aspects like specific modeling approaches, frameworks and implementations, we focused in this paper on the need for appropriate requirements engineering techniques.

Difficulties in this area arise from the traditional focus of engineering approaches that guarantee specific behavior with a minor amount of openness or flexibility.

In order to address this problem, we focused on the integration of requirements engineering approaches with traditional knowledge engineering approaches. The particular approach we are using is especially suited for addressing changes in the underlying service context. However, it can also be extended for spatial context and other context factors. This requires particular extensions of the goal-oriented requirements engineering approach. We are currently studying these issues extensively in order to arrive finally at a full-fledged requirements engineering approaches with a focus on context-aware systems.

References

[Aarts04] E. Aarts, Ambient Intelligence: Building the Vision, In “365 days’ Ambient Intelligence research in HomeLab”, online available at: http://www.research.philips.com/technologies/misc/homelab/downloads/homelab_365.pdf

[Ant96] A. Antón; Goal-Based Requirements Analysis; Second IEEE International Conference on Requirements Engineering (ICRE `96), pp. 136-144, 1996

[ASG06] Adaptive Services Grid Project (ASG); http://asg-platform.org/

[BF+99] J. Bayer, O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schmid, T. Widen, and J.-M. DeBaud. PuLSE: A Methodology to Develop Software Product Lines; In Proceedings of the Fifth ACM SIGSOFT Symposium on Software Reusability (SSR’99), 1999

[CAPS05] Proceedings of the Workshop on Context Awareness for Proactive Systems (CAPS 2005); P. Floréen, G. Lindén, T. Niklander, K. Raatikainen (Eds.), Helsinki University Press 2005; http://www.cs.helsinki.fi/hiit_bru/conferences/caps2005/

[CN01] P. Clements, L. Northrop; Software Product Lines: Practices and Patterns; Addison-Wesley, 2001

[DAML06] OWL-based Web Service Ontology, OWL-S; Version 1.1; http://www.daml.org/services/owl-s/1.1/

[DEAS05] Proceedings of the International Workshop on the Design and Evolution of Autonomic Application Systems (DEAS); D. Garlan, M. Litoiu, H. Müller, J. Mylopoulos, D. Smith, K. Wong (Eds); 2005. http://www.deas2005.cs.uvic.ca/

[Lam03] A. Lamsweerde, Goal-Oriented Requirements Engineering: Goal-Oriented Requirements Engineering: from System Objectives to UML Models to Precise Software Specifications, ICSE '03 Tutorial, Portland, 2003

[ML+05] B. Mrohs, M. Luther, R. Vaidya, M. Wagner, S. Steglich, W. Kellerer, S. Arbanowski; OWL-SF – A Distributed Semantic Service Framework; Proceedings of the Workshop on Context Awareness for Proactive Systems, pp. 67-78, 2005

[OSGI] The OSGi Service Platform, http://www.osgi.org.

[PBL05] K. Pohl, G. Böckle, F. van der Linden; Software Product Line Engineering: Foundations, Principles, and Techniques; Springer, 2005

[Prz05] M. Przybilski; Distributed Context Reasoning for Proactive Systems; Proceedings of the Workshop on Context Awareness for Proactive Systems, pp. 43-53, 2005

[PS+04] V. Poladian, J. Sousa, D. Garlan, and M. Shaw. Dynamic configuration of resource-aware services, Proceedings of the 26th International Conference on Software Engineering, Edinburgh, Scotland, pp. 604-613; 2004

[RSA98] C. Rolland, C. Souveyet, and C. Achour; Guiding Goal Modelling Using Scenarios; IEEE Transactions on Software Engineering, Vol. 24, No. 12, pp. 1055-1071, 1998

[Sat01] M. Satyanarayanan, Pervasive Computing: Vision and Challenges, Personal Communications, Vol. 8, No. 4, 2001

[SEG05] K. Schmid, M. Eisenbarth, M. Grund; From Requirements Engineering to Knowledge Engineering: Challenges in Adaptive Systems, Workshop on Service-oriented Requirements Engineering (Soccer) at the International Conference on Requirements Engineering (RE'05), 2005, also available as Publication of the Fraunhofer Institute for Experimental Software Engineering, IESE-Report No. 118.05/E

[ST05] M. Sbodio and W. Thronicke; Specification and Design of Framework-Based Context Processing Modules; Proceedings of the Workshop on Context Awareness for Proactive Systems, pp. 79-91, 2005

[YM98] E. Yu and J. Mylopoulos; Why Goal-Oriented Requirements Engineering Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality; pp. 15-22; 1998

[W306] http://www.w3.org/TR/rdf-syntax-grammar/

[Wei91] M. Weiser, The computer for the 21st century, Scientific American, September 1991

[WSMO06] http://www.wsmo.org/TR/d2/v1.1/

An Approach Using Memory-Based Reasoning for Determining the Behavior of Mobile, Location-Aware Services

Olivier Coutand, SianLun Lau, Sandra Haseloff, Olaf Droegehorn, and Klaus David

University of Kassel, Chair for Communication Technology,
Wilhelmshoeher Allee 73,
34121 Kassel, Germany
{coutand, slau, haseloff, droegehorn, david}@uni-kassel.de
http://www.comtec.eecs.uni-kassel.de/

Abstract. The concept of context awareness aims to build novel services that adapt to the situation of the user. In context-aware frameworks, adaptation is often performed by applying predefined rules that associate the user's context with some actions performed by the service. Services are adapted, but only in contexts that have been foreseen by the service designer or the user and for which rules have been defined. When the action of the service is not specified for a user's context, the system can no longer carry out adaptation. In this paper we present our approach for determining the service behavior expected by the user when no action is specified. We discuss the various steps that address this issue, considering the particular case of location-aware systems, i.e. systems that adapt their behavior to changes in a user's location. We also describe the architecture of the system that implements our approach.

1 Introduction

Context awareness aims to custom-design services to the user's context, to automatically select services and devices settings. Using these services demands less user attention. It frees users to interact directly with services in order to obtain responses, for example by typing a series of key strokes to input information. This is all the more suitable for mobile services, because the user can only devote partial attention to the services, concentrating on his primary task of moving [1].

Context-aware frameworks have been developed that provide a middleware layer to adapt services to the user's current context ([2]-[6]). These frameworks allow the gathering of context information from sensors placed in the user's environment, and transform these sensor data into high-level and human understandable context information. According to the user's context the system can determine the service behavior that satisfies the user the most. In various users' contexts, the service performs different actions, for example modifying the output mode (text or audio) or the types of information it displays.

The following example illustrates the user interactions with a context-aware service.

> *David is a music fanatic. He listens to music nearly all the time. To satisfy his passion, he uses a novel music player on his mobile device. The music player provides David with play lists adapted to his likings in his current context. For instance, when David wakes up the music player plays a play list with rock songs. When David has breakfast he rather prefers listening to jazz songs. The music player adapts its behavior accordingly.*

Current context-aware frameworks deal correctly with such a scenario, where service behaviors can be clearly identified with regard to a particular user‘s context ([2], [3], [5], [6]). An expected service behavior is related to a particular user's context via a conditional expression (e.g. *when my context is "A" then play rock music*).

This approach however presents a major limitation. It implies that users' contexts are defined beforehand in an exhaustive way. For each context, a service behavior is specified. However this does restrict the way the user can utilize the service. For example, David must write the following rules to specify the behavior of the services in the morning:

WHEN I wake up at 7 am, on a work day, at home, THEN play rock music (1)
WHEN I wake up at 8 am, at the weekend, at home, THEN play jazz music (2)

But what about David waking up in a hotel room, having slept late, etc? In these cases, the current context of David is not expressed in any rule. Thus no action is specified for the service. David must write a new rule to specify what he expects. Otherwise until he does, he will be notified (bothered) every time no rule can be applied to his current context.

In this paper we propose an approach that addresses this limitation and aims to determine services' behaviors when no adaptation action has been defined for the user's context. In particular we focus on the location of the user. The approach is based on memory-based reasoning that uses specific user actions from the past. It aims to compare the similarities between locations.

The paper is structured as follows. In section 2, we provide a brief review of the current context-aware systems and highlight their limitations when it comes to perform actions in non-specified contexts. In section 3 we define the most important terms and we describe our approach in section 4. In section 5 we depict the architecture of the system that implements our approach. Finally section 6 concludes the paper.

2 Current Context-Aware Systems

Context is defined as "any information that can be used to characterize the situation of an entity, where 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 the application themselves" [9]. As such context is composed of a number of elements. These elements are for example user's location, user's environment, user's identity etc. Comparing contexts and computing the similarities between two contexts is therefore very arduous. In our work we only focus on the location element in order to reduce complexity.

Many context-aware frameworks have been developed to provide a middleware layer to adapt applications and services to the user's current context. As pointed out in [7], issues addressed by these frameworks can be classified in three categories: gathering context, interpreting and managing context and adapting the service behaviors.

Context gathering approaches aim to collect raw context data from heterogeneous sensors and augment these data. Interpreting context consists of transforming raw sensor data into human understandable high-level contexts. High-level contexts are composed of data from different context data sources or of different context types (location, temperature, etc). Interpretation is carried out by using predefined rules as in [3], [6]. A slightly different approach is presented in [2], where context is modelled and interpreted based on ontology. Following this approach context is represented as predicates written in OWL [8]. Context interpretation is performed by a context reasoning engine that supports RDF-S and OWL [8] reasoning and general rule based reasoning.

Service behavior adaptation is performed by determining the adequate reactions to events and interpreted contexts. Mobile services adapt their behavior to the context by applying predefined rules. In [2] actions are triggered by a set of rules whenever the current context changes. Service developers write pre-defined FOL (First Order Logic) rules and specify what method is invoked when a condition becomes true. All the rules are saved in a file and pre-loaded into the context reasoner component of the system. A similar approach is carried out in [6] where an inference engine component based on CLIPS is used.

Current context-aware frameworks use predefined rules to both interpret contexts and adapt service behaviors. In contexts where the rules no longer apply, no action can be carried out until the current rules are modified.

3 Using Context-Aware Services

In our approach we use the past user actions to determine the expected service behaviors. This way, we are able to address the limitations imposed by the use of rules in context-aware services. Before detailing the approach, we define the most important terms we use throughout this paper.

Context awareness enables *services* to adapt to the current context of the user. A service is a self-contained, stateless object that accepts one or more requests and returns one or more responses through a well-defined interface. A mobile service is a service that is accessed from a mobile device. Following this definition, a mobile service can independently reside on a remote server or on a mobile device.

The responses of a service are referred to as *service behaviors*. We should distinguish between the *gross* and the *detailed* behaviors of a service [10]. A service is designed to provide one or a set of *gross* behaviors (playing music, enabling chat, displaying messages, etc). When it comes to adaptation, the gross behavior of a service should remain the same. For instance, a music player must remain a music player irregardless of the context. However the *detailed* behavior of a service can change with the context. The type of music played or the algorithm that is used to construct the play list may be adapted. As stated in [10] behavior changes result in a parameterization rather than a change in the code of the function that is provided by the service.

Our aim is to determine the service behavior in a context for which the system does not know what the desired response of the user is. We call this context as an *unknown context*. Inversely, when the service behavior is specified, the context is said to be a *known context*.

The desired response of a user from a service depends on the user behavior. The behavior of a user characterizes his likings and preferences in any situation. It also determines the user's expectations towards systems. We base our work on the postulate that in similar contexts the behavior of the user remains the same and as such, that the user demands towards a service are identical. For instance, when a user is used to listening to music when he wakes up at home, we postulate that it is likely that the user behaves the same way (i.e. listen to music) when he wakes up in a hotel room.

However the potentially many elements, that are used to compose context, will make it challenging to predict the user behavior and consequently the expected service behaviors. Indeed it is difficult to exhaustively determine all the elements that are relevant for describing the current context of an individual. The influence of the context elements may also differ according to the user's situation. For example, when the user is in a hotel room, his context description can be more strongly affected by his activity (having a meeting) than by its location (hotel room).

As noticed in [1], context-aware systems have been developed in different research areas, e.g. tourist guidance, shopping, office visitor information, etc. In these studies the location of a user is the main element used in the systems to adapt their behaviors. In our work, in order to predict the service behaviors we also constrain context to the location element only.

4 Service Behaviors in Unknown Locations

4.1 Approach

Context-aware systems are expected to provide adapted and personalized services to users. It appears that the success and the proliferation of context-aware services in everyday life, as depicted in many scenarios, will depend on their acceptance by end users. The effort to use context-aware services should be significantly less than their benefits to the user. Thus, systems have to provide two capabilities that are somehow contradicting to each other. On the one hand, they have to be minimally intrusive, demanding minimal user attention while using them. This means that for instance the user does not have to press key strokes to select the service behavior every time he uses the service. The user also should not have to input new rules every time an adaptation cannot be properly carried out. On the other hand, systems have to meet user expectations and rarely behave unexpectedly. In other words, the systems should react exactly the way the user expects it to.

As mentioned in section 3, we base our work on the postulate that in similar contexts (e.g. locations) the behavior of the user is the same. As such, the user demands towards a service are identical. Thus, in order to determine the service behavior, we propose to analyze the similarities between an unknown location and other, known locations. By considering the similarities, we can then retrieve the pair of locations that best matches. Consequently, we deduce the service behavior of the unknown location from the service behavior of the known location.

Our approach is based on reasoning from the past user actions. In the literature this is referred to as memory-based reasoning [11]. The goal of memory-based reasoning is to make decisions by looking for patterns in data. Unlike rule induction it does not observe the patterns to create classification rules, but rather it induces similarities between items by making direct reference to past items.

In the following we describe the various operations of our approach in determining the service behaviors expected by a user in unknown locations.

4.2 Basic Operations

Building a data set

To perform a selection of the service behaviors based on the past user actions and the similarities between the user's locations, the first operation consists of building a data set of past user experiences. A past user experience (feature) is composed of the user location and the related service behavior for this location.

To build the data set we need then to gather the features that allow the prediction of the service behaviors. The system must monitor all behaviors required by the user and the corresponding user's location. By applying learning algorithms both the interactions of the user with the service and the user's location can be learnt. Hence one obtains a list of locations, each one associated with one or many service behaviors. This list contains the past actions of the user (requested service behaviors) with regard to his context.

Monitoring the user location can be achieved by several means. The user can be located with different positioning systems (like GPS), by identifying the mobile network cell in which the mobile device of the user is located, or by identifying a service point (WLAN, Bluetooth or infrared technologies).

Monitoring the service behavior is more challenging. We propose to specify the gross and the detailed behaviors of a service in the service description. To allow automatic evaluation of the behaviors of a service a service description is required that semantically depicts the service's functionalities and characteristics. One influential work in the area of semantic service description is [12]. It supplies web service providers with a core set of markup language constructs for describing the properties and the capabilities of the service. Thus it answers questions like: what the service requires from the users and provides for them, how the service works and how the service is used.

Measuring similarities between locations

The second operation that must be carried out to determine the service behavior that is demanded by the user in an unknown location consists of computing the similarities between the features of the data set.

Similarity is a widely used concept and definitions can therefore slightly differ. In computer science, similarity depicts the costs of mapping one entity onto another one [13]. When measuring similarities, it is important to find a common reference frame between the concepts. Indeed "A is similar to B" does not contain information as long as the concepts are not specified [13].

When it comes to location, similarity obviously can be characterized by the distance between two concepts (i.e. two locations). For example *my office* and *the corridor next to my office* are more similar than *my office* and *my kitchen at home*. But other aspects can also be taken into consideration for computing similarities. Considering semantic proximity (e.g. office A vs. office B – both locations are office – may be more similar than office A and kitchen C) and affiliation/privacy aspects (my office vs. all other offices) can also help to measure similarities between locations. To develop a function that measures similarities between locations, we should consider these different aspects.

Retrieving the service behavior

Finally the last operation consists of determining the service behavior that is demanded by the user in an unknown location. In this location no service behavior has been learnt from a past experience or is specified e.g. by a rule.

The information collected from past user actions (as discussed in 3.3) provides a set of reference samples (instances). Each instance is characterized by a value that measures its level of similarity to the unknown location. To determine the service behavior, different data mining techniques can be applied. For instance, after the computation of the similarities, the system can find out that the most similar location to *my bedroom* is *my living room*. When using a nearest neighbor algorithm [14], the system can offer me in *my bedroom* the same service behavior as in *my living room.* Alternatively an unknown location can be compared to clusters, where instances are classified according to the service behavior they are related to [14].

5 System Architecture

In this section we present the architecture of a system that implements the approach described above.

The approach aims to facilitate the user interactions with any mobile, location-aware service. As a first design principle, we have decided to develop a middleware system that supports many services. [7] introduces a conceptually layered framework, to associate the functionalities implemented in various context-aware systems to various layers. The system architecture described below follows this framework. However we consider location as the only context element. Thus the bottom three layers of the framework are implemented in the single context processing and gathering layer.

The system architecture is composed of four components that cooperatively fulfil the operations described in section 4 (Figure 1).

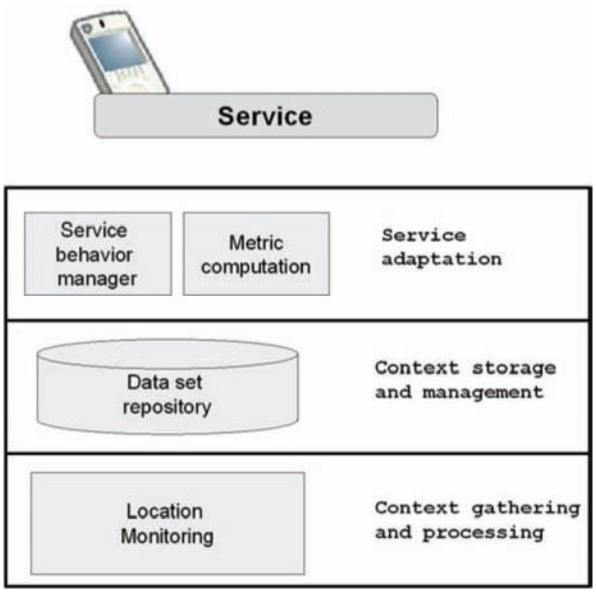


Fig. 1. System architecture

The **service behavior manager component** is responsible for monitoring the service behavior, when the user interacts with the service. To label the service behavior, the component uses the semantic description provided by the service. Every time the user performs an action, the location monitoring component is triggered.

The **location monitoring component** is part of the context processing and gathering layer. According to the method used to locate the user, the component gathers the data from different types of sensors. When different methods are jointly used, for example to determine the user location both indoor and outdoor, the component also processes data to provide a uniform and system-manageable description of the user locations. Indeed location data must be represented the same way, in order to ensure that the similarities can be further calculated.

User locations and related service behaviors are associated by and stored in the **data set repository**. An entry of the database is composed of a user location and the associated service behavior(s).

Last but not least, the **metric computation component** is responsible for applying the location metric to calculate the similarities between the location features (stored in the repository) and the unknown location. When all similarity values have been computed, the component retrieves the most similar locations and applies a decision algorithm. The metric computation component must also deal with the several possible outcomes of memory-based systems [11]: 1) No feature is or a too small number of features are similar to the current location to make a decision. 2) Inversely, there are several possible service behaviors among the nearest *n* features. 3) A significant number of location features are retrieved and all have similar diagnoses. When a service behavior has been identified, the component finally calls the service to request it.

6 Conclusions and Future Work

Context awareness has become a major topic in facilitating user interactions in ubiquitous computing systems. It allows the development of services that automatically adapt to the user's situation and requires less user's attention when interacting with the services.

Current context-aware frameworks have limited capabilities for dealing with unknown contexts, i.e. contexts for which the user demand has not been explicitly expressed.

In this paper we discussed the issue of determining service behaviors in unknown location based on past user actions. Firstly we describe our approach that consists of identifying the behaviors of a service, learning the user demands in some locations, computing the similarities between locations, based on various metrics, and finally applying a data mining algorithm to determine the most appropriate service behavior. Our approach provides the user with an adapted service, even when no service behavior has been specified beforehand. Secondly we propose an architecture of a system providing a support for any mobile, location-aware service that implements this approach.

In order to demonstrate our approach, we are developing an algorithm to calculate similarities between locations. We will also select an algorithm to determine a service behavior based on the set of past user action features.

References

1. E. Kaasinen, "User needs for location-aware mobile services", Personal and Ubiquitous Computing, Volume 7, Issue 1, Pages 70 – 79, 2003
2. T. Gu, H. K. Pung, D.Q. Zhang, "A middleware for building context-aware mobile services", In Proceedings of IEEE Vehicular Technology Conference (VTC), Milan, Italy, 2004
3. M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell and K. Nahrstedt, "A middleware infrastructure for active spaces", IEEE Pervasive Computing, 1(4), pp 74–83, 2002
4. P. Korpipaä, J. Mäntyjärvi, J. Kela., H. Keränen, and E-J. Malm, "Managing context information in mobile devices", IEEE Pervasive Computing, 2003
5. P. Fahy, S. Clarke, "CASS – a middleware for mobile context-aware applications", In Workshop on Context Awareness, MobiSys 2004
6. G. Biegel, V. Cahill, "A framework for developing mobile, context-aware applications. In Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communication, 2004

7. M. Baldauf, S. Dustdar, F. Rosenberg, “A Survey on Context Aware Systems. International Journal of Ad Hoc and Ubiquitous Computing”, forthcoming, 2006
8. www.w3.org (last visited 12.02.2006)
9. A. K. Dey, “Providing Architectural Support for building Context-Aware Applications”, PhD thesis, College of Computing, Georgia Institute of Technology, December 2000
10. S. Dobson and P. Nixon, “More principled design of pervasive computing systems”, Engineering for Human-Computer Interaction and Design, Engineering Human Computer Interaction and Interactive Systems, Joint Working Conferences EHCI-DSVIS 2004, Hamburg, Germany, July 11-13, pp 292-305, 2004
11. C. Stanfill, D. Waltz, "Toward Memory-Based, Reasoning", Communications of the ACM, 29(12):1213-1228, December 1986
12. http://www.daml.org/services/owl-s/1.0/ (last visited 12.02.2006)
13. S. Winter, “Location-Based Similarity Measures of Regions”, .In Proceedings of ISPRS Commission IV Symposium "GIS Between Visions and Applications", in Stuttgart, Germany (7-10 September, 1998), Published by International Society of Photogrammetry and Remote Sensing, International Archives of Photogrammetry and Remote Sensing, Vol. 32, pp: 669-676, 1998
14. I.W. Witten, E. Frank, “Data Mining: Practical Machine Learning Tools and Techniques”, Morgan Kaufmann, Second Edition, 2005, ISBN 0-12-088407-0, 2005

Applying Rescorla–Wagner Model to Need Awaring Multi-Agent System

Ohbyung Kwon[1], Keunho Choi[1], Sungchul Choi[2]

[1]School of International Management, Kyunhee University
449701 YonginSi, South Korea
{obkwon, kino4u}@khu.ac.kr
[2]Department of Industrial and Management Engineering, POSTECH
790784 PohangSi, South Korea
blissray@postech.ac.kr

Abstract. This paper aims to propose a proactive need awaring mechanism by applying agent, semantic web technologies. The Rescorla–Wagner model is adopted as an underlying need awaring theory. We have developed a prototype system, NAMA (Need Aware Multi-Agent)-RFID, to demonstrate the feasibility of the need awaring mechanism proposed in this paper. The NAMA agent considers the context, the user's profile including preferences, and information about currently available services, in order to discover the user's current needs and then link the user to a set of services, which are implemented as web services. Moreover, to test if the proposed prototype system works in terms of scalability, we performed a simulation and describe the results. User acceptance test was also conducted to show how individual differences affect the intention to use the context-aware services such as NAMA-RFID.

1 Introduction

Out of many context-aware services, a reminder system has been proposed as a method to increase quality of care in the area of personal services such as medical examination, safe driving, and advertisement [6, 7, 17]. Abowd proposed that an ideal reminder system should be characterized by the following abilities [1]:

- Proactively provides a variety of contextual data, which includes not only time and location, but also richer context information.
- Accepts input from the user about events, tasks, *etc.* of which to be reminded.
- Creates reminder contents from a variety of input devices.
- Allows the user to receive suitable reminder contents with any devices.
- Accepts and uses all signal and description types of contents.
- Displays the full list of available reminder systems to the user.

However, legacy reminder systems do not satisfy all these functions mainly because they cannot collect context data to be used to operate the reminder service. Additionally, prior systems are limited in their ability to proactively determine the user’s needs. Meanwhile, although not a few prototypes using context aware technology have been proposed, methodologies that could automatically recognize the current user’s needs are still very rare. It is difficult to build awareness of the user’s dynamic needs *a priori* in a specific context.

Hence, the purpose of this paper is to propose a need awaring methodology to automatically estimate and recognize the user’s needs. To do so, we consider not only existing context aware computing technology, but also a need-aware mechanism and a multi-agent based intelligent system. As a basic theory of being aware of user’s needs based on context, the Rescorla-Wagner model is newly visited. Semantic web and web service architectures are adopted for automatic and intelligent coordination between user and task agents. To show the feasibility of the need awaring methodology proposed in this paper, we implemented a prototype system called NAMA (Need Awaring Multi-Agent system)-RFID.

2 Need Identification

A representative model in respect to need identification in marketing is CBB (Consumer Buying behavior). The CBB model is organized as six phases: need identification, product brokering, merchant brokering, negotiation, purchase & delivery, and product service & evaluation [14]. Among these, in the need identification stage, a buyer becomes aware of some unmet self-need. Since the unmet needs could be identified as problems by the unmet needs owners, this stage is also called as problem recognition stage in the Engel-Blackwell model [5]. Despite its importance, most prior research has not focused on the need identification phase, simply because need identification has been believed to be the last to be controlled by the computer-based system [11]. Prior approach was passive in identifying needs that occur in a specific environment. Agent technology has been used to learn and predict the user’s needs by assuming that a lot of historical data about a user’s transaction was available. For example, Eyes and Firefly have been regarded as the systems that follow an agent-based approach. [8, 9, 11, 12]. First, Amzon.com developed notification agents called Eyes. This agent recommends new books based on prior data of user’s book consumption. Eyes notifies the customer when a series of events occur which may be of interest to the customer. It infers user’s need by analyzing common features that correlate to the user's choices and behaviors. Two other agent systems similar to Eyes are Fast-Parts.com’s Auto Watch and Exicite.com’s Cool

Notify. Another system is Firefly, an automated recommendation mechanism that uses a collaborative filtering method. Firefly recommends products via an automated "word of mouth" recommendation mechanism called collaborative filtering. However, Firefly is not pure need identification process, but rather is closer to a product finding agent.

Consequently, since legacy need identification is passive and static, it cannot identify the user’s dynamically changing needs when these changing needs occur in a series of context. This situation frequently happens when users are nomadic and hence the context changes continuously. A more proactive and dynamic need identification mechanism should be required for context-aware services.

3 Rescorla–Wagner Model and Need Awareness

In this paper, need identification based on context awareness is called “need awareness” to differentiate existing need identification. A need awareness mechanism is designed based on associative theory [4, 13, 16]. Among the associative theory models, we revisited Rescorla–Wagner which has been used to infer human behavior for several decades. According to the model, the social learning approach to discovering causality assumes the operations of *association principle* and *contextual principle* [10]. The model suggests that the strength of association (V) is determined by conditional stimulus by associative principle, and unconditional stimulus by contextual principle. In addition, the change of the strength of association (ΔV) consists of the strength of conditional stimulus (α), the strength of unconditional stimulus (β), and the differences between the strength of association of unconditional stimulus (λ) and the strength of overall association. (V_T) as (1):

$$\Delta V = \alpha\beta\,(\lambda - V_T) \tag{1}$$

Rescorla-Wagner's model has important implications for need awareness. First, a buyer may form an intention to purchase not only through product information conditionally provided, but also through the contextual stimulus that is unconditionally provided. Second, these two kinds of stimuli have their own strength levels, which also influence the strength of association. Hence, to accurately identify a user's need, a system must acquire not only the contextual data detected when an object is exposed to the user, but also the object itself via product information that may be gathered from the product's website. Lastly, association can be revealed only by unconditional stimulus. In this paper, unconditional stimulus is regarded as context. Fig. 1 shows that the need identification is revealed by obtaining product information as a stimulus when the user’s wish-to-purchase list is in the user’s memory or in her/his secondary storage device. At that time, a set of context data may also be provided to the user as unconditional stimuli. If these stimuli are accumulated, then the user could perceive needs within a set of context data, say unconditional stimuli, without direct stimulus such as wish-to-purchase list.

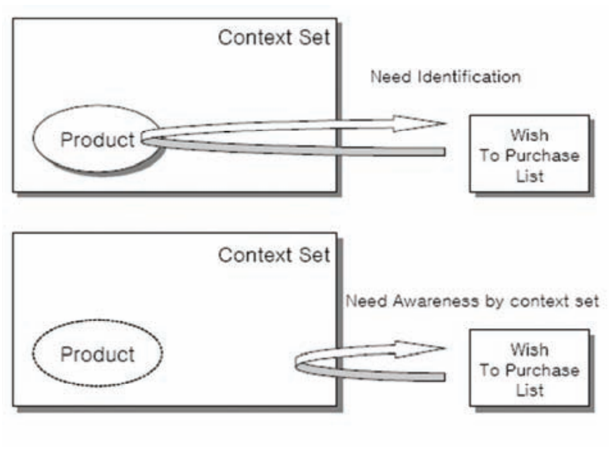


Fig. 1. Need Awareness by context set and Need Identification

4 NAMA-RFID

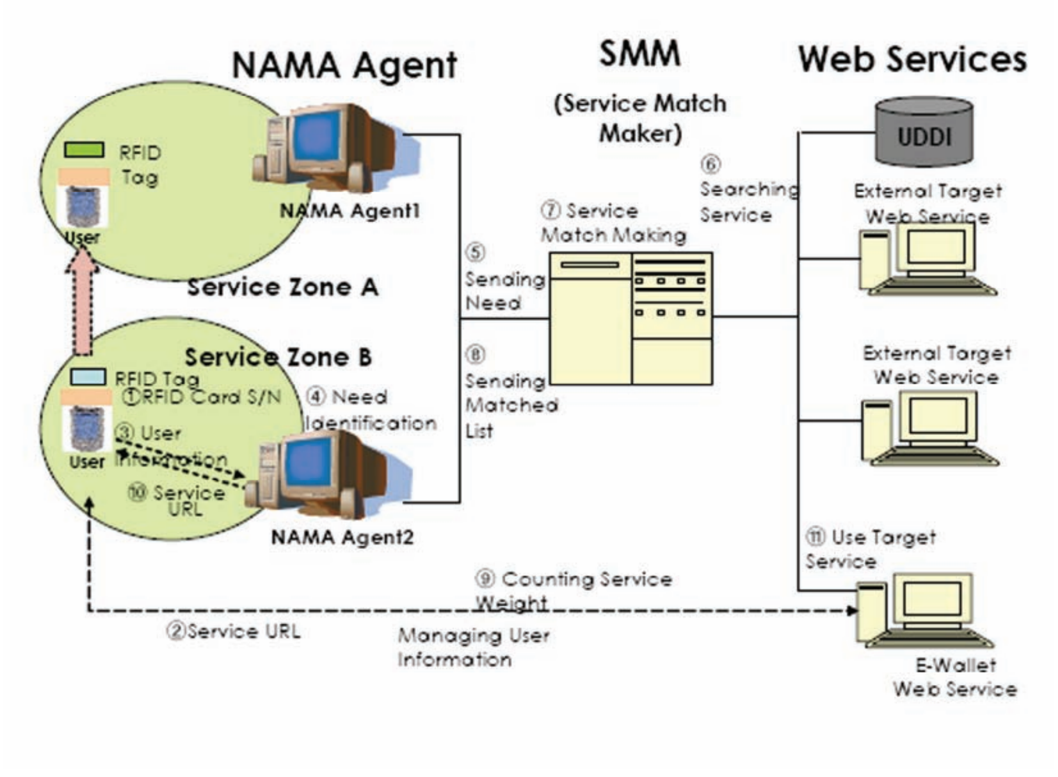


Fig. 2. NAMA-RFID Architecture

Based on Rescorla–Wagner model, a need awaring agent has been proposed in this paper to autonomously make decisions on the user's behalf by proactively reasoning the user's association based on (2). To automatically identify the user's unrevealed need, the NAMA agent uses a specific agent on the user's behalf. Since RFID-based context sensing is adopted for NAMA, the system is called NAMA-RFID system. The NAMA-RFID's architecture system is divided into three parts as shown in Fig. 2: NAMA agent, Service Match Maker (SMM), and Web Services.

The NAMA agent plays three important roles. First, it controls the user's current location by using context data in TILE (Time, Identity, Location, and Entity) format. The context data is basically used to find the most fitting reminding result to users. Second, the NAMA agent queries the user agent in the NAMA agent's service zone for whom the user agent is working, and requests the user information if a series of reminding services is needed. In service zones such as shopping malls, department stores, or streets, the NAMA agent should control and manage a considerable volume of context data since users are probably walking around. Finally, the NAMA agent provides the user with proper service by determining the user's need based on the user's prior context data. In this paper, case-based reasoning method is applied for learning from context data. Based on the TILE context format, we construct a CBR algorithm to perform need awareness from context data as follows:

==

Stage 1: Recognize a set of context as $C_{ij} = < T_{ij}, I_{ij}, L_{ij}, E_{ij} >$.
Stage 2: Attain inferred context, compound context and decomposed context from $C_{ij} = < T_{ij}, I_{ij}, L_{ij}, E_{ij} >$. Then obtain augmented context set, $C^A_{ij} = < T^A_{ij}, I^A_{ij}, L^A_{ij}, E^A_{ij} >$, including the attained context.
Stage 3: Obtain the intersection set, $C^I_{ij} = < T^I_{ij}, I^I_{ij}, L^I_{ij}, E^I_{ij} >$, between $C^A_{ij} = < T^A_{ij}, I^A_{ij}, L^A_{ij}, E^A_{ij} >$ and unconditional context set, $C^C_{ij} = < T^C_{ij}, I^C_{ij}, L^C_{ij}, E^C_{ij} >$.
Stage 4: Identify $C^I_{ij} = < T^I_{ij}, I^I_{ij}, L^I_{ij}, E^I_{ij} >$ as problem set with 'to purchase list.' Then obtain the most similar case using the nearest-neighbor retrieval method using the following function

$$Similarity(C^I, S) = \sum_{j=1}^{n} f(C^I{}_j, S_j) \times w_j$$

where S indicates a set of cases stored in the case base and w indicates a weighted value of each feature of the case base.

==

SMM (Service MatchMaker) uses a service matchmaking method that incorporates the above features. SMM receives the user's need from the NAMA agent to determine a set of relevant services for the user's need. Then

SMM visits UDDI which the web services enroll in their own deploying URIs, which indicates the service ontology of a target web service. The UDDI does not itself have the task web service information directly, as a way to enhance the NAMA agent's autonomy and cooperation more than a direct query done by the user [15].

5 Implementation

To show the feasibility of the idea for need awareness proposed in this paper, we have developed an RFID-based context-aware multi-agent system, NAMA-RFID. NAMARFID uses RFID and semantic web based location-tracking technologies, and Protégé-2000 to create and maintain OWL-based service ontologies. The OWL-based file is parsed and managed by task web services using JENA 2.0 API. Pointbase embedded in the personal device is used to manage the users' personal data preferences which can be restricted from sharing. JAVA SE 1.4.x, the standard VM of Sun Microsystems, is used for the JAVA virtual machine. EVM of Jeobe is the virtual machine for the Pocket PC. We developed the system on Windows XP Professional and LINUX-based Redhat 9.0. IBM's J9, which follows CDC and personal profile of Sun Microsystems, is used to develop the client system for PDA. We compiled the system on WSDD release 5.1 and built execution files as jxe, a link file of J9, and common jar files. kSOAP is furnished to the PDA client device to directly invoke web services.

6 Scalability Test

6.1 Experiment Design

To verify the prototype system proposed in the paper in terms of scalability, we actually built a test bed site and conducted an experiment with a simulation method. Table 1 shows the overall system specifications to simulate the proposed system.

Table 1. System specifications

Server	Petium4 2.8
	1Gb RAM
	Windows XP SP1
	Tomcat Server: jakarta-tomcat-4.1.30
	Axis 1.2

	Access DB
	Java version 1.4.2_04-b05
SMM (Service Match Maker)	Pentium4 1.6Ghz
	256 RAM
	MS Windows Server 2003
	Tomcat Server: jakarta-tomcat-4.1.30
	Axis 1.2
	Java version 1.4.x
Service Zone MatchMaker	Pentium4 2.0Ghz
	512 RAM
	Windows XP SP1
	Tomcat Server: jakarta-tomcat-4.1.30
	Axis 1.2
	Access DB
	Java version 1.4.2_04-b05
E-wallet Web Service	Celeron CPU 1100Mhz
	448 RAM
	Windows XP SP1
	Tomcat Server: jakarta-tomcat-4.1.30
	Axis 1.2
	Java version 1.4.2_04-b05
Communication environment	100Mb-class network

6.2 Results

According to the number of web services and mean service time and size of queue, a total of 220 scenarios are generated. For each scenario, 100 simulation runs are performed, and the results are obtained. For example, Table 2 shows the results in conducting the simulation when the size of queue is 10.

We could find some implications from this evaluation. First, in reading and processing RFID data, a critical problem in terms of service time and error rate of service failure was not exposed during the simulation. An RFID-based event triggering method turns out to be appropriate in a need awaring system like NAMA-RFID. Second, however, scalability problems with adjusted mean service time seemed to occur in web service invocation, especially invoking UDDI. Next, the middleware software, Tomcat and AXIS, seem to have limitations in adequately managing scalability. Considering the simulation data, when INT=0.1, size of queue=100, and WS (web service)=10, errors arise in connecting to the NAMA agent. Otherwise, when INT=0.1, size of queue=70, and WS=10, error rate was reduced considerably. In other words, a significant problem happens when multiple user agents try to connect to the system at the same time. Last, the system must consider the stability and safety of UDDI. When inspecting the SMMException, we could see that errors arose in connecting to UDDI. This would be more critical as the number of user agents and NAMA agents increased because in that case the number of invocations to UDDI access would also increase. In conclusion, the scalability issue happens not just to the NAMA-RFID system itself but to the web services, especially UDDI. Hence, NAMA-RFID architecture proposed in this paper is applicable in the real world when the fundamental issues of web services are resolved.

Table 2. Simulation results (size of queue = 10)

		Number of web services									
		10	20	30	40	50	60	70	80	90	100
Inter arrival Time	0.1	32.5	72.9	117.6	203.3	260.7	326.0	325.2	570.7	584.0	594.9
	0.2	37.8	89.1	183.4	210.2	285.7	300.0	526.9	586.2	534.4	541.7
	0.4	36.4	105.0	145.8	271.8	353.6	275.1	331.2	593.4	557.1	565.4
	0.6	26.9	73.9	125.2	179.6	217.3	262.8	145.6	584.6	587.0	590.6
	0.8	33.3	37.4	46.6	50.4	31.9	36.7	95.4	474.2	580.9	556.2
	1	42.0	91.2	150.4	183.2	187.2	294.7	584.6	531.9	574.3	580.8
	2	30.9	62.7	220.4	327.1	325.8	430.8	362.6	444.0	415.1	407.0
	4	18.0	17.9	14.0	21.9	13.0	20.9	18.0	21.8	17.9	21.9
	6	18.9	25.7	15.0	6.1	15.0	15.0	19.0	19.9	14.0	19.9
	8	16.1	12.0	19.8	12.0	20.0	12.0	19.8	19.8	20.0	13.0
	10	23.0	13.0	22.8	21.5	21.8	14.0	23.9	14.0	15.0	14.0

7 User Acceptance Test

This paper also aims to test how individual differences affect the intention to use the context-aware services such as NAMA-RFID. It is crucial to examine which dimension of user profile is useful in market segmentation and hence should be considered to deploy the NAMA-RFID system. Personal innovativeness is the individual's willingness to try out any new information technology. Self-efficacy has been one of the individual characteristics in prior researches [2]. Perceived sensitivity on contextual pressure, which is rooted in neuroticism in psychology, will be newly considered if it could be a significant determinant of NAMA-RFID system usage. Our base model is the technology acceptance model (TAM) originated by Davis, because of its reputation of accurately explaining whether users will use a particular technology or not [3].

A survey technique was used to collect data. The population sample was selected from among managers, information system experts in Korean companies, and business students at a major Korean university. There were total of 206 usable survey responses out of 240. The sample consisted of 53.1% male and 46.9% female participants, whose ages ranged from 20's to 50's. In selecting participants, we chose users who already had some understanding of NAMA-RFID system.

A confirmatory factor analysis using LISREL 8.7 was conducted to test the measurement model. The fitness of the overall measurement model was estimated by various indices provided by LISREL (see Table 3). Consequently, we could proceed to examine the path coefficients of the structural model.

Table 3. Fit indices for measurement and structural model

Goodness-of-fit measure	Recommended value*	Measurement Model	Structural Model
Chi-square /degree of freedom	≤ 3.00	1.64	1.79
Goodness-of-fit (GFI)	≥ 0.90	0.90	0.89
Adjusted goodness-of-fit (AGFI)	≥ 0.80	0.86	0.85
Normalized fit index (NFI)	≥ 0.90	0.96	0.95
Non-normalized fit index (NNFI)	≥ 0.90	0.98	0.97
Comparative fit index (CFI)	≥ 0.90	0.98	0.98
Root mean square error of approximation (RMSEA)	≤ 0.10	0.06	0.06

* Recommended values have been adapted since Hair *et al.* (1998).

Table 4 shows path coefficients, t-values, and causal paths, including an explained variation of structure equation model of the presented model. We discovered that perceived sensitivity on contextual pressure (PSCP) affects perceived ease of use and perceived usefulness. This implies that a person who is highly sensitive on contextual pressure perceives that the NAMA-RFID system would be useful in dissolving her/his problems, and hence is willing to try NAMA-RFID system. Therefore, even though the user with higher PSCP feels uncomfortable to use the innovative system for the first time, the user would be more positive because of its usefulness. These findings imply that more customized services should be considered to those who have more level of personal sensitivity on contextual pressure.

Table 4. Results of test of NAMA-RFID acceptance model

	Beta	**t-value**	**R^2**
Perceived ease of use			0.485
= Personal innovativeness	0.31	4.72***	
+ Self-efficacy	0.56	8.17***	
+ Perceived sensitivity on contextual pressure	-0.14	-2.29**	
Perceived usefulness			0.425
= Perceived ease of use	0.66	6.55***	
+ Personal innovativeness	-0.01	-0.11	
+ Self-efficacy	-0.00	-0.03	
+ Perceived sensitivity on contextual pressure	0.12	1.90*	
Behavior intention to use			0.565
= Perceived ease of use	0.17	2.29**	
+ Perceived usefulness	0.63	7.26***	

Note: * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$. Beta: standardized coefficients

8 Conclusion

This paper aims to build a need awaring mechanism, which is constructed by context-awaring and agent-based web service on personalized services. To do so, we have developed a reminder system whose requirements and architecture, NAMA-RFID, are proposed. For needs awareness, we applied the Rescorla-Wagner model, an association theory, to show that the user could be reminded only with context, not direct stimuli such as product and services which are declared *a priori* by the user's wish-to-purchase list.

We expect to contribute to actually evaluating the performance of the system which uses ubiquitous computing technologies such as web services, agent, and RFID-based sensing technology. Considering performance evaluation, we examine the problem that legacy web service middleware and UDDI may cause scalability concerns to actually deploying web service technology. A user acceptance test is also performed to determine whether NAMA-RFID could be actually deployable. The outcomes show us that persons who have more personal innovativeness, self-efficacy, and perceived sensitivity on contextual pressure are more likely to accept the NAMA-RFID system. User interface issues turn out to be another important determinant to increase the acceptability of NAMA-RFID system. We believe these results could be generalized in examining the acceptance of context-aware systems.

9 Acknowledgement

This research is supported by the ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

References

1. Abowd, G.D. Software Engineering Issues for Ubiquitous Computing. Proceedings of the 21st International Conference on Software Engineering (ICSE '99), Los Angeles, CA (1999), 75-84
2. Compeau, D. and Higgins, C. Computer Self-efficacy: Development of a Measure and Initial Test. MIS Quarterly, 19 (1995), 189-211
3. Davis, F. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly, 13 (1989), 319-340
4. Dickinson, A. and Shanks, D.R. Animal Conditioning and Human Causality Judgment. In L.G. Nilsson & T. Archer (Eds.), Perspectives on Learning and Memory, Hillsdale, NJ: Lawrence Erlbaum Associates, Inc., 1985
5. Engel, J. and Blackwell, R. Consumer Behavior, 4th ed. CBS College Publishing, 1982
6. Kralj B., Iverson D., Hotz K., and Ashbury F.D. The Impact of Computerized Clinical Reminders on Physician Prescribing Behavior: Evidence from Community Oncology Practice. American Journal of Medical Quality, 18, 5 (2003), 197-203
7. Kwon, O, Choi, S.C. and Park, G. NAMA: A Context-Aware Multi-Agent Based Web Service Approach to Proactive Need Identification for Personalized Reminder Systems. Expert Systems With Applications, 29, 1 (2005), 17-32
8. Maes, P., Robert H., Guttman, R., Alexandros G. and Moukas. Agents that Buy and Sell: Transforming Commerce as we Know It. Communications of the ACM, 37. 7 (1999), 31-40
9. Pivk, A. and Gams, M. E-commerce Intelligent Agents. Proceedings of ICTEC'00 (2000), 418-429
10. Rescorla, R. A. and Wagner, A. R. A Theory of Pavlovian Conditioning: Variations in the Effectiveness of Reinforcement and Non-reinforcement. A. H., & Prokasy, W. F. (Eds.), Classical conditioning II: Current Research and Theory, NY: Appleton-Century-Crofts, 1972
11. Robert H., Guttman, R., Alexandros G., Moukas and Maes, P. Agents as Mediators in Electronic Commerce. Electronic Markets, 8, 1 (1998), 22-27
12. Rockinger, R. and Baumeister, H. BABSy: Basic Agent Framework Billing System. Proceedings MAMA 2000, Wollongong, Australia (2000)
13. Shanks, D.R., Holyoak, K.J. and Medin, D.L. The Psychology of Learning and Motivation, 34, San Diego, CA: Academic Press, 1996
14. Terpsidis, I., Moukas, A., Pergioudakis, B., Doukidis, B. and Maes, P. The Potential of Electronic Commerce in Re-Engineering Consumer-Retail Relationships through Intelligent Agents. J.-Y. Roger, B. Stanford-Smith, and

P. Kidd (eds.), Advances in Information Technologies: The Business Challenge, IOS Press, 1997

15. Trastour, D., Bartolini. C. and Gonzalez-Castillo, J. A Semantic Web Approach to Service Description for Matchmaking of Services. Proceedings of the International Semantic Web Working Symposium (SWWS), Stanford, CA, USA (July 2001)
16. Wasserman, E.A. Attribution of Causality to Common and Distinctive Elements of Compound Stimuli. Psychological Science, 1 (1996), 298–302
17. Williams AF, Wells JK and Farmer CM, “Effectiveness of Ford's Belt Reminder System in Increasing Seat Belt Use,” Injury Prevention, 8, 4 (2002), 293-296

Context-Aware Preferences for Internet Services?

Pekka Jäppinen

Lappeenranta University of Technology
P.O.Box 20
53850 Lappeenranta
Finland
Pekka.Jappinen@lut.fi

Abstract. Modern services change their behaviour based on the preferences of the customer. The preferences of people depend on different contexts from their physical location to their current role. Therefore the context has to be taken into account when service is personalised. This paper describes how customer preferences acquiring is affected when context is also taken into account. Different methods used for personal information acquiring by Internet services are analysed from the context-aware preference point of view.

Keywords: personalisation, personal preferences, identity management, context awareness, privacy

1 Introduction

Due the evolution of the communication technologies and access devices the way we use Internet today has changed from the days when Tim Berners-Lee, along with his colleagues, developed WorldWideWeb in CERN. Back in the 90's, Internet services were used from fixed location with a desktop computer that had a decent monitor. Browsing the service content was done by a mouse and a keyboard was used for inputing data. The convergence of tele- and datacommunication networks allowed an access to the services via new types of devices. At the same time new types of services and service access emerged.

Today the access to services may be done via variety of devices using different types of access methods and communication paradigms (Figure 1). This variety provides new challenges for service providers. Service providing cannot rely that customer have big screen, reliable and fast connection and proper equipment to input data effectively. In order to provide good use experience for the customer, the service have to be adapted for the user hardware capabilities. These capabilities can be acquired in the form of User Agent Profile defined by 3GPP. Furthermore, solutions to the challenges provided by different hardware used on Internet services is researched in

W3C Device independence workgroup [1]. The hardware capabilities is not the only challenge that have emerged. With the new small terminals the user's are not fixed to a certain location for service access and use. The addition of Wi-Fi capabilities to mobile phones allows the use of free regional wireless networks [2] instead of rather expensive cellular technology such as GPRS.

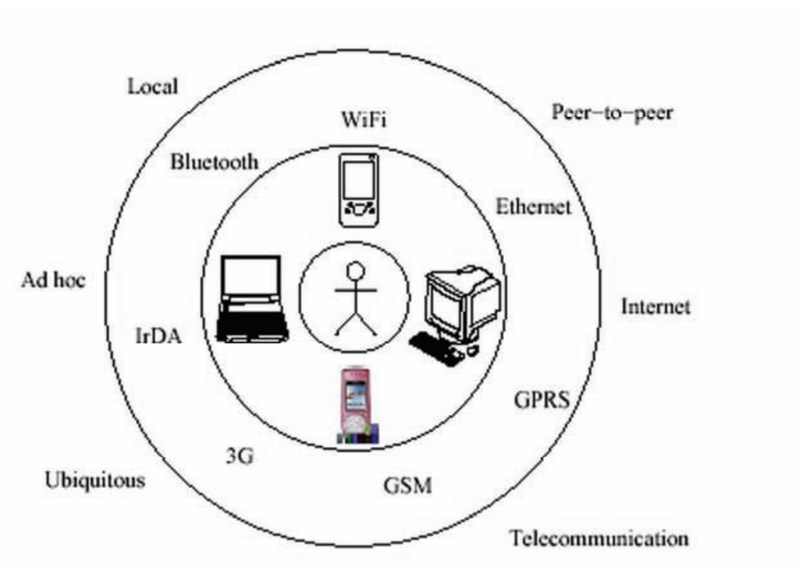


Fig. 1. Heterogeneous service access and use

Due to these new challenges and possibilities the personalisation of services for customers is becoming more important. Mobile customers are less willing to browse through several pages in order to find the information or product they are looking for. Effective personalisation will also provide better customer satisfaction and thus increase the customer's loyalty towards the service [3,4]. In order to conduct the personalisation of the service, the service requires information about the customer's preferences. These preferences can be acquired from the customer by asking them for example during registration to the service. Another possibility used for example by amazon.com is to monitor the behaviour of customers during their service use and gather data that can be used to personalise the service.

When the services are used through mobile devices, the context of where the customer is, may change more than when using the desktop computer at home. For example, the location of desktop computer user will never change during the service use while mobile phone user may move from the street full of sunlight down into the darkness of the subway station. These changes may affect to the preferences the customer has i.e. the preference has dependency on come context information. For example, in the direct sunlight user prefers light background with dark letters while in the darkness the background is preferred to be dark and letters light. The preferences may also be dependant on several context. For example, the preferred type of lunch is different when customer is having lunch in short lunch break at work than it is on vacation in some exotic location. Several dependencies for personal preferences have been identified in [5].

Taking context into account for determining user preferences will require changes in the way the personal preferences or personal information is managed. In this paper the current methods of personal information management are evaluated in Chapter 2. Chapter 3 discusses how context-awareness can be taken into account and what kind of problems have to be solved to get to context-aware identity. Chapter 4 concludes the paper.

2 Personal Information Management

In order to personalise services the service has to have access on personal information. The location where the preference data is stored and how it is accessed affects also how the context data can be taken into account. The traditional way for acquiring the data is by letting user fill a registration form that asks variety of questions and the service then stores this user profile in database. However that is not the only possibility. If we look at the ways services are accessed, several possibilities can be identified. Figure 2 presents these possibilities, namely 1) registration, 2) database at network, 3) browser and 4) Mobile device based approaches.

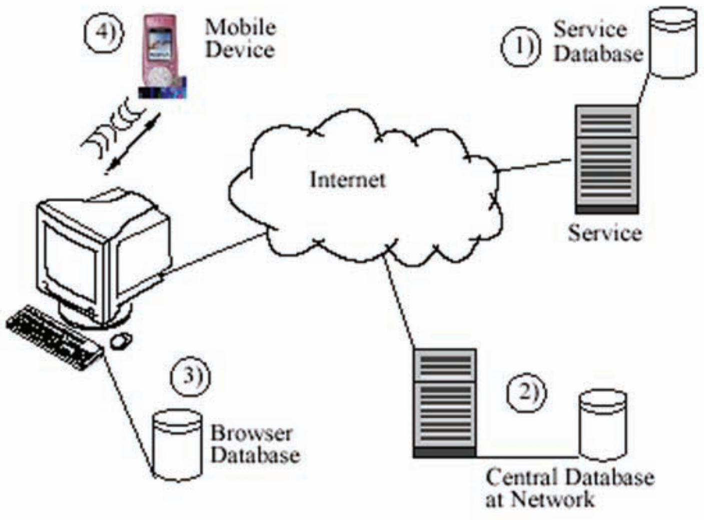


Fig. 2. Different locations for personal information

If we take a look at the registration approach. First of all users have to register for the service before they may start using it. The service stores the information about the customers, i.e. user profile, into its database for the further usage. Thus next time the user accesses the same service he doesn't have to provide the same information again. Instead, the service looks up the user profile from its database. In order to find the correct personal information from the database a user authentication is required. Besides the basic identifying information, services may request the user's preferences, follow the user's behaviour on the server and store the gathered information into appropriate database. The stored information can then be used to personalise the service i.e. adapt the service outlook and content into more customer friendly form.

There is some drawback on this approach. As the variety of the used services grows, the places to where the user has to provide his personal information grow and the confusion about the contents of the user profiles increases. If user preferences change, the user needs to know all the places where the information is stored and update the data there. It is unlikely that the user remembers all these places thus resulting contradictory information. Another problem arises when considering the security of the personal information. When the information is stored in the service database, user has to trust that the service provider is capable of storing the information securely enough. The more services user uses, the bigger is the chance that one of them has weaknesses in their security system. To minimise the amount of databases the personal information is stored, the customer can easily decide to concentrate to only few service providers.

Second possibility is to store the personal information in a single database on the network. This database can be administrated by a trusted third party or the customer. Some of the single sign-on (SSO) approaches such as Microsoft .NET passport [6] and Liberty Alliance [7] can be used to provide a third party based solution for the personal information storage problem. However, originally .NET passport required the service provider to pay a fee to Microsoft. This meant that small service providers could not afford to adopt the passport. It is also unlikely that temporary services like conference registrations are willing to do the extra work required to join in the SSO system especially when all they need is simple registration information for one time event.

In Liberty architecture service and identity providers form circles of trust in which the participants transfer information about the user. The creation of a circle of trust requires negotiations between the participating partners. Like in .NET passport this is unsuitable for one time services provided by small service providers. In these approaches the user has to trust to the third party and that the third party uses secure methods when handling the user's sensitive data. Also the trusted party has to be same party for both user and service provider. If there are several third parties providing similar personal profiles, the service provider has to have contract with most of them, in order to not exclude any users. Obviously, information stored on third party servers, is available only when a connection can be formed between the service provider and the servers. Thus the use of service relies not only on the reliability of the network between customer and service but also the network between service and the third party. This problem is graver in the countries that have poor international Internet connections. Finally, the third party service may change the way it functions and the terms of use. For example Microsoft has stated that they will cease to offer passport functionality outside their own services. In Microsoft and Liberty approaches the user has no control over his/her profile, which is a serious drawback.

Other centralised approaches have been proposed. Koch proposes IDRepository that allows users to own their own profiles [8]. At the same time this approach supports complex user profile attributes. In Wireless World Research Forums (WWRF) book of visions Bettstetter et al. suggested user controlled personal profile server [9]. More detailed solution was described by Thai et al. [10]. Their Integrated Personal Mobility Architecture relied on the customer's home network as a location to store personal information. Compared to the third party approaches the user controlled database is easier for service providers since they don't have to have contracts with different third parties. The data is available for services when the profile server is up and running. Security conscious user will turn the service on only when needed, thus increasing security.

The third obvious location for the personal information storage is at the user end. New web browsers have the capability to store some information for the user. Thus, the information is stored in the place, from where it is easiest to use and update. There exist two approaches that can be used to determine what information will be filled in an empty form. In the first approach the stored information depends on the web page. The browser remembers what the user has typed into the form fields, when the form was filled before. In the second approach the web page supports the browser's personal information scheme, so that the browser itself can determine what information is given in what field [11]. Although the web browsers will enhance this personal information storage functionality rapidly, it does not help the mobile or nomadic users. Mobile and nomadic users typically access the services from several different terminals at different places.

Final possible location is to store the data at user's mobile device. Mobile Electronic personality (ME) [12] is one such solution, where the data can be requested via Bluetooth connection by the desktop computer using a browser plug-in [13]. Similar approach is defined by the EU funded Simplicity project [14] that concentrates on the use of services through different terminals and devices. In order to provide seamless usage of services in different devices Simplicity project proposes the use of so called Simplicity device to store the user profile. Although several possibilities for the storage are given they conclude that the SIM + smart phone combination is not very far from the ideal storage and processing device.

3 Context and Personal Information

Taking the context into account when deciding the customer preferences will require some changes on existing personal information management systems. Let's see how a context aware preferences can be used. At first let's assume that context dependent personal information is already stored at some database based on the approaches defined in previous chapter. When the customer connects to the service, the service connects to a database to get the personal information. In order to select correct piece of information the customer context has to be to determined. The context is then delivered either to the service or to the database holding the personal information. If context data is delivered to the database, then the database can handle the proper mapping i.e. when service requests customer's favourite drink, it gets the proper response based on the current context of the customer. If the service is given the context data, then service needs to fetch all the preference data and do the context mapping itself. Of course the service can just relay the context data to the database for handling.

Some of the context data can be thought to be private and thus customer may be unwilling to disclose it to foreigners. The idea that the service they frequently use, can gather a database about the changes of their context may sound intrusion to privacy. For example the service may follow the locations from where service is used at which time via the context information. Thus from privacy point of view the service based databases are risk to privacy. With third party approaches the risk is even higher, since the third party gets the context data whenever the users uses any personalised service.

In the mobile device based approach the context data the context data is not transmitted anywhere but is handled locally at the customer's mobile device. The service receives just the preference data that is correct for the given situation. The actual context information remains secret. With this approach it is also possible to use context data to affect on privacy settings in different situations. For example customer may allow restaurant to receive his identity on work hours to gain lunch bonuses, but on his free time he wants to remain anonymous[15].

Another question is, how to describe the context. Let's think about seemingly simple context like location. While location at first sounds like unambiguous piece of data, it can be defined in variety of ways. GPS provides exact coordinates for location, while from preference point of view it might be more interesting to know whether the user location is in shopping mall or on the street [16]. In order to get proper piece of data from the database for the given context, the context provider and context user has to use notation that they both understand. The user should also be able to understand the context for which the preference is defined for. Unified identifiers for different kind of contexts can help the interoperability. On the other hand user's might come up for their own contexts. On some mobile phones it is possible for users to define different profiles, such as meeting, home, or work profile, that affect on how the phone behaves. These profiles can be one source for context and an example of user definable context. The user definable context are easy to understand by the user and thus defining the preferences for these contexts is more direct.

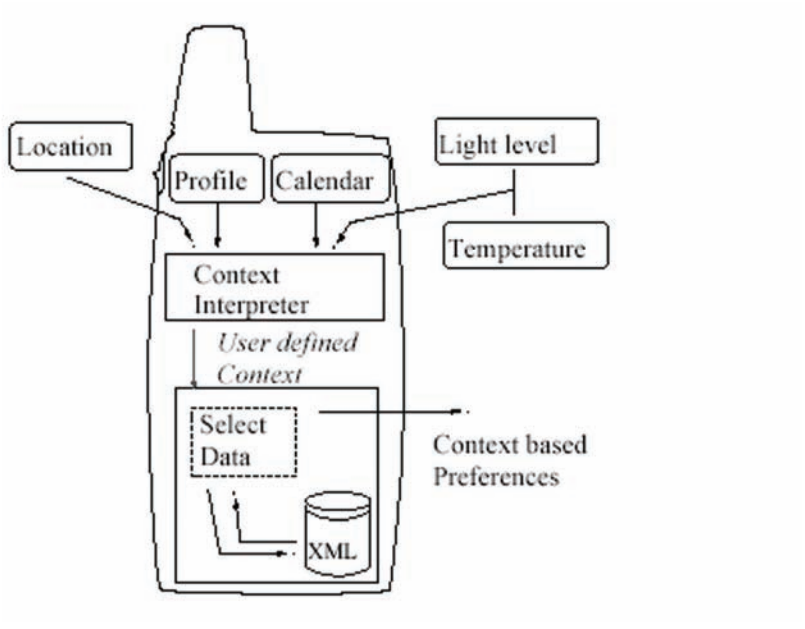


Fig. 3. From several context to user defined context

The user definable contexts can also take input from outside e.g. time. In order to map the context information gained from different sensors to the contexts defined by user, an interpreter is required (Figure 3). The context interpreter has to have specific rules how to combine the different context data into a single context. The result from the context interpreter can be used to find the proper piece of data from the preference database or XML file like in Figure 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<Personality xmlns="http://www.lut.fi/%7Epjappine/ME"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.lut.fi/%7Epjappine/ME
     ME-context.xsd">
          <Lunch>Pizza</Lunch>
          <Lunch context="vacation">Shrimps</Lunch>
          <Lunchtime>1pm</Lunchtime>
          <Lunchtime context="work" >11:30</Lunchtime>
          <Sports>Hockey</Sports>
</Personality>
```

Fig. 4. Simple context-aware preferences XML file

Another problem arises when there are several context dependencies for certain data. For example a person might prefer fast food for lunch when at work. On the other hand same person might prefer local food, when visiting another country. The question is, what is the preference when the user is visiting another country on a work trip? Should new context named “work abroad” be created. This can lead of forming huge amount of contexts and cause confusion for the user. Context interpreter and user defined contexts can help on this problem by reducing the amount of possible context.

4 Conclusions

The personal preferences depend from the context the person is. To provide accurate personalisation the context where customer is should be taken into account. Current personal information management systems require changes in order to support such an approach.

The main issue on supporting context-aware preferences is to find way for matching the preference data with current context. Multiple context sources adds complexity to the whole system. Providing context data to service may easily intrude the privacy of user. Thus services should not get the raw context data but rather just the preference data that has been processed with the context in mind. That reduces the amount of suitable personal information management approaches. Finally defining the preferences in different context may be tiresome task for the user and thus simple interfaces for up keeping the data is required.

The context information itself may be sensitive and thus should be protected to ensure user’s privacy. However context awareness can help user to manage different types of identities and handle security differently on different situation. Before these advantages can be accomplished a lot more research is required.

References

1. Device independence workgroup webpage. http://www.w3.org/2001/di/, 2001. Accessed February 16, 2006
2. J. Ikonen and J. Oksanen, “Wireless LANs and Regional Networks,” in Emerging Personal Wireless Communications (O. Martikainen, J. Porras, and J. Hyvärinen, eds.), (Lappeenranta, Finland), pp. 197–208, IFIP, Kluwer Academic Publisher, Aug. 2001
3. L. Ardissono and A. Goy, “Tailoring the Interaction with Users in Web Store,” User Modeling and User-Adapted Interaction, vol. 10, no. 4, pp. 251 – 303, 2000
4. B. Kasanoff, Making It Personal. Perseus Publishing, 2001

5. P. Jäppinen and J. Porras, "Analyzing the Attributes of Personalization Information affecting storage location," in International Conference on E-Society, IADIS, June 2003
6. Microsoft, ".NET Passport Review Guide." Available at: http://www.microsoft.com/net/services/passport/review_guide.asp, 7 Oct. 2003. Accessed January 20, 2006
7. Liberty Alliance, "Liberty Architecture overview-v1.1." Available at: http://www.projectliberty.org/specs/liberty-architecture-overview-v1.1.pdf, 15 Jan. 2003. Accessed January 12, 2006
8. M. Koch, "Global Identity Management to Boost personalization," in Proceedings of 9th Research Symposium on Emerging Electronic Markets, (Basel, Switzerland), pp. 137–147, Sept. 2002
9. C. Bettstetter, W. Kellerer, and J. Eberspächer, Book of Visions 2000, ch. Personal Profile Mobility for Ubiquitous service Usage, pp. 67–69. Wireless Strategic Initiative, 2000
10. B. Thai, R.Wan, A. Seneviratne, and T. Rakotoarivelo, "Integrated Personal Mobility Architecture: A Complete Personal Mobility Solution," Mobile Networks and Applications, vol. 8, pp. 27–36, Feb. 2003
11. G. W. Bauer, "User data management aka Privacy/Convenience Feature." Available at: http://www.mozilla.org/projects/ui/ communicator/browser/wallet/, 26 Jan. 2006. Accessed January 27, 2006
12. P. Jäppinen, ME - Mobile Electronic Personality. PhD thesis, Lappeenranta University of Technology, 2004
13. M. Yrjölä, P. Jäppinen, and J. Porras, "Personal information transfer from a mobile device to web page," in Proceedings of the IADIS International Conference on WWW/Internet, pp. 485–492, Nov. 2003
14. N. Blefari Melazzi, G. Bianchi, G. Ceneri, G. Cortese, F. Davide, N. Davies, N. Dellas, E. Fischer, T. Frantti, Friday. Adrian, J. Hamarid, M. Helbing, S. Kapellaki, K. Kawamura, W. Kellerer, L. Kotsoloukas, C. Meyer, C. Niedermeier, C. Noda, J. Papanis, C. Petrioli, E. Rukzio, S. Salsano, R. Seidl, O. Storz, J. Urban, I. s. Venieris, and R. Walker, The Simplicity Project: Managing Complexity in a Diverse ICT World, ch. 10, pp. 179–211. IOS Press, 2004
15. P. Jäppinen, "Context Based Preferences with Mobile Electronic Personality," in The proceedings of LCN 2005, The IEEE Conference on Local Computer Networks (LCN), pp. 521–522, Oct. 2005
16. J. Hjelm, Designing Wireless Information Services. John Wiley & Sons, Inc, 2000

The ACAS and Ambient Network Approaches to Context-Aware Networks

Theo G. Kanter[1]

[1] Ericsson Research, KI/EAB/TGB, SE-164 80, Stockholm, Sweden
Theo.Kanter@ericsson.com

Abstract. There is an increasing need for individuals and groups to interact on the Internet using various applications and services that are delivered across an increasingly heterogeneous network infrastructure, such that users can move freely without experiencing difficulties in their interaction with services or each other. This paper presents and compares two complementary approaches to meeting the challenges posed by delivering services to users who move in this infrastructure. Both the ACAS and Ambient Network approaches have as their premise that context acquired from sensors in network infrastructure (including users) enables service delivery support to negotiate heterogeneous infrastructure and improve the user experience, but differ in where and in what layer decisions are made. Finally, the paper discusses the implications of using Context Networks for how the IMS as mobile service delivery infrastructure could evolve to support users who move about in heterogeneous 3G and beyond networks.

1 Introduction

There is an increasing need for individuals and groups to interact on the Internet using various applications and services. At the same time, these individuals and group expect to be able to move connecting via either fixed access from home or from work, but also via wireless access networks when commuting in between, and traveling by e.g. car or airplanes.

So far, research in ubiquitous computing has addressed the challenge of improving the user experience and facilitating the interaction with services such that users can focus on tasks and spend less time on understanding how to make use of communication reaching other people and how to invoking applications and services in order to accomplish these tasks. For instance, the MobiLife project [1] demonstrated the automatic arrangement of services between individuals & groups of people and dynamic participation in group applications, by leveraging so-called context information obtained from users and their use of services.

These obvious benefits for users will be lost unless we address other challenges posed by increasingly heterogeneous network infrastructure and heterogeneous wireless access for delivering services to users who move about in such a heterogeneous infrastructure. Users of mobile communication can access services via multiple access networks with varying properties in terms of price, data rates, latency, error rates, etc. Multiple devices may be used to interact with services, which further adds to the complexity faced by a user. Some of these devices allow truly mobile access due to their form factor, function, etc. – i.e., mobile phones or mp3-players with wireless interfaces. Therefore, we need to enable automatic (re-)arrangement of communication, services, and interaction with services. Moreover, new usage patterns are expected to emerge as a result of minimizing the effort required by users to arrange & manage their communication, services, and interaction across heterogeneous network infrastructure.

We should also assume that such adaptive services need to be centered around the user, as we can no longer rely on a single network or single operator to arrange and manage these services. This causes us to investigate new architectural choices which move service control and coordination closer to the user. But in making these choices, there can be different approaches in where to make decisions concerning service delivery control and their properties can guide the design as to how the network layer and service layer could or should interact in order to provide users with applications and services that are able to negotiate heterogeneous infrastructure.

The challenges of supporting applications and services in negotiating heterogeneous infrastructure has been addressed specifically in two research projects by means of creating so-called Context Networks but with entirely different and complementary approaches. The approaches taken in the ACAS and Ambient Networks projects [4, 2] differ in the sense that the basic premise of ACAS is that decisions should be moved close to the user in end-devices and between entities attached *to* the network. The approach taken in Ambient Networks (as its name indicates) assumes that decisions should be made *inside* the network although this does not preclude functions residing in end-devices to participate in these decisions. The divisor is thus whether an administrative authority exists in the infrastructure which is involved in *both* providing connectivity as well as making decisions concerning service delivery.

Below we will study the architectural choices and motivations of the architectures of the context networks developed by ACAS and Ambient Networks projects, but also present and compare some experimental results that enable us to discuss in what fashion they enable applications and services to negotiate heterogeneous infrastructure and minimize involving users in their decision-making.

Finally, section 5 discusses the implications of each approach in using Context Networks for how the 3GPP IP Multimedia Subsystem could evolve as mobile service delivery infrastructure to support applications and services delivery to users who move about in heterogeneous 3G and beyond networks.

This is relevant to the ongoing studies concerning how 3GPP IMS and the related 3GPP Long Term Evolution / System Architecture Evolution (LTE/SAE) [7] could evolve to build a converged network infrastructure carrying services and application allowing users to individuals and groups to interact on the Internet and move freely without experiencing difficulties in their interaction with services or each other.

2 Context-Aware Networks

A 'Context Network' refers to the context provisioning support for acquiring, processing and disseminating context information to functional entities in the network that need it. The Context Network can involve entities in the link layer, network layer, and application layer. A 'Context-Aware Network' refers to when the network entities use information provided by a Context Network to modify either their individual behavior or their collective behavior. The Context Network employed by Ambient Networks is also known as ContextWare. Figure 1 that provides an overview of the roles of Context Networks adopts this terminology.

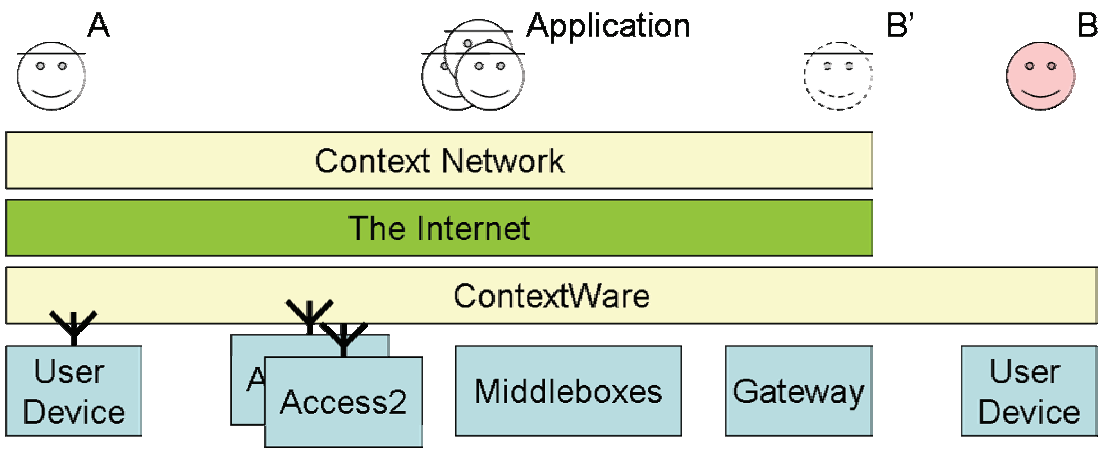
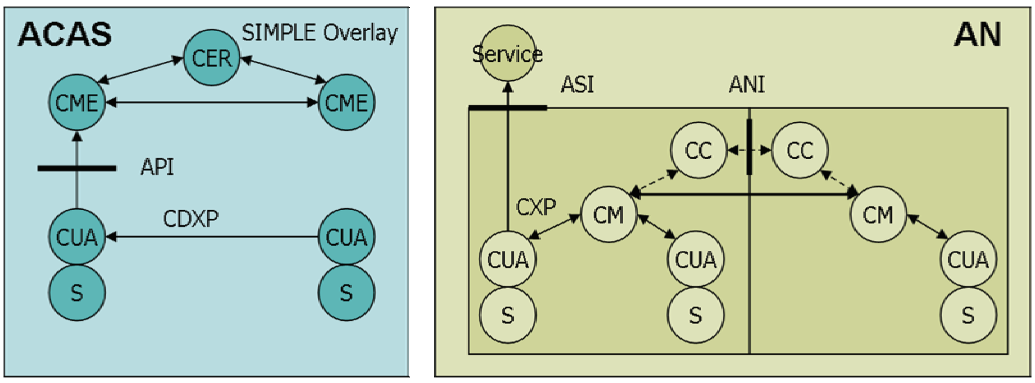


Fig. 1. Overview of Context Networks and ContextWare

The Context Network depicted in Fig.1 as situated *below* IP is used to enhance network entities with ambient properties. These properties may be the ability to dynamically compose, automatically arrange communication paths, or auto-negotiate larger areas of connectivity through the use of with context-awareness functions and thus create an ambient network. It is important to note that the Context Network, that is used to accomplish this, is depicted as below the IP layer only, because the *knowledge* that is carried is belongs to functions that occur *below* the IP-layer. The actual protocols that carry context information to create dynamic networking properties can very well use IP – and in fact they are, making use of the fact that IP offers a connectivity layer. However, the protocols cannot be designed as application layer protocols *on top of* TCP since we would be making too many assumptions of intermediate support being in place, and limit the applicability of the design.

In contrast, the application-level Context Network that is depicted *above* IP in Fig. 1 is used to provide applications with knowledge pertaining to entities *above* IP. Therefore, the knowledge speaks about users and what these users use the communication for. The protocols that implement this Context Network can be application layer on top of e.g. TCP and benefit from this.

The distinction between the two Context Networks that carry knowledge pertaining to either above or below IP is profound. In the first, which is *above* IP with regard to the knowledge it carries and the mechanisms it makes use of, we can represent users and associate the identity of users with the identities of a number of communication devices and resources (e.g., a group server) that the user employs when interacting with services. In the Context Network, which is *below* IP with regard to the knowledge it carries and the mechanisms it makes use of (a.k.a ContextWare), no such information about users is available per default. ContextWare has to rely on the application-level Context Network to register associations between user identities and identities of a number of communication devices and resources in order to be able to make subsequent deductions, such as the rearrangement of communication paths when a user moves and media should be pre-cached in a server belonging to the access work that the user is predicted to move into.

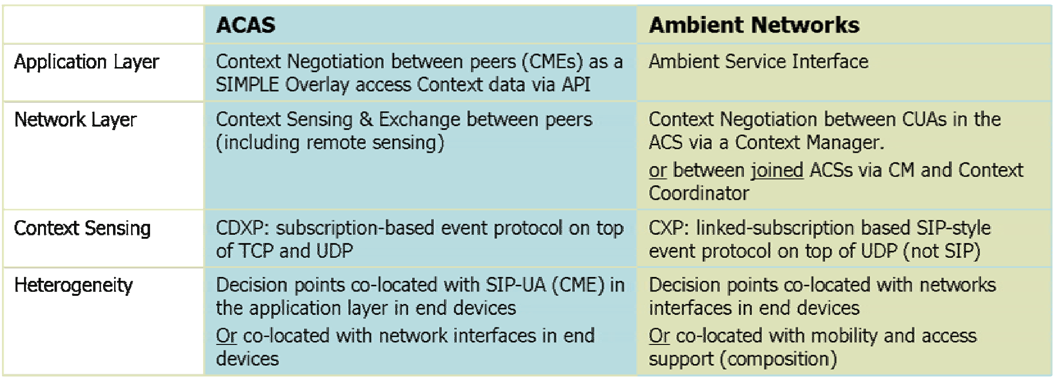


CER	Context Entity Registrar	CM	Context Manager
CME	Context Management Entity	CC	Context Coordinator
CUA	Context User Agent	ANI	AN Network Interface
API	Application Programming Interface	ASI	AN Service Interface
CDXP	Context Data eXchange Protocol	CDXP	Context Data Exchange Protocol

Fig. 2. ACAS and AN Context Networks

3 ACAS and Ambient Networks

In the subsections below, I present the Context Network architectures of the Adaptive and Context-Aware Services (ACAS) as well as the Ambient Networks (AN) projects (Fig. 2) comparing them to the earlier model (Fig. 1). The ACAS and AN Context Network architectures represent entirely different and opposite approaches, which makes them especially interesting to study their potential contributions to how 3G-and-beyond service infrastructure should evolve and address the challenges mentioned in the Introduction.



	ACAS	**Ambient Networks**
Application Layer	Context Negotiation between peers (CMEs) as a SIMPLE Overlay access Context data via API	Ambient Service Interface
Network Layer	Context Sensing & Exchange between peers (including remote sensing)	Context Negotiation between CUAs in the ACS via a Context Manager. or between joined ACSs via CM and Context Coordinator
Context Sensing	CDXP: subscription-based event protocol on top of TCP and UDP	CXP: linked-subscription based SIP-style event protocol on top of UDP (not SIP)
Heterogeneity	Decision points co-located with SIP-UA (CME) in the application layer in end devices Or co-located with network interfaces in end devices	Decision points co-located with networks interfaces in end devices Or co-located with mobility and access support (composition)

Fig. 3. ACAS and AN Comparitive overview

The reason for selecting ACAS for our comparison is that it employs application-layer P2P techniques for context data exchange and includes entities in the services layer, but lacks explicit network composition support as available from ContextWare in the Ambient Control Space, hence providing alternative choices for dealing with network heterogeneity at different layers. Thus, ACAS and AN ContextWare serve as *exemplars* of complementary approaches where ACAS moves communication decisions to end devices without relying on network support, while AN and ContextWare provide network support for communication decisions.

The presentation of ACAS and Context Networks is followed by a characterization of their properties and early results from experiments that provide further information about their advantages and disadvantages in addressing network heterogeneity, which (as mentioned in the Introduction) will put users who wish to interact with mobile applications and services anytime and from anywhere at a disadvantage or even render these unusable.

3.1 ACAS

The Adaptive and Context-Aware Services project has developed context middleware (left-hand side of Fig. 2) that employs Context Management Entities (CMEs) which may register themselves in a Context Entity Registrar

(CER) and use the CER for discovery of other CMEs and context objects in wide-area networks, which unlike local-area networks are not suited to use the discovery support provided by the protocols. Communication between CMEs is application layer and uses an overlay according to SIMPLE [9]. However, the ACAS also enables entities to interact with Context User Agent which acquire and make available local as well as remote sensor information. Thus the application-level Context Network in ACAS has access to network data. ACAS does not rely on any decision support *in* the network. Communication decisions are executed locally or remoted via the application layer.

3.2 Ambient Networks

In contrast to ACAS, the Ambient Networks Context Network (a.k.a. ContextWare) makes no explicit use of application-layer protocols (TCP). The context information concerns network entities and is used to negotiate connectivity, join islands of connectivity, or communication paths to the extent possible without involving the application layer. In his thesis [3], Sergio Quintanilla-Vidales demonstrates the feasibility and scalability of network composition with Ambient Networks ContextWare [8]. The thesis also shows how ContextWare reduces the complexity of end devices and reduces the burdon from computations and communication. Various features have been demonstrated in earlier ad-hoc networking research but the purpose of Ambient Network is to provide a framework in which these can be accommodated dynamically, instead of requiring engineering or demanding user intervention.

The important distinction compared to ACAS with respect to service delivery and application behavior is that decisions guided by ContextWare are entirely uninformed about what applications or users are trying to achieve. On the other hand, any decisions that concern the network only, need not be propagated to application layer.

4 Comparing the Achievements

Below I present a scenario description below adapted from Ambient Networks and use the above comparative model of Fig. 2 to (a) determine the merits of the approaches and alternative choices offered by the Context Network approaches of ACAS and Ambient Networks, and to (b) determine how the design of ContextWare supports making of communication-decisions in order to deal with network infrastructure heterogeneity. Equally important, in the Conclusions I point at some alternative choices with their advantages and disadvantages which have been in addressed in related work or may be addressed in future work.

4.1 Scenario

1. Alice is going to take a train from Amsterdam to a business meeting in Paris. She has a Personal Area Network (PAN) with a smartphone, a PDA, and a laptop configured as a low-complexity ambient network.
2. Alice meets her colleagues Bob and Carol at the train station in Amsterdam. Their PANs compose and form a larger ambient network ("group network"). While waiting, they synchronize their agendas and exchange some documents, which they intend to present during the meeting.
3. Their train arrives. Alice, Bob and Carol get onboard and their ambient network dynamically composes with the train's own ambient network. Thus, new services are available to Alice and her colleagues, such as a stored BBC world-service newscast, access to a repository with other *podcasts* of music videos, and games allowing passengers to play against other passengers.
4. During the trip, the train makes a longer stop at Brussels Midi where the train ambient network composes with an available free Wireless LAN network and shares services. Thus, during the stop, passengers can access the Internet through Wireless LAN. Alice and her colleagues take advantage of this opportunity to access the corporate network and read email.
5. When the train leaves the station, the train's network and the station's network lose connectivity and decompose. Access to the Internet is no longer available, but the train's network has updated its services and passengers have the last version of BBC news among other new contents.
6. The train arrives at Gard du Nord in Paris. Outside the station, Alice and her colleagues are waiting for a taxi when Alice receives a multimedia call from her daughter with her smartphone.
7. Carol finds a free taxi and they get in. The taxi has a vehicular ambient network and their group network composes with it. Alice is notified that there is a bigger screen available (at the back of the front seat) and is offered to redirect the video stream to that screen.

4.2 Scenario Analysis

Step 2 — In step 2, the PANs of Alice and her colleagues compose. Alternatively, in the case of ACAS, we can assume that the CMEs representing them are members of a group CME. In this step and the steps that follow steps, any decisions regarding communication between Alice and her colleagues concerning transport, services, and applications can be guided by the fact that they entertain mutual relations as members of this group. In the case of AN ContextWare, application behavior is outside the model but can benefit from context data that is available via the ASI.

Step 3 — In step 3, networks compose by means of mobility and access support functions available in the Ambient Control Space (ACS) leveraging context available from AN ContextWare. The thesis provides a proof of concept of dynamic composition of ContextWare itself. In ACAS, the CMEs located in the user's end devices can make per packet decisions utilizing multiple wireless interfaces, as demonstrated in Gulio Molas thesis [11], which combined with exchanges of context knowledge between devices, enables composition of transport, services, and applications. Again, in contrast to AN ContextWare, this is initiated from the application layer.

Furthermore, in the case of ACAS, a group CME is able to recognize that the latest news broadcast need not be sent to subscriber since the application employs *PodCasts* and caching in large memories available in end devices. Subscribers can download any missing content again at the next station in step 4 (or decide to rely on a caching server attached to the train network), but these decisions are initiated from the application layer.

Step 7 — In step 7, another device is selected better matching the requirements of the media stream. In ACAS, the Context-Aware Service Allocator co-located with a CME achieves allocation of the service to the screen. In Ambient Networks, users delegate such service allocation decisions to ContextWare network support, which assumes pre-existing relations (expressed in user device and service profiles) between the user and the network. Therefore, future work should address the choices for interaction between the service and network layer.

4.3 Achievements

From the results presented in Sergio thesis [3] alluded to in subsection 3.2 and the above analysis, we conclude that ContextWare helps minimizing the communication and computational burdon on and complexity of end devices. The ACAS approach offers more informed decisions from the application level regarding what the communication should achieve. ACAS can help minimizing the load and complexity of end devices by moving decisions to group servers attached to the Internet.

Network Composition – Ambient Networks are IP-networks enhanced with capabilities to dynamically compose or decompose islands of connectivity connected to the Internet. The prototype of Sergio's thesis [3] successfully demonstrates that the solution enables entities and groups of entities to dynamically compose and arrange communication paths between them. Dynamic network composition, of which the prototype serves as proof of concept, facilitates dealing with heterogeneity of network infrastructure, since these capabilities enable network infrastructure to dynamically arrange new paths of communication, which we can utilize in either transport, in services, or in applications. Such dynamic rearrangements of connectivity are facilitated by pre-negotiable policies for Context Level Agreements combined with Service Level Agreements.

Network Heterogeneity – The design of the ContextWare architecture has a number of merits regarding its contributions for how ambient networks can support communication (transport, services, or applications) in overcoming network heterogeneity.

The ACAS architecture emulates similar behavior described as overlay networks in Gustav Söderström's thesis [10]. The advantage of the approach is that the composition decisions are informed about what the applications are trying to achieve. The disadvantage being that these decisions rely on the fact that connectivity is in place or at least possible to be negotiated in the application layer via B2B relations or customer-service provider relationships.

5 IMS towards Beyond 3G

Finally, the paper discusses the implications of using Context Networks for how the IMS as mobile service delivery infrastructure could evolve to support users who move about in heterogeneous 3G and beyond networks.

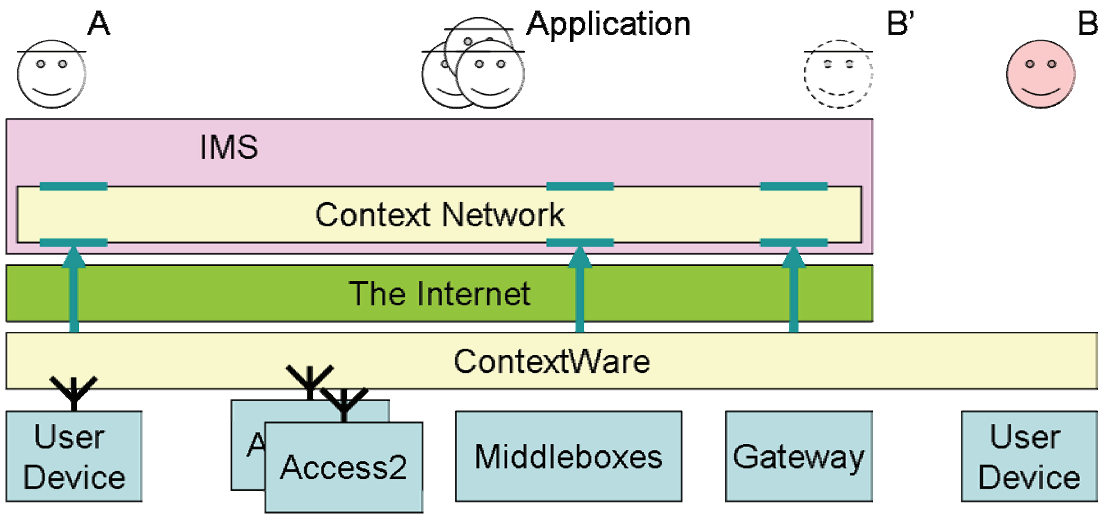


Fig. 4. IMS towards Beyond 3G

The 3GPP IP Multimedia SubSystem is currently positioned to support applications and services that are provided by the same operator who provides access or another operator with whom the first operator has service level agreements with, such as in the case of roaming. The scenarios which were researched in the FP6 MobiLife and Ambient Networks project [6] are not well supported. This is mainly due to underlying architectural decisions.

Service decisions are made in servers in the network. The extended SIP signaling that is gatewayed via the ISC interface to IMS Application servers precludes the adoption of flexible strategies to negotiate heterogeneous infrastructure as present in ACAS. On the other hand the use of SIMPLE for the 3GPP Presence Framework is well positioned to be extended towards Rich Presence according to the GeoPriv specifications in the IETF. However, this requires entities in a more loosely coupled IMS infrastructure to be able to access network data via interfaces which resemble the Ambient Service and Resource Interfaces of Ambient Networks (ASI and ARI). Sergio Quitanilla's thesis [3] demonstrates how this can be achieved in end devices.

6 Conclusions and Future Work

The previous sections highlighted merits of the different approaches taken in ACAS and AN towards overcoming the hurdles of increasingly heterogeneous infrastructure. From this we could deduce some choices for future developments of the 3GPP IP Multimedia Subsystem (IMS) such that it is better positioned to support users who need to interact with services via dynamically arrangements heterogeneous infrastructure and dynamic arrangements of applications, services, and users. However, this study also unearthed large areas that remain left unaddressed and in which further research is needed:

1. The actual semantics of Context carried in Context Networks is not at all well defined, neither in standards nor in research. This threatens to lead to interoperable systems but also raises questions about the actual nature of Context which must be clearly defined, in order for it to be usuable otherwise Context Network will just become a general metadata middleware for different purposes.
2. Interaction across the ASI, recognizing that some decisions in the application layer can be much more information about what the communication is trying to achieve, can be used to improve the overall system performance, as shown in the ACAS project. On the other hand, not every decision should be taken by application layer entities, because it would effectively mean we would duplicate the network control logic in the application layer. Therefore, we need gain insights what the application layer needs to know about infrastructure and a good model for dividing the decision-making.
3. Furthermore, in some dynamic network situations we may expect context changes to occur frequently by also to be highly unstable, e.g., ontology changes. In such cases with less predictable network situations, P2P operation of the context exchange (e.g., CDXP in ACAS) may be advantageous. This mandates further research in a combined approach in

which a Context Manager could delegate or assume/reclaim responsibility for context exchanges between CUAs, complementing the distributed approach with e.g. Distributed Hash Tables [12].

References

1. The IST FP6 MobiLife project: http://www.ist-mobilife.org/
2. The IST FP6 Ambient Networks project: http://www.ambient-networks.org/
3. Sergio Quintanilla-Vidal, "Context-Aware Networks: Design, Implementation and Evaluation of an Architecture and a Protocol for the Ambient Networks Project", forthcoming M.Sc. Thesis Linköping University, 2006
4. The ACAS project, http://psi.verkstad.net/acas/
5. Andreas Wennlund, "Context-aware Wearable Device for Reconfigurable Application Networks", Masters of Science thesis, Department of Microelectronics and Information Technology (IMIT), Royal Institute of Technology (KTH), Stockholm, Sweden, March 2003. ftp://ftp.it.kth.se/Reports/DEGREE-PROJECT-REPORTS/030430-Andreas_Wennlund.pdf
6. Ambient Network, Scenarios, Requirements and Draft Concepts, ST-2002-507134-AN/WP1/D02. Available from http://www.ambient-networks.org/publications/Appendix%20of%20D1_2%20public.pdf
7. UTRA-UTRAN Long Term Evolution (LTE) and 3GPP System Architecture Evolution (SAE) http://www.3gpp.org/Highlights/LTE/LTE.htm
8. D 6.3 Ambient Networks ContextWare - Second Paper on Context-Aware Networks. Available from http://www.ambient-networks.org/publications/D6_3_Ambient_Networks_ContextWare_Second_Paper_on_Context-Aware_Networks_PU.pdf
9. SIP for Instant Messaging and Presence Leveraging Extensions (simple) Charter http://www.ietf.org/html.charters/simple-charter.html
10. Gustav Söderström, "Virtual networks in the cellular domain", Masters of Science thesis, Department of Microelectronics and Information Technology (IMIT), Royal Institute of Technology (KTH), Stockholm, Sweden, March 2003.
11. G. Mola, "Interactions of Vertical Handoffs with 802.11b wireless LANs: Handoff Policy", Department of Microelectronics and Information Technology (IMIT), Royal Institute of Technology (KTH), Stockholm, Sweden, March 2004.
12. Distributed Hash Tables entry in the Wikipedia: http://en.wikipedia.org/wiki/Distributed_hash_table

Efficient Indexing of Symbolic Location Information

Tobias Drosdol[1], Dominique Dudkowski[1], Christian Becker, Kurt Rothermel

University of Stuttgart,
Institute of Parallel and Distributed Systems (IPVS),
Universitätsstraße 38, 70569 Stuttgart, Germany
`<firstname.lastname>@ipvs.uni-stuttgart.de`

Abstract. Providing context-awareness in future applications demands for accurate information about the positions of a potentially large number of mobile objects. This requires location management systems that are capable of coping with a huge amount of such highly dynamic data. To efficiently retrieve relevant information, e.g., all objects located inside a given area, these systems must utilize adequate index structures. In the past, suitable indexes have been explored for geometric representations of location information only. The contribution of this paper is a generic index structure for the efficient management of symbolic location information. Based on different index variants an adaptive approach is derived that can be adjusted to accommodate different ratios of update and retrieval load. Our experimental results show how this can significantly increase the overall performance of a location management system.

1 Introduction

The current location of mobile objects is generally considered as an inherent part of a user’s context [14] and plays a crucial role in today's ubiquitous computing systems. Future context-aware applications will rely on highly accurate location information of a large number of mobile objects to support novel application scenarios, e.g. factory resource management systems [2]. Supporting such applications requires a *location management system* that provides efficient and generic access to relevant location information. Such a system stores the current positions of mobile objects and enables the efficient retrieval of relevant data. This includes spatial retrieval operations, like querying all objects in a distinct spatial area (*region queries*) and monitoring an area for the arrival or leaving of objects (*region monitors*).

An important aspect in realizing such systems is the supported representation of location information. For geometric coordinates a number of location services have been proposed [7, 11, 12]. They typically aim at

[1] funded by the German Research Foundation (DFG) within the Center of Excellence (SFB) 627.

scalable location management and support a variety of queries, object-based as well as region- and distance-based. For such systems, there exists a variety of spatial indexes [6] that accelerate data access by organizing locations within a multi-dimensional space spanned by a geometric coordinate system. These indexes partition the managed area geometrically – leading to hierarchical structures, such as Quad- or R-Trees [6, 16]. Consequently, such indexes require object locations to be represented as geometric shapes. In contrast, especially indoors a variety of positioning systems are solely based on *symbolic identifiers*, i.e., well-defined symbols such as "Cafeteria" [8]. This representation can provide a valuable abstraction of location information at the same time. Humans can more easily interpret descriptive identifiers because their names may contain an implicit expressiveness and often are self-documenting. From a system point of view, however, symbolic identifiers lack spatial relations. To enable region-based retrieval operations, the spatial inclusion relationship between these identifiers must be explicitly defined by means of a suitable *location model* [1, 12].

Geometric index structures are not directly applicable to manage symbolic location information. Yet, an appropriate location management system must still provide fast execution of all retrieval operations without delaying any position update. Furthermore, the system should be able to cope with both a large number of mobile objects and many applications. This requires frequent update operations to be processed in a timely manner. At the same time, the number of processable queries should be maximized. In this paper, we investigate the efficient organization of symbolic location information, in order to optimize the overall throughput of the system. We present two efficient index structures that support a broad range of location models and enable the efficient processing of frequent location updates and region-based retrieval operations. Then, an adaptive indexing approach is derived from these two variants. It enables fine-grained adjustments to accommodate different load characteristics. An experimental evaluation of these indexes indicates how calibration can increase the performance of location management systems.

Existing work in supporting symbolic coordinates [e.g. 9, 10, 12] focuses mainly on the expressiveness of location models with respect to the supported spatial relations but not on the efficient management of mobile objects and query processing. An approach taken by some projects is to convert symbolic coordinates into a geometric representation [10, 12]. However, this is suboptimal for two reasons. First, the underlying location model would have to define this mapping for each location, which would in turn increase the required manual effort for creating and maintaining such a model considerably. Second, the conversion would introduce significant runtime overhead for each update and retrieval operation. In contrast, Brumitt [4] and Leonhardt [12] have described the use of standard relational databases to maintain symbolic positions of mobile objects as well as the location model itself. Yet, with this approach resolving the spatial inclusion relations always requires a recursive search through the database table. This has to be performed on each region query as well as on each position update, in order to trigger all registered region monitors. This overhead could be minimized by utilizing the known hierarchy of spatial inclusions. To the best of our knowledge, performance issues for the management of symbolic coordinates have not been investigated so far.

The remainder of this paper is structured as follows: The next section describes the underlying system model and Section 3 presents our approaches to efficient indexing of symbolic coordinates. Subsequently, the indexes are evaluated in Section 4. Finally, Section 5 concludes the paper with an outlook on future work.

2 System Model

The task of a *location management system* (LMS) is to provide applications with the current positions of mobile objects (MOs). The LMS receives position updates from MOs or tracking systems and executes retrieval operations on behalf of various applications. In order to provide this functionality on symbolic coordinates, the system must operate on a suitable location model. In the following, we describe our assumptions with respect to such a model and then discuss the resulting functionality in detail. For the scope of this paper, we focus solely on *region-based* retrieval operations, including region queries and monitors. An overview of the resulting architecture is depicted in Fig. 1.

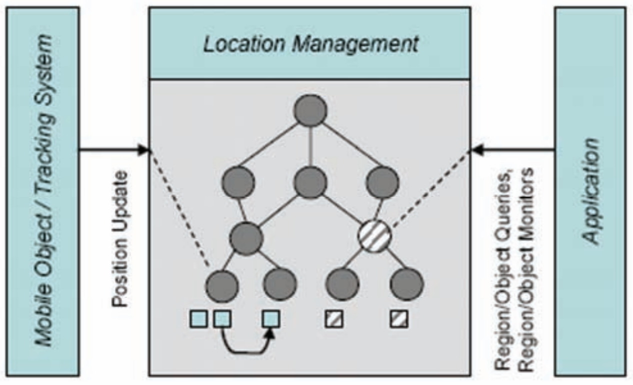

Location Management
Mobile Object / Tracking System
Position Update
Region/Object Queries, Region/Object Monitors
Application

Fig. 1. location management system

2.1 Location Model

A location model must reflect all relevant spatial relations between the symbolic coordinates. In order to support a broad range of different location models, an LMS should only require a minimal set of properties from the underlying model. Ideally, it should be independent of the location model that defines the spatial relations. At the same time, retrieval operations require a location model to allow spatial reasoning. For the scope of this paper, we therefore require that spatial inclusions between different locations are modeled. Thus, let **L** = (L, R_l) denote a symbolic location model, where $L = \{loc_1, \ldots, loc_n\}$ defines the set of symbolic locations and $R_l = \{(loc_i, loc_j), \ldots, (loc_r, loc_s)\}$ is a transitive, irreflexive relation that defines the spatial inclusion between two locations (such that loc_i contains loc_j iff $(loc_i, loc_j) \in R_l$). In its most general form, such a model can be interpreted as a directed acyclic graph. Each node represents a symbolic location, and each directed edge represents the fact that the target location is spatially included in the source location. Note that this assumption is not overly restrictive and is supported by a large variety of existing location models. We further define $loc_i \in L$ to be a *leaf* iff it does not include any other location: $loc_i \in L_{leaf} := \{loc_j \in L \mid \neg\exists\ (loc_j, loc_k) \in R_l\}$. Finally, the *level* of any location $loc_i \in L_{leaf}$ is defined as $level(loc_i) = 0$, and for all $loc_i \in L \setminus L_{leaf}$ as $level(loc_i) = 1 + max\{level(loc_j) \mid (loc_i, loc_j) \in R_l\}$. Consequently, the *height* of a location model is defined as one plus the maximum level, that is, $height(\mathbf{L}) = 1 + max\{level(loc_j) \mid loc_j \in L\}$.

2.2 Service Primitives

The service primitives we consider in this paper consist of position updates and retrieval operations. The latter can be classified along two dimensions. With respect to the subject of interest an operation can be either *object*-based or *region*-based. In addition, we distinguish two different modes of interaction. The current state of the subject can be either *queried* (synchronous operation) or *monitored* for changes (asynchronous operation). Consequently we consider the following operations:

Position Update: Each position update sets the location of an MO identified by *oid* to the specified location loc_i. Value *oid* is a system-wide unique identifier for a certain MO and loc_i denotes a symbolic location of a location model **L**. Since leafs of this model represent the finest granularity of locations, we assume that position updates always refer to a single node on the lowest level (leaf):

$$PositionUpdate\ (oid, loc_i \in L_{leaf})$$

Object Query: An object query returns the current position of an MO specified by its *oid*. The position of the MO is represented by a location loc_i that corresponds to the last position update for that MO. Hence, it always represents a leaf of the location model as well.

$$ObjectQuery\ (oid) \rightarrow loc_i \in L_{leaf}$$

Object Monitor: Monitoring a certain MO will report any change in its position to the subscribed observer. The subscription will remain active until explicitly unsubscribed. Again, the monitored MO is identified by its *oid*. The observer is identified by a callback address *obsaddr* to which change notifications are sent.

$$Subscribe_{OM}(obsaddr, oid) \rightarrow_{\text{async}} loc_i \in L_{leaf}$$
$$Unsubscribe_{OM}(obsaddr, oid)$$

Region Query: A region query returns all MOs (represented by each object's *oid*) that are currently located within the queried region (identified by a given location loc_i). In contrast to position updates, the location specified in a region query may denote *any* node of the location model. All MOs whose positions are included in that location will be part of the result set.

$$RegionQuery\ (loc_i \in L) \rightarrow \{oid_1, oid_2, ...\}$$

Region Monitor: Monitoring a certain region generates change reports on the set of MOs located inside that region. Two elementary *types* of changes exist, in which an MO is either entering or leaving the given region. Both can be subscribed separately.

$$Subscribe_{RM}(obsaddr, type, loc_i \in L) \rightarrow_{\text{async}} oid$$
$$Unsubscribe_{RM}(obsaddr, type, loc_i \in L)$$

This set of operations can also be used to realize a variety of additional, more complex functionality. For example, continuous queries on regions or objects can be composed of a one-time query and subsequent monitoring; the provided monitoring primitives can contribute to a general framework for monitoring complex *context events* [3]; and the managed location information may be combined with additional information about the mobile objects [5]. Yet, all such extensions will likewise benefit from an efficient realization of the presented core functionality.

2.3 Data Storage

In order to provide the presented operations, an LMS must maintain the most recently reported position of each registered MO. The corresponding data entry for each MO consists of the *oid* and a symbolic location *loc*. Due to the highly dynamic nature of an MO's location, we do not require data to be stored on persistent and thus slow storage. In accordance with other research activities for location management [11], we assume the LMS to operate solely on main memory. Since MOs change their location frequently, lost state can be restored with the next position update of each MO.

Nevertheless, an efficient realization of the update and query operations requires fast access to the corresponding set of data entries. In particular, every region query requires efficient access to all entries with a symbolic identifier that is included in the requested location. Therefore, all spatial inclusion relations defined by the underlying location model have to be processed efficiently. For position updates and object queries the corresponding entries for a given *oid* must be found. Since each position update might trigger a change notification, their fast execution also includes finding all affected monitoring operations efficiently. This requires fast access to all object monitors with a given *oid* and to all region monitors with a location that includes either the old or the new position of the object (*but not both*).

To accelerate the required update and query operations, we introduce two index structures: an *object index* for speeding up the access by a given *oid* and the *spatial index* for efficient access by location. Both reference the corresponding data entries and registered monitors. To realize the object index, conventional indexing techniques such as hashing can be applied. Each object-based access then requires a single lookup of the given *oid* to retrieve the corresponding data entry or object monitors. Finding all affected region monitors and processing of region queries is more complex, because all inclusion relations have to be resolved. Realization of the spatial index therefore is our main focus and will be covered in detail in the next section.

3 Spatial Index

Efficient processing of region queries and region monitors requires a spatial index that is structured according to the underlying symbolic location model. Thus, each symbolic location is represented by one index node, and each inclusion relationship is represented by a link between the corresponding index nodes, linking each index node to its predecessors (*parents*) and successors (*children*). A conventional indexing technique, mapping symbolic location identifiers to corresponding index nodes, is used to allow efficient retrieval of single index nodes. Furthermore, each index node is associated with a mapping that enables the retrieval of all MOs that are currently located inside the region represented by that index node. The basic approach is to recursively traverse the spatial index starting from an initial index node and to consult all relevant mappings to determine MOs located inside the region represented by the initial index node. However, since these mappings must be adjusted on each position update, an approach aiming at a spatial index introduces a fundamental tradeoff between update and query costs. In the following, we present two basic schemes using different mapping implementations to minimize update and query costs, respectively. We then derive an adaptive approach that is able to balance the total costs depending on a given scenario.

3.1 Index Variant 1: Optimizing Updates

In the first approach each MO is referenced exactly once from within the spatial index. More precisely, it is referenced by the index leaf node that represents its current symbolic location. Hence, each index leaf node references MOs whose current symbolic location matches the one it represents. Accordingly, each region monitor is associated with those index leafs that represent a location contained inside the monitored region.

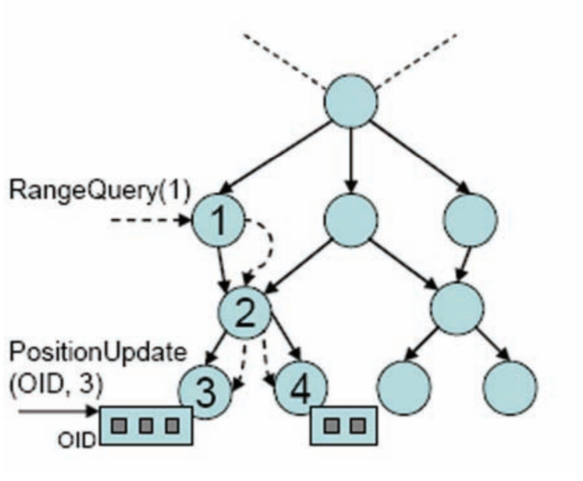


Fig. 2. Index variant 1

Since MOs are stored only at the leaf nodes of the spatial index, **position updates** require minimal processing overhead: First, the stale index entry of an affected MO is obtained by a lookup in the object index and deleted from both the object index and the corresponding node of the spatial index. The MO’s current position is in turn reflected by inserting the *oid* at the appropriate index leaf and an entry into the object index. While updating, all affected **region monitors** are found by comparing the monitors’ references in both leaf nodes. Monitors referenced only by the old (new) index node must be notified about the object leaving (entering) the monitored region. In contrast to the small update costs, **region queries** require notably higher processing cost. To determine all objects located inside the requested region, the spatial index must be traversed recursively by resolving the inclusion relation of the location model. Starting with the node that represents the requested area (cf. Fig. 2, node 1), MOs are collected from index leafs during recursive traversal of the index structure (cf. Fig. 2, nodes 2, 3, and 4) and merged into the final result.

3.2 Index Variant 2: Optimizing Queries

The processing of region queries can be optimized by shifting the required overhead for iterating through the index towards the update side. Thus, in the second variant, each index node directly references all MOs that are located inside the location represented by that index node. Hence, any *oid* is stored at each index node whose corresponding location contains the MO’s current location (cf. Fig. 3, the *oid* is stored at nodes 3, 2, 1, 5 and subsequent nodes on the paths to the root).

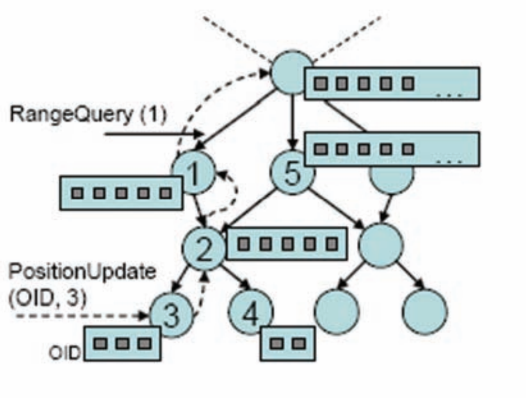


Fig. 3. Index variant 2

This variant of the spatial index eases the processing of **region queries** dramatically, because a single lookup of the node corresponding to the requested region immediately returns the complete result. However, the processing of **position updates** is more complex, because each update operation requires updating the object list of a number of index nodes accordingly. Based on the inclusion relationships, the index structure is traversed towards its root for both the previous and the new position of the MO, such that the object is deleted from the former and inserted at the new index nodes. Note that if an index node contains both, the former and the new position of an object (cf. Fig. 3, node 2), update propagation stops since the updated MO remains within all children of that node. **Region monitors** can be associated directly with the index node representing the monitored region. Thus, all relevant monitors that must be notified can be determined by merging the references of each updated index node during upward traversal of the index structure during a position update, while all unaffected region monitors are skipped during traversal when update propagation stops. Note that the movement of mobile objects is typically continuous so that two locations involved in a location update tend to be close to each other. That proximity is typically reflected by common ancestors in the containment hierarchy. Thus, there is a fairly large number of nodes that contain the former and the new position of an MO and hence remain unaffected by the update.

3.3 Variant 3: Balancing Costs

The two presented variants provide minimal processing costs for either position updates or region queries. In each case the other operation must resolve the inclusion relationships and thus becomes more costly. An alternative for spatial indexing is to distribute an index traversal among both operations in order to balance the total costs more evenly. We accomplish this in our third approach by varying the number of index nodes that directly reference mobile objects. More precisely, we restrict the propagation of position updates along the index structure by introducing a threshold, denoted by *th*. Only nodes whose *level* (cf. Sec. 2) is not greater than *th* will store and propagate any position update. Accordingly, the same set of index nodes will store references to region monitors. Any monitor for a location with a higher level will still be registered at all contained index leafs (as in variant 1). Note the important

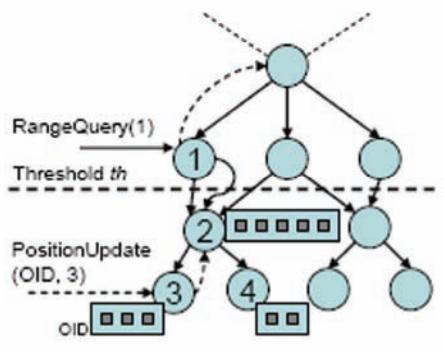


Fig. 4. Index variant 3

property that any node of the index is either affected by *all* update operations or does *not* maintain *any* object references at all. As a consequence, a region query that corresponds to a node whose level is not greater than *th* may directly return this node's list of objects without any further index traversals.

Introducing a threshold *th* accelerates the processing of **position updates** by decreasing the number of nodes that have to be traversed compared to variant 2. At the same time, the average amount of processing required for **region queries** is reduced in comparison to variant 1, because only region queries for locations whose level is greater than *th* require additional traversal until they reach nodes below *th*. Note that variant 3 is a generalization of the other variants where $th = 0$ and $th = height(\mathbf{L})$ realize variant 1 and 2, respectively. Hence, the threshold can be used to tune the overall processing costs by increasing either the performance of region queries (large values of *th*) or position updates (small values of th). Adapting *th* thus allows balancing the overhead between both operations for different load configurations.

4 Performance Evaluation

We now discuss the evaluation of the presented indexing variants. To verify the intended performance characteristics we measured the time required to perform single update and query operations in different scenarios. Subsequently, we determined the overall performance under load. The LMS was implemented in Java 1.4 and executed on a single node (Pentium 4, 2.5 GHz, 512 MB RAM). Update and query operations were issued using local communication. We used randomly generated location models of three different sizes to reflect different scenarios: *small location models* with height = 4 and 25×25 leaf locations (to represent e.g. small buildings with floors, rooms, etc.); *medium location models* with height = 7 and 50×50 leafs (e.g., a campus) and *large location models* with height = 10 and 100×100 leaf locations (e.g., city centers). The mobility of MOs was simulated using random, but continuous movement, so that objects do not "beam" to distant locations but change between neighboring locations. Region queries were generated by randomly choosing locations from the location model. Fig. 5-7 show the results of our evaluation for the stated parameters. Each measuring point represents the average over a total of 100 measurements (5 measurements for each of 20 randomly generated location models).

Fig. 5 depicts the average execution time of a single region query in relation to the number of managed MOs. The time consumption increases with the number of registered MOs, because more objects are located at the same location and thus processed and returned for each query. Furthermore, processing time clearly decreases with larger threshold values *th*. Index variant 2 (V2) yields best performance for all sizes of the location model. In

Fig. 6 the time required to process a position update is shown in relation to the number of managed MOs and region monitors. The execution time is primarily influenced by the average amount of region monitors to be processed per update operation, which increases with the overall number of registered monitors and smaller location models. But under all conditions, update costs decrease for smaller values of *th*, which determines the level up to which position updates are propagated. The experiments show how the costs of position update and region query processing oppose each other. Higher update costs result in lower region query costs and vice versa. This observation affirms the algorithm design.

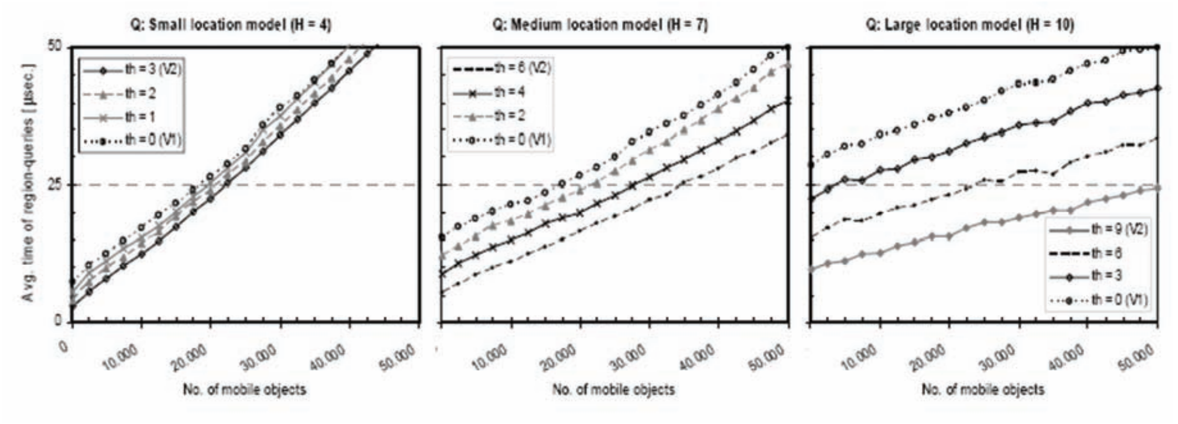


Fig. 5. Query costs for an increasing amount of mobile objects

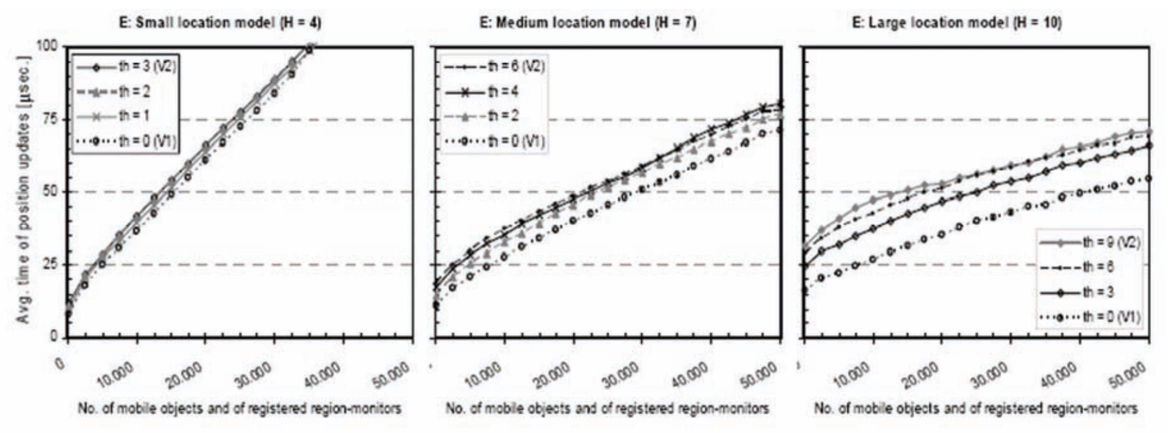


Fig. 6. Update costs for an increasing amount of mobile objects and region-monitors

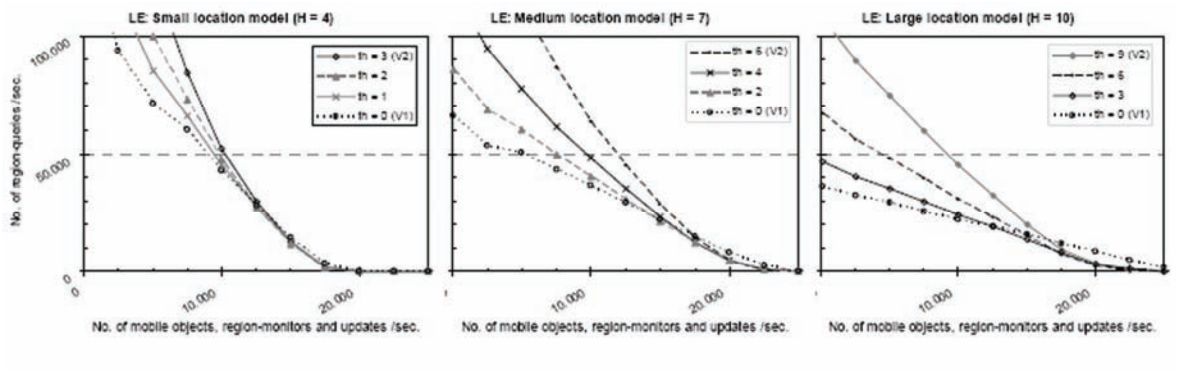


Fig. 7. Queries under load (with an increasing amount of mobile objects and region-monitors)

In order to asses the throughput of the LMS under load, we varied the load by increasing the number of MOs and update operations. Since we require high accuracy of position information, update operations are never dropped or delayed. As a result, the overall performance under load is given by the number of region queries that can be processed without delaying any update operation. Therefore, we prioritized the updates and measured the saturation of the system with region queries. Fig. 7 shows the resulting throughput in relation to the amount of registered region monitors and MOs, each MO issuing one position update per second. For all three scenarios, variant 2 (V2) achieves the best throughput for a relatively small number of objects – but also shows a rapid decline for larger values. The overall performance of variant 2 is mainly dominated by the high update costs that crush query processing for large numbers of position updates. On the other hand, variant 1 (V1) reveals a gentler slope. For a large number of MOs the low update costs pay off accordingly. Consequently, the LMS performs better under a heavy load of update operations and is able to process a larger amount of position updates under full utilization. Variant 3 succeeds in smoothing out the gap between the first two approaches. Nevertheless, this variant never shows best overall throughput for any threshold value, which is due to the required overhead for traversing the index structure for both operations. Yet, it allows balancing the costs between update and query costs to support a large range of scenarios.

5 Conclusion

Location management is a crucial task for location-based services. We have presented index variants to efficiently manage the position of mobile objects and support range and object queries based on symbolic coordinates. The indexes allow balancing the cost for update operations and region query processing. The requirements on the underlying location model are typically fulfilled by inclusion-based location models allowing a broad applicability of this approach. In the future, we will focus on automatic adaptation of the index with respect to current load characteristics. Furthermore, the efficient integration of purely symbolic location management into geometric location management will be investigated.

References

1. C. Becker, F. Dürr, "On Location Models for Ubiquitous Computing", Personal and Ubiquitous Computing, Vol. 9, No. 1, Jan. 2005, pp. 20-31
2. M. Bauer, L. Jendoubi, O. Siemoneit, "Smart Factory – Mobile Computing in Production Environments". Proc. Work. Appl. of Mobile Embedded Syst. (WAMES '04), USA, 2004
3. M. Bauer, K. Rothermel, "How to observe real-world events through a distributed world model", Proc. 10th Int'l Conf. Parallel and Distributed Systems (ICPADS '04), USA, 2004
4. B. Brumitt, S. Shafer, "Topological World Modeling Using Semantic Spaces", Proc. Workshop Location Modeling for Ubiquitous Computing, Atlanta, USA, Sep. 2001
5. T. Drosdol, T. Schwarz, M. Bauer, M. Großmann, N. Hönle, D. Nicklas, "Keeping Track of 'Flying Elephants': Challenges in Large-Scale Management of Complex Mobile Objects". Proc. 34th Ann. Conf. of the Gesellschaft für Informatik e.V.(GI); Ulm, Germany, 2004
6. V. Gaede, O. Guenther, "Multidimensional access methods", ACM Computing Surveys, Vol. 30, No. 2, pp. 170-231, 1998
7. A. Harter, A. Hopper, P. Steggles, A. Ward, P. Webster "The Anatomy of a Context-Aware Application", Proc Int'l Conf. Mobile Computing and Networking, Seattle, USA, 1999
8. J. Hightower, R. Want, G. Borriello, "Location systems for ubiquitous computing", IEEE Computer, Vol. 34, No. 8, pp. 57-66, 2001
9. H. Hu, D. Lee, "Semantic Location Modeling for Location Navigation in Mobile Environments" Proc. Int'l Conf. Mobile Data Management (MDM '04), Berkeley, USA, 2004
10. C. Jiang, P. Steenkiste, "A Hybrid Location Model with a Computable Location Identifier for Ubiquitous Computing", Proc. 4th Int'l Conf. Ubiquitous Computing. Göteborg, 2002
11. A. Leonhardi, U. Kubach, "An Architecture for a Distributed Universal Location Service", Proc. European Wireless Conference (1999), Munich, Germany, 1999
12. U. Leonhardt, "Supporting Location-Awareness in Open Distributed Systems", PhD-thesis, Imperial College of Science, Technology and Medicine, University of London, 1998

13. Z. Mao, C. Douligeris: “A Distributed Database Architecture for Global Roaming in Next-Generation Mobile Networks”, IEEE/ACM Trans. Netw., Vol. 12, No. 1, Feb. 2004
14. K. Rothermel, M. Bauer, C. Becker, „Digitale Weltmodelle – Grundlage kontextbezogener Systeme“. Total vernetzt – Szenarien einer informatisierten Welt. Xpert.press, Mai 2003
15. A. Roy, S. Bhaumik, A. Bhattacharya, K. Basu, D. Cook, S. Das, “Location Aware Resource Management in Smart Homes”, Proc. 1st IEEE Int’l Conf. Pervasive Computing and Communications (PerCom '03), Fort Worth, USA, Mar. 2003
16. H. Samet, “The Quadtree and Related Hierarchical Data Structures”, ACM Computing Surveys, Vol. 16, No. 2, pp. 187-260, 1984

Multimodal Interaction in Context-Adaptive Systems

Alexander Schneider, Andreas Lorenz, Andreas Zimmermann, Markus Eisenhauer

Fraunhofer Institute for Applied Information Technology
Schloss Birlinghoven
53754 Sankt Augustin, Germany
{Alexander.Schneider, Andreas.Lorenz, Andreas.Zimmermann, Markus.Eisenhauer}@fit.fraunhofer.de

Abstract. This paper reports on the architectural development process implemented in the MICA (Multimodal Interaction in Context-Adaptive systems) project. The goal is to make a major step towards natural human interaction with multimodal systems and the provision of pro-active assistance: we aim to do this by combining explicit and implicit interaction on different modalities reflected in a new layer based architecture for multimodal interaction.

1 Introduction

The project MICA (Multimodal Interaction in Context-Adaptive systems) aims at making a major step towards natural human interaction with multimodal systems: To realize this vision we aim to refer to and strengthen fundamental knowledge about natural interaction and information processing and develop applications that have the capability to understand the intentions of the user. Especially in applications for which extensive user training is not feasible, applications need to anticipate a mix of purposive behavior and uncertainty, to render information in a way that is compatible with the characteristics of the terminal, the situational context, the cognitive load, and the user's personal preferences.

We implemented a system to assist warehousemen in the picking process. The warehouse scenario poses a real challenge for multi-modal interaction. Because the workers are always using their hands, hands-free support is required. The environment is often very noisy and the light conditions might change as well. The interaction needs to use different modalities according to the needs of the current situation and task. Warehousemen often have to work under time pressure requiring a very responsive system. Intelligent fusion of fine grained tracking with RFID technology, in combination with pen or speech input and adaptable audio and graphical output will lead to natural blended interaction with the MICA-system.

The disadvantage of most approaches to multimodal interaction especially when gesture input is concerned is that the action to be performed by the users is often social obtrusive. Excessive gestures performed without a human counterpart look strange to a casual bystander. Our approach is to take advantage of natural behavior: It's a very natural thing, that if you are interested in something you approach the object to get a closer look and to investigate it in more depth. The movement of a worker is also used to determine if he is in need of help and in which modality pro-active assistance has to be delivered. MICA demonstrates the benefit of sensor fusion for a new interaction paradigm through movement of the users in the environment. The paradigm is portable to diverse application scenarios; possible application scenarios range from maintenance, warehouse, training activity support to IT-management support.

2 Context-Aware and Multimodal Systems

Context-aware computing is a mobile computing paradigm in which applications can discover and take advantage of contextual information (such as user location, time of day, nearby people and devices, and user activity). The relationship between systems and devices is constantly changing due to user mobility. In an early approach to model the context of a mobile user, Schilit et al. [15] present a facility for mobile application customization called "dynamic environment variables". Since it was proposed about a decade ago, many researchers have studied this topic and built several context-aware applications to demonstrate the usefulness of this new technology.

From a general point of view, Dey and Abowd [6] define 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". Focusing on what kind of information describes the subject's context, we condense the definitions of Schilit et al. [16] and the four context-dimensions of Gross and Specht [8]: User context, computing context, time context and environment. Additionally, when the context-information is recorded across a time span, we can obtain histories like interaction history, movement history, or event histories, which could also be useful for certain applications [3].

Currently, context-awareness lacks standards for exchanging information as well as a generic architecture and processing pipeline. Through the course of the MICA project, we developed an architecture hosting several components to achieve reusability, domain-independency and efficiency in the developing process. Most important we aimed at mapping the contextualization process to corresponding layer based architecture. The contextualization process is divided into four steps, each corresponding to one layer in the architecture:

Sensoring – recognizing the relevant parameters, Modeling – analyzing information and assigning meaning, Controlling – dialogue management, and Rendering – Generating appropriate output.

3 Related Work

So far, all multimodal services and demonstrators have been designed and built by trial and error. This was inevitable, because of the general lack of understanding of the basics of multimodal interaction. Components that relate to semantic interpretation are far from being standardized. As a result, services tend to come with idiosyncratic interaction styles, which must be learned from scratch by new users. This confronts prospective customers with long and steep learning curves, and it severely reduces the sustainability of systems and services. MICA will ameliorate this state-of-affairs significantly, by providing an open platform for application development, with the focus on smart sensor combinations for flexible natural interaction with human users solidly grounded in comprehensive models of human cognitive capabilities.

Many recent approaches place one information-mediating component in the heart of the architecture enabling agents to pass messages either to single agents or groups of agents: For example a blackboard in [2], the MCUBE in [9] or just a facilitator in [11] (based on the Open Agent Architecture [12]). These approaches have in common that all other components are placed unstructured around the mediator. The reason might be that those architectures are more driven by the need of inter-agent communication than by the requirements introduced by multi-modality. Larsen and Brønsted [10] described a promising extension of [2] overcoming this problem. We will later describe our approach of integrating brokering components into the architecture and the communication.

Recent architectures for multimodal applications (e.g. [4], [7]) have especially achieved success in support for sensing the multimodal user interaction, recognition of valuable input (like [5]) and disambiguation of recognition errors [13]. This research results in high-end abilities for recognition and synthesis in common modalities such as speech and handwriting.

In our work, we go beyond observing input from the different modalities to also integrate the recognition and interpretation of meaningful user-related and environmental parameters. From our point of view, the crucial tasks of building and maintaining a meaningful user-model as well as an environmental model are currently underrepresented. For structuring our components we are also aware of the three layers of the Seehiem model for UIMS design [14]:

- The top layer contains the input and presentation layer
- The middle layer ids the dialogue management
- The bottom layer is the application knowledge base

As visible for example in [17], this approach drives the middle-layer to host almost all components (here experts performing specific tasks). In combination with our own research work in context-aware systems [19] and the structure proposed in [1], we defined the MICA architecture, which will be described in detail in the next chapter.

4 The MICA Architecture

It is essential to improve our understanding of the interactive capabilities that are most important for an automated system to conduct a natural multimodal dialogue. We need to integrate a wide range of behavioral data from human users interacting with multimodal applications. Such analysis provides concrete hypotheses for directions to improve components of open multimodal context-adaptive systems, as well as the overall design and architecture of such systems.

The MICA architecture and its contextualization process is divided into four steps: Input, Modeling, Dialogue Management and Output, fulfilling each a specific role corresponding to a single layer in the architecture but residing and being implemented in a mix of centralized and distributed processing, with central processes and layers – Modeling and Dialogue Management at the Server side and Input and Output on the client side.

Figure 1 illustrates the architecture with respect to the distribution of the processes to server and devices. The main division is that In- and Output are localized on the client side and the main components of Modeling- and Dialogue-management are being implemented on the server side.

The Client is equipped with a monitoring component being able to react on local changes like a non functional sensor or actuator at the device or changing network conditions and a cache that serves as a temporary data repository. The architecture includes new important components like a task manager being responsible for the identification of finished and open tasks and illustrates the conception of repositories for Tasks, Dialogue Types, Content, User Preferences, Environmental data and Domain Knowledge.

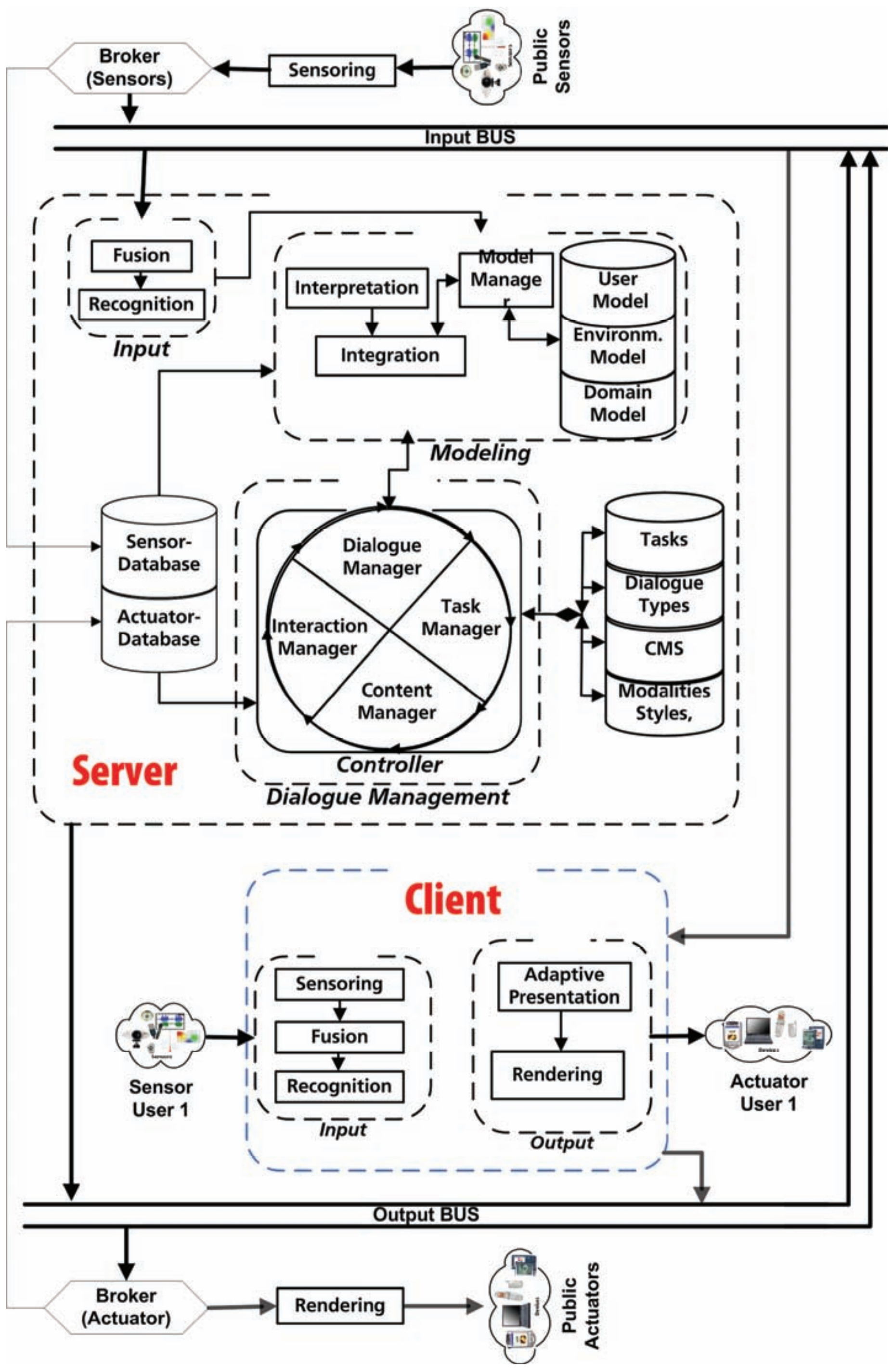


Fig. 1. MICA Architecture with centralized and distributed processing

4.1 Input – Processing Incoming Raw Data

The input layer consists of Sensoring and recognizing components. Sensors of a multimodal and context-sensitive system perform two tasks: recognition of the user's interaction with the system and sensing of user-related (e.g.

tracking data, user's heart beat) or environmental contextual parameters (e.g. room temperature, light conditions). An interactive multimodal implementation will use multiple input modes such as audio, speech, handwriting, keyboard, and other input modes. Additionally, several context-parameters need to be observed to enable the application to adapt its behavior and to be able to interpret the input of the user correctly. Therefore, a set of physical sensors observe variable parameters of a domain, measure their current values, and make the data stream available for recognition components. Examples for sensors are:

- Microphone, pen, pointer, and keyboard for multimodal interaction.
- Tracking-systems, (head-)orientation sensors, light and noise sensors, or clocks for contextual information.
- RFID-readers, scale that deliver other useful task related data.

The result of the Sensoring process is raw digital data not yet specific for any application. The data is forwarded to recognition components which capture natural input from the user and translates it into a form useful for later processing. Multimodal recognition extracts commands from speech or gestures and patterns of implicit feedback, e.g. the recognition of movement patterns. As a result, recognition delivers what input the user provided to the application, for example

- Converting spoken speech, key presses, or handwritten symbols into text.
- Converting pointing gestures into direction and angle, or button presses into (x,y)-positions on a two-dimensional surface.
- Converting different satellite-positions into GPS-coordinates.

Treated as a special enhancement of sensors, a fusion step was placed inbetween the Sensoring and recognition step. Regarding context parameters sensor fusion means aggregating the results of different sensors together to produce a reasonable approximation of the system state. This means that even if some sensors fail or give erroneous answers, the system will still be able to determine the current state. Different statistical methods might be suitable like taking the median instead of the average, or defining a critical value e.g. all values above the standard deviation are ignored. This method would avoid drastic measures being taken by the context system to correct what it considers to be temperature variations but are actually simply the result of sensor malfunctions. The result is robust system behavior.

Besides other sensor-related data, the sensor database contains grammars, vocabularies, templates and patterns necessary for a successful recognition process.

4.2 Modeling – Information Interpretation and Integration

On the modeling layer, interpretation and integration components analyze the data and put the information into relation with the information about the user and the context. The interpreting step answers the question, what the current value of a sensor means to a specific application. Each interpretation component identifies the "meaning" or "semantics" intended by the user and adds that to the recognition. The emerging knowledge is placed in different models to reflect aspects of the user and user-related data, as well as to describe the environmental context. Examples for interpretation are:

- Interpretation of recognized spoken text to derive its meaning (e.g. is it a command, and which command is it? - for example “display here”).
- Interrelation of positioning system data with the spatial model and user model to derive the absolute location (i.e. the room) of the specific user.
- Interpretation of the direction and angle of a pointing gesture to derive the targeted position, e.g. something left to the user.
- Interpretation of the user motion (e.g. is she moving slow or fast?).

The integration step combines output of several interpretation components. Intelligent modules aggregate information by interpreting the data with meta-information. In excess of multimodal fusion, integration puts the meaning of several interpreted information sources into relation, like

- Combining the positioning data from a user with the distance measures to get the distance and relative location of the user to other entities (e.g. “in front of”, “far away from”, “in eyesight of”).
- By integration of the user’s position, orientation, and her spoken command (“display here”), and the direction and angle of her pointing gesture towards the left wall, the application determines the user’s demand to use this wall as display.

The integration of the incoming information with existing knowledge about the user and her context and other multimodal input refines the knowledge-base and can be used for disambiguation. The environment and user model can be used to resolve ambiguities by considering the contextual parameters defining e.g. the co-occurring situation and task. The different modes of a multimodal system are not simple analogues of one another and don’t involve redundant but complementary information and can be used for mutual disambiguation [13]. The ambiguity in one modality can be resolved by the input of the other.

4.3 Dialogue Management – Controlling Content Selection and Presentation

The dialogue management layer plans, assembles and refines sequences of commands to control the behavior of the system. At this stage, the decisions to be taken by the system include issues like privacy or pedagogical objectives, and handle the dialogue with a set of users in cases where different users are to be reached by the same or different output channels.

In this layer, the content manager selects the appropriate content from the content management system and the interaction manager selects the appropriate modality. The user model particularly determines the selection of content to support the user by providing the right information. The information about the user's environment will be taken into account as well in order to adapt the presentation to the right connotation and intonation.

In order to distinguish between the different parts of dialogue management and to define the processes more precisely, we divided the dialogue management package into three parts, which actually interplay with each other in a parallel process. A dialogue manager, an interaction manager and a content manager jointly co-operate, in order to take appropriate actions, that base on derived knowledge. The dialogue manager as such is responsible for the coordination of the content manager whereas the interaction manager queries them and combines their suggestions.

The content manager is responsible for managing the information-flow of the content-repository. From the research work in context-aware systems of Zimmermann and Lorenz ([18], [19]) we concluded that it is not enough just to supply content without the consideration of the recipient, her current task, and situation (the time, the location, the particular technical environment, and even biological data). Context-sensitive content and information processing is an asset for the generation of added value.

The interaction manager creates and traces certain interaction strategies (e.g. in case of an emergency, it presents information concisely, or uses examples extensively in a learning situation). Additionally, it adapts to certain conditions in the environment or in the user context (e.g. to switch from audio to visual output, if background noise rises). The adaptive or strategic methods are implemented domain-independently, although the expressivity of a modality might be domain-dependent.

4.4 Output – Adapt and Render the Presentation

The output layer handles the connection back to the users. The presentation of the content is adapted to the different technical capabilities of the targeted devices (e.g. screen size). The processed content is then rendered to the appropriate output channel provided by the device. Thus, decisions that have been taken by the content and interaction manager in the preceding layer result in real world actions.

For configuration and properties of the output components we also added an actuator database with the required information of connected actuators.

5 Communication

As the basis for the communication between distributed components, we decided to rely on a standard-based instant messaging server based on the XMPP protocol also known as Jabber (http://www.jabber.org) and on the Attribute-Relation-File-Format (ARFF, refer to http://www.cs.waikato.ac.nz/~ml/weka/arff.html) as the message format. Jabber is a set of streaming XML protocols and technologies that enable any two entities on the Internet to exchange messages, presence, and other structured information in close to real time. Besides being standardized, Jabber also offers other benefits like open source servers existing on most common platforms and libraries and available in most common languages.

The Jabber-server acts as a message distribution center which handles authentication of users, multi-user-chat room management, message distribution and caching. Each Jabber-client has a personal jabber-address where one can send messages directly to other users. Jabber-clients can subscribe to multi-user-chat rooms where everyone receives a message sent to this room. To enable maximum flexibility there is no one-to-one mapping between the client program on a device and a Jabber-client. A client program exists of many components which are reusable and therefore we decided that each component is a Jabber-client.

These multi-user-chat rooms are used for distributing messages that are potentially valid for more than one receiver e.g. location-sensor-data. The type of messages exchanged in a certain room is identified by the room-name. Each component interested in that kind of information has to subscribe to that room. For example we defined a room named “roomtagidentification” where messages can be posted for identifying a RFID-tag. If a client-component wants to know which RFID-tag was placed on a RFID-reader it will send a message to that room with the RFID-tag-ID. Other components subscribed to that room will receive this message. In our system there is only one component responsible for that so this component will try to identify the tag and send a response back. This response is not sent to the room but directly to the sender because the reply is not interesting to other components subscribed to that room. This allows for efficient messages distribution between all components of the system and is flexible with only minor complexity even in large systems.

6 Conclusion

Our approach in MICA was to enhance current multimodal applications to consider more than speech and gestures and to define an architecture for flexible, adaptable and pro-active systems. From our research in context-aware system development we concluded to integrate other non-intrusive modalities to support natural blended interaction. In particular, the movements of the user observed by a tracking system provide valuable implicit and explicit feedback to the system. The aspects to be taken into account were reflected in the description of the system architecture for the MICA project, which is being implemented and will be evaluated for a warehouse application.

References

1. Barnett, J., Bodell M., Raggett D. and Wahbe A.: W3C Multimodal Architecture and Interfaces, http://www.w3.org/TR/mmi-arch/, 2006
2. Bondesen, P., Poulsen, P., Lykkegaard, M.: SmartPool – A multi modal pool training system. Master thesis, Aalborg University, 1999
3. Chen, G. and Kotz, D.: A survey of context-aware mobile computing research. Technical Report TR2000-381, Computer Science Department, Dartmouth College, Hanover, New Hampshire, 2000
4. Cohen, P., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., and Clow, J. Quickste: Multimodal interaction for distributed applications. Proceedings of the Fifth ACM International Multimedia Conference, New York, NY: ACM Press, 1997, 31-40
5. Conçales, L.M.G., Grupen, R.A., and Oliveira, A.A.F.: A Control Architecture for Multi-modal Sensory Integration. Proc. of XI Int. Conference on Computer Graphics and Image Processing (SIBGRAPI'98). Rio de Janeiro, Brazil, 1998
6. Dey, A.K., and Abowd, G.D.: Towards a better understanding of context and context-awareness, Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology, 1999
7. Eccher, C., Eccer, L., Falavigna, D., Nardelli, L., Orlandi, M., and Sboner, A.: On the usage of automatic voice recognition in a web based medical application. Proceedings of ICASSP 2003, Hong Kong, China, 2003
8. Gross, T., Specht, M.: Awareness in context-aware information systems. In H. Oberquelle, R. Oppermann, and J. Krause, Editors, Mensch und Computer - 1. Fachuebergreifende Konferenz, pages 173–182, Bad Honnef, Germany, Teubner-Verlag, 2001
9. Johnston, M., Bangalore, S., Vasireddy, G., Stent, A., Ehlen, P., Walker, M., Whittaker, S., and Maloor, P.: MATCH: An Architecture for Multimodal Dialogue Systems, Proceedings of the Annual Meeting of the Association for Computational Linguistics Philadelphia, USA, pp. 376-383, 2002

10. Larsen, L.B., and Brøndsted, T.: A Multi Modal Pool Trainer. Proc. of the International Workshop on Information Presentation and Natural Multimodal Dialogue - IPNMD-2001, Verona Italy, pp. 107-111, 2001
11. Lemon, O., Bracy, A., Gruenstein, A., and Peters, S.: The WITAS Mulit-Modal Dialogue System I. Proceedings of Eurospeech 2001, 2001
12. Martin, D., Granlung, G., Kuchcinski, K., Sandewall, E., Nordberg, K., Skarmann, E., and Wiklund, J.: The Open Agent Architecture: a framework for building distributed software systems. Applied Artificial Intelligence, 13(1-2), 1999
13. Oviatt, S.: Mutual Disambiguation of Recognition Errors in a Multimodal Architecture. Proceedings of CHI 1999, Pittsburgh, USA, pp. 576-583, 1999
14. Pfaff, G.: User Interface Management Systems, Berlin: Springer Verlag, 1985
15. Schilit, B.N., Theimer, M.M. and Welch, B.B.: Customizing mobile applications. In Proceedings of USENIX Symposium on Mobile and Location-independent Computing, pp. 129–138, Cambridge, MA, US, 1993
16. Schilit, B.N., Adams, N.I., and Want, R.: Context-Aware Computing Applications. Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, pp. 85–90, 1994
17. Wilson, M.D., Sedlock, D., Binot, J.-L., and Falzon, P.: An Architecture for Multimodal Dialogue. In: Taylor, M. M., Neel, F., and Bouwhuis, D. G., Editors, Proceedings of the second Vencona Workshop on Multi-Modal Dialogue, 1991
18. Zimmermann, A. and Lorenz, A.: Creating Audio-Augmented Environments. Journal of Pervasive Computing and Communication, 1(1): 31-42, 2005
19. Zimmermann, A., Lorenz, A., and Specht, M. (2005). Personalization and context management. User Modeling and User Adaptive Interaction, 15:275–302, 2005

The Quick Step to Foxtrot

Tino Löffler, Stephan Sigg, Sandra Haseloff, and Klaus David

Chair of Communication Technology,
Department of Electrical Engineering / Computer Science,
University of Kassel,
D-34121 Kassel, Germany;
phone: +49 561-804-6446, fax: +49 561-804-6360
comtec@uni-kassel.de

Abstract. We propose Foxtrot, a highly flexible architecture for context aware systems that adapts to the versatile requirements of ubiquitous computing environments. To further enhance the gains of the architecture we introduce a context prediction paradigm that incorporates the possibility to increase the accuracy of proactive context aware applications. To utilise this potential we propose a novel context prediction algorithm. Additionally we introduce a methodology for designing context aware systems. Our development guidelines and the corresponding software equip the system designer with powerful analysis tools that support the creation of systems with greatly improved reliability.

1 Introduction

Context awareness is the ability of an application or service to sense the context in which it is currently executed and to react accordingly. Context awareness plays a dominant role in modern software systems and is about to increase its influence through novel user adaptive applications. Consider for example the following scenario.

When Milinda puts on her context–aware wrist watch in the morning, it starts scanning the environment for external sensors or devices. In the kitchen it senses the availability of the CD-player as well as the humidity, temperature and light sensors attached to the kitchen window and obtains data from them. When Melinda positively answers the prompt for turning on the CD-player, the watch initialises to play stimulating music since it is to be a bright day. After breakfast, on the stairs under way to her car Melinda meets her new neighbour Tom and stops for a chat with him. Due to the context time line the watch has observed so far it senses that Melinda is heading to work. Consequently it reminds her at 8:34 that she will be late if she kept on talking.

In this short example several interesting features of upcoming context aware applications neatly stick together. These are the sensing and utilising of remote devices and sensors, the context interpretation and the context prediction.

We will in this paper propose an architecture for context awareness that is highly flexible in supporting several concepts for context aware computing and may also be distributed among several mobile and stationary computing devices.

We further give a short insight into our approach to the research topics context prediction and context aware system development. Both are considered to significantly improve the scope of context aware applications beyond their current horizon. This work is organised as follows. In Sect. 2 we will give a brief overview of the related work, in Sect. 3 we present our architecture for context awareness while in Sect. 4 a novel methodology to design context aware systems and to distinguish the vital sensors relevant to a specific context is presented. Section 5 proposes our approach to context prediction and introduces the prediction module in detail. Section 6 summarises our results.

2 Related Work

Since context awareness has been recognised as a promising research field, various architectures for context aware systems have been proposed. The Context Toolkit [1] provides support for context aware systems. Applications can subscribe to predefined context sources, which can be plain sources, aggregators and interpreters. With our module based approach Foxtrot (Framework fOr ConteXT awaRe cOmpuTing) we adapt the general idea of this concept and further add the ability to distribute all modules to various mobile computing devices in a ubiquitous computing environment. Foxtrot further supports the addition, removal or update of modules at runtime. This is accomplished by adapting our architecture to the Framework for Applications in Mobile Environment (FAME^2)[2]. FAME^2 enables applications and services to be distributed between various devices and to be added, updated or removed at runtime. Another architecture for context aware applications is proposed by Henricksen [3] who concentrates on context modelling and proposes a layered architecture. Although similar to our approach, Henricksen does not reach the same flexibility in ubiquitous systems. The Solar platform [4] represents a context aware architecture which employs a graph-based abstraction for context aggregation and dissemination. Our approach is, while not graph-based, similar to that of the Solar platform but additionally considers a new context prediction layer on top of the context acquisition step and adds the aforementioned runtime flexibility.

This is also the main difference to architectures proposed by other authors. The authors of [5–8] have proposed an architecture for recognising context from multiple sensors in the TEA and Smart-Its projects. While [9] also considers a context prediction layer, the authors propose to include it at a later stage in the context processing procedure. In Section 5 we will argue why it is beneficial to apply the context prediction in an earlier stage.

3 Architecture for Context Awareness

We present Foxtrot, an architecture for context awareness that is inspired by our main research interests and that will serve as the basis for current and future research projects related to context awareness. Since we want to utilise the architecture also for upcoming research projects we focus on a modular design and extensibility. Parts of the architecture are subject to constant changes. This means that the possibility to easily exchange modules, if possible even at runtime, is critical.

Furthermore the architecture is supposed to be executed in a distributed environment, so the deployment of parts of the architecture, discovery and network communication are also required. To meet these requirements, we adopted a service oriented architecture approach and implemented all functionalities as interacting, loosely coupled services. These services were built using the FAME^2 [2] framework. FAME^2 enables adding, removing and updating of services at runtime, which eases the maintenance of the system. Additionally it allows us to integrate several service discovery technologies and communication protocols to access arbitrary services from any other device reachable.

3.1 Logical Segmentation

We divide our architecture into the four functional entities *context acquisition, context prediction / preparation, context processing* and *context usage* as illustrated in Fig. 1. Every functionality incorporates modules that are explained in the following.

Context Acquisition The context acquisition part is responsible for acquiring context. It comprises context broker modules which read from attached sensors and convert the data into our context representation.

We refer to the output of any sensor as raw sensor data since it most probably needs further interpretation. Different manufacturers produce sensors with varying output even if the sensors are of the same class. This is because of possibly different encodings of the sensed information or due to a different representation or accuracy. After being processed by a context broker raw sensor data has become low-level context information. The low-level context information of two arbitrary sensors of the same class measured at the same time in the same place is identical with the exception of a possibly differing measurement accuracy, provided that both sensors are in good order.

Output values of a sensor are firstly transformed by the context broker from raw sensor data to low-level context information. The sequence of low-level context information attained from a sensor is stored to the output buffer of the context broker.

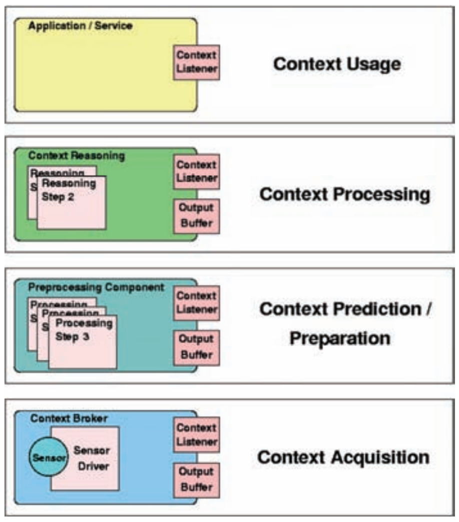


Fig. 1. Architecture for context awareness

Context prediction / preparation Usually the low-level contexts processed by the context acquisition part of Foxtrot have to be further prepared for the following context processing step. Typical preprocessing modules in this step are the context prediction, the smoothening of values, the calculation of a mean value or other statistical methods. We refer to these operations as preprocessing steps. A preprocessing component may string various preprocessing steps and apply them to the low-level contexts in any order. The output of a preprocessing component is again low-level context information.

Context Processing The interpretation and aggregation of low-level context to high-level context information is done in the context processing part by a context reasoner. A designer of context aware applications may only with great effort make a funded decision based on low-level context information. He expects the low-level contexts to be aggregated to high-level context information that yields more meaningful information. A typical high-level context is for example *in a meeting* or *indoors*.

The context processing part of Foxtrot basically consists of one or more context reasoning components. Reasoning components may be stringed or may operate in parallel to other context reasoning components. It is therefore possible to directly compare the accuracy of different context reasoning components in real life applications or in simulations. We have currently implemented a reasoning module based on the RuleML language [10]. Future projects may reveal the need for a reasoning module that is not based on fixed rules.

Context Usage The high-level context obtained after the context processing step is transformed into an event and delivered to an event handler. Context aware applications can subscribe to the event handler and will be informed if a specific context occurs.

3.2 Support of Ubiquitous Computing

Since we believe that ubiquitous computing is one of the key application fields of any architecture for context awareness, we require Foxtrot to be highly distributable between various computational devices. Every module of Foxtrot can be distributed to various computing devices. It is therefore possible that the modules of a functional entity are spread among various computing devices. Furthermore every module may be available several times in the computing environment.

In a fluctuating ubiquitous computing environment a module may unexpectedly disappear because the computing device has moved out of reach. We expect the disappearance and reappearance of devices to be a common phenomenon in a flexible ubiquitous computing system.

To serve these demands we implemented our architecture as a FAME^2 compliant middleware [2]. This enables us to integrate several discovery technologies and communication protocols to access arbitrary modules from any other device reachable. Each module in Foxtrot is implemented to represent a service in FAME^2. Services in any FAME^2 compliant middleware may appear and disappear or may be transferred from one device to another without the necessity to configure the device or the application itself.

Since all modules of one functional entity may be composed in virtually any order, the application designer might construct any desired application flow by choosing from a given set of modules. It is even possible that several different application flows are executed at the same time by the same modules.

4 Context-aware System Development

In Section 1 we already stated that context-aware systems are able to recognise or forecast context information. Based on this information they are supposed to adapt to the context automatically and to provide useful services and information to the user.

Context information can have different levels of abstraction. These levels range from measurable contexts like temperature to abstract, not measurable situations like meeting. Contexts that are not measurable must be inferred from known context information. To stick with our example: the context meeting could probably be inferred from the time, the location, nearby people and sound patterns. These dependencies between measurable and not measurable contexts are often not obvious. For example, we have investigated the possibility to infer the location (outdoor/indoor) from temperature and humidity.

This leads to the conclusion that the selection of sensors and the knowledge about dependencies between contexts predefines the reachable reliability of our context detection.

Henricksen [3] proposes a situation abstraction supported by a context modelling language. The context to be detected is modelled from parts whose state is measurable. This approach works well if dependencies between sensor data and context are clear and sensor selection is predefined.

Clarkson et al.[11] use an interesting method to determine Hidden Markov Models (HMMs) for their context detection. In their approach the user labels context states. The data is examined afterwards and the best performing HMMs are chosen for the detection system.

Our proposal of a methodology for designing context aware systems shares some basic ideas with the latter approach.

4.1 Methodology

Our aim is to design a methodology and a framework for the development of context-aware systems that helps programmers find dependencies between contexts and choose the right sensors for their system. We try to exploit experimental development procedures and data mining techniques to discover dependencies between contexts.

Previous to the development of the context-aware system we propose a three step experimental procedure to determine which sensors are appropriate and how the context can be inferred from their data.

Data acquisition In the first step we define the contexts that are to be detected and determine sensors that can be utilised depending on e.g. price, size and availability. These sensors are deployed depending on context and sensor type. The measured data is stored in a database. The occurrence of a predefined context is manually marked by the user.

Data interpretation In the second step we examine the dependencies between the sensor data and the manually marked contexts with data-mining techniques.

Inference logic design Based on the results of the data interpretation step the third step in the experimental procedure is to design the inference logic depending on the applied inference mechanism, i.e. defining rules or building a Bayesian network.

4.2 Framework

Beside the procedure and the guidance for the development of context-aware systems we plan to support developers with a software framework based on the introduced context-aware architecture. It will consist of tools supporting an easy integration of new sensors into the architecture, logging to miscellaneous data bases and analysing the data with various data-mining algorithms.

4.3 Benefits

We argue that context inference and hence context-aware systems can be made more reliable if they are created following our approach.

The selection of sensors can be tailored to the context and the inference mechanism are backed by statistical data, whereby context detection errors are minimised. Deploying only sensors needed for context detection improves the system performance, because only needed sensor data is analysed.

5 Context Prediction Based on Low–Level Contexts

As already mentioned in section 3.1 we propose to include a context prediction module as one preprocessing component in the context preparation layer. Since context prediction in the literature is typically executed on high-level context information, we will in this section discuss the benefits and drawbacks of several context prediction schemes.

5.1 Context Prediction Schemes

In the literature context prediction is usually based on high-level context information (see for instance [12–16, 9]). This approach is appealing as long as the number of high-level contexts is low. Compared to the high number of combinations of low-level contexts from all available sensors the set of high-level contexts in typical examples is considerably small.

However, the prediction based on high-level context information has vital restrictions due to a reduced knowledge about the context itself.

Reduction of Information. Some of the information contained in a low-level context is lost when transformed to a high-level context since the transformation function is typically not reversible. If the sampled low-level context information suffices to conclude a high-level context we can obtain this context at any time in the future provided that the low-level context information is still available. Once we abstract from low-level context information to high-level context information we cannot unambiguously obtain the low-level contexts we abstracted from. A time series of observed contexts consists of several samples from various sensors. The associated low-level contexts may indicate some general trend. However, the high-level contexts may mask this trend to some extend due to the higher level of abstraction.

Reduction of Operators. The only mathematical operator applicable to high-level contexts is typically the operator '='. All prediction methods that require other operators are not or only with additional computational overhead applicable to high-level context information. Popular context prediction methods therefore implicitly support non–numerical contexts but do not profit from contexts that provide additional information. The number of prediction methods suitable to low-level contexts is therefore larger than the number of prediction methods appropriate for prediction on high-level context information.

Reduction of Certainty. The prediction accuracy may be decreased when prediction is based on high-level context information. This is due to the different sequence in which the context prediction and context interpretation steps are applied. Let $P_{\text{acquisition}}$ be the probability that no error occurred in the context acquisition step. Furthermore $P_{\text{interpretation}}$ and $P_{\text{prediction}}$ are the probabilities that no error occurs in the context interpretation and the context prediction step respectively. We write $P_{\text{prediction}}(i)$ if we want to address the probability that the context element at time t_{0+i} is predicted correctly. Assume that the prediction method bases its decision on a context history of t time series elements and predicts l future contexts. In the case of low-level context prediction each of the time series elements is composed of m low-level contexts. For high-level context prediction each time series element is represented by one high-level context. If the prediction is based on the low-level contexts, the probability that the predicted context element at time t_{0+i} is correct is

$$P_{low-level}(i) = P_{acquisition}^{m \cdot t} P_{prediction}^{m}(i) P_{\text{int}erpretation} \quad . \tag{1}$$

If the prediction is based on high-level contexts the probability, that the predicted context element is correct is

$$P_{high-level}(i) = P_{acquisition}^{m \cdot t} P_{\text{int}erpretation}^{t} P_{prediction}(i) \quad . \tag{2}$$

Comparing these results we obtain

$$\frac{P_{low-level}(i)}{P_{high-level}(i)} = P_{prediction}^{m-1}(i) \cdot P_{\text{int}erpretation}^{1-t} \quad . \tag{3}$$

Provided $P_{\text{interpretation}} = P_{\text{prediction}}$, prediction on low-level contexts leads to the higher accuracy compared to prediction based on high-level context information if the number of considered context elements from the context history is less than the number of low-level contexts in one time series element. Otherwise the ratio of $P_{\text{interpretation}}$ to $P_{\text{prediction}}$ further influences the prediction accuracy of low-level and high-level context prediction.

Reduction of the Long Term Accuracy. Provided that a sensor is in good order, the low-level context information reflects the actual situation with respect to the measurement inaccuracy. We therefore argue that low-level contexts are of higher credibility than high-level contexts. By interpreting low-level context information to high-level context information we take a guess based on the information that is available in the current situation. Since the context of the user is also dependent on information that is not measurable by sensors [17], we cannot exclude the possibility to misjudge the current context.

A prediction based on high-level context information might therefore be based on erroneous information. This does not only affect the instantaneous prediction accuracy but may also affect the long term prediction accuracy.

5.2 Time Series Prediction

For the above stated reasons, we base context prediction on low–level context information. In order to utilise the additional information on the context elements we use a time series prediction method that implements an alignment search method. The observed context time series is aligned with every time series in a data base of typical context time series. By finding the optimal local alignment we also get the most probable predicted time series. For a decent study on alignment methods we refer to [18].

6 Conclusion

With Foxtrot we have proposed an architecture for context awareness that incorporates a high flexibility due to its modular structure. Because of its service oriented approach it is especially suited for highly flexible ubiquitous computing environments. Novel context aware concepts are easily adapted by Foxtrot even at runtime since the architecture is based on FAME^2. In our current research projects we study the impact of low-level prediction methods as well as context modelling techniques. We have argued that low-level context prediction has significant benefits to high-level prediction methods. Especially the prediction accuracy may be improved, while also novel prediction methods may be applied. We proposed a novel context prediction method that utilises the additional knowledge available for low-level contexts. To further guide the application designer, our proposed methodology for designing context aware applications provides powerful tools aiding the development process.

References

1. Dey, A.K.: Providing architectural support for building context-aware applications. PhD thesis (2000) Director-Gregory D. Abowd
2. Wüst, B., Drögehorn, O., David, K.: Framework for platforms in ubiquitous computing systems. In: Proceedings of 16th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC). (2005)
3. Henricksen, K., Indulska, J.: A software engineering framework for context-aware pervasive computing. In: PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04), Washington, DC, USA, IEEE Computer Society (2004) 77
4. Chen, G., Kotz, D.: Context aggregation and dissemination in ubiquitous computing systems. In: WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications, Washington, DC, USA, IEEE Computer Society (2002) 105
5. van Laerhoven, K., Lowette, S.: Real–time analysis of data from many sensors with neural networks. In: Proceedings of the fourth international Symposium on Wearable Computers (ISWC). (2001)
6. Schmidt, A.: Ubiquitous Computing – Computing in Context. PhD thesis, Lancaster University (2002)
7. Schmidt, A., Beigl, M.: There is more to context than location: Environment sensing technologies for adaptive mobile user interfaces. In: Workshop on Interactive Applications of Mobile Computing (IMC'98). (1998)
8. Schmidt, A., van Laerhoven, K.: How to build smart appliances. In: IEEE Personal Communications. (2001) 66–71
9. Mayrhofer, R.M., Radi, H., Ferscha, A.: Recognizing and predicting context by learning from user behavior. In: The International Conference On Advances in Mobile Multimedia (MoMM2003). Volume 171. (2003) 25–35
10. http://www.ruleml.org: (2005)
11. Clarkson, B., Pentland, A., Mase, K.: Recognizing user context via wearable sensors. In: ISWC '00: Proceedings of the 4th IEEE International Symposium on Wearable Computers, Washington, DC, USA, IEEE Computer Society (2000) 69
12. Laasonen, K., Raento, M., Toivonen, H.: Adaptive on–device location recognition. Number 3001 in LNCS (2004) 287–304
13. Ashbrook, D., Starner, T.: Learning significant locations and predicting user movement with gps. (2002)
14. Davison, B.D., Hirsh, H.: Predicting sequences of user actions. In: AAAI/ICML Workshop on Predicting the Future: AI Approaches to Time–Series Analysis. (1998)
15. Mayrhofer, R.M.: An Architecture for Context Prediction. PhD thesis, Johannes Kepler University of Linz, Altenbergstrasse 69, 4040 Linz, Austria (2004)
16. Nurmi, P., Martin, M., Flanagan, J.A.: Enabling proactiveness through context prediction. In: CAPS 2005, Workshop on Context Awareness for Proactive Systems. (2005)

17. Barkhuus, L.: How to define the communication situation: Context measures in present mobile telephony. In: Context, Stanford, CA, Springer (2003)
18. Boeckenhauer, H.J., Bongartz, D.: Algorithmische Grundlagen der Bioinformatik. Teubner (2003)

Towards an Open Context Infrastructure

Manfred Wojciechowski[1], Jinhua Xiong[2]

[1] Fraunhofer Institute for Software- and Systems Engineering (ISST)
Emil-Figge-Str. 91, 44227 Dortmund, Germany
wojciechowski@do.isst.fraunhofer.de
[2] Institute of Computing Technology, Chinese Academy of Sciences,
100080, Beijing, China
xjh@ict.ac.cn

Abstract. This paper presents an approach towards an open context infrastructure that allows to sharing resources between developers and applications, thus reducing development costs and enabling business models for specialized context service providers. The main focus of this paper is to identify aspects and challenges that emerge from the openness and application independence of such an infrastructure. In an attempt to address and meet some of these challenges, the aspect of modeling contexts is examined in detail within this paper. The result is the definition of application specific and infrastructure context models and the description how these can be applied.

1 Introduction

There are a number of research projects concerned with the development of consistent, reliable and secure context frameworks that facilitate the development of context-aware applications [1], [2], [3], [4]. Suitable architectures, tools and models have been developed that can be utilized by applications operating on context information. Mechanisms for the management of context sensors, for the refinement of context information and for the definition of context models can be used to reduce the workload of developers in terms of the integration of context sensors and the usage of context information. Nevertheless, the developer still has to deal with the identification and integration of suitable context sensors as well as with the selection and refinement of suitable sensor data.

An important aspect concerning the development of context-aware applications for actual use is the cost of the initial set-up of such context environments. Using existing approaches each context-aware application relies on its own specific context environment, which is not very cost-effective. The development of an application-independent and open context infrastructure as proposed in this paper is considered to be a solution to this problem. The openness of such an infrastructure allows the ad-hoc integration of existing context information services that can be supplied by different service providers. Having an application-independent infrastructure allows context providers to integrate their context services without explicit knowledge of actual application-specific requirements. Application developers, on the other hand, are then able to make use of context information as provided by different, unspecified context providers. This allows for new business models for context service providers and a potential growth of service offers within this infrastructure. Such a context infrastructure is also an important aspect of information-logistical solutions, which is the basis of our context approach.

In the paper we identify the key problems to be solved in order to build an open context infrastructure that goes beyond existing frameworks. One important aspect is context modeling, which has a different focus for context service providers and for application developers.

In the first section the related approaches are examined. Then the basic requirements for an open context infrastructure as well as further aspects of context infrastructures that are beyond the scope of this paper are identified. The paper then focuses on the aspect of context modeling. An analysis of the different requirements and the resulting model elements is given. Finally we describe how the different models can be used within the context infrastructure and give a short conclusion.

2 Related Work

There are different approaches to provide context frameworks that can be used to make applications context-aware. They allow a separation of the application logic from the logic of context information detection and refinement. A prominent example for such a framework is the Context Toolkit [5]. State of the art frameworks describe a layered architecture that defines different activities in the process of context detection and provision, e.g. see [7]. In [8] another layered infrastructure is described and a five-layer abstraction architecture for structuring context-aware applications is presented. Other layered approaches can be found in [5], [9], [13]. These approaches give solutions for the general problem of how to make context information usable for context-aware applications and how to separate that from the core application logic. A layered architecture is a suitable abstraction mechanism that also has to be considered for an infrastructure approach. Another approach for modeling context can be found in [10].

Another aspect relevant for the context infrastructure is the distribution of context information and context providers. The approach in [11] comes close to our intention. In that paper the NEXUS platform is presented that allows to share the effort of setting up a global and detailed context model between different providers by federating their partial models. The NEXUS architecture defines three tiers that distinguish between usage of context information, provision of services and a mediating federation tier. One limitation of that approach is the common context model that is used for context service mediation and application-specific context operations. This strictly binds the applications to the common context model which can be a limitation since application-specific context models can be different in their semantics and scope depending on application domains and its intended use. Another interesting approach can be found in [12]. A context middleware framework is described allowing ad hoc discovery and integration of context sensors. A major focus of it is the selection of appropriate sensors regarding QoS parameters. This framework also is based on a common context model for any context aware applications.

Other related work that is not directly associated to context-awareness can be found in the field of open infrastructures that provide information usable as context. Any content can be used as context information if it is structured, thus machine readable and if it is related to a context model which defines its structure, format and semantics. Therefore, the Internet including the many Web sites and Web services is a potential pool of context information. Other relevant and interesting platforms and infrastructures include geo infrastructures (e.g. OpenGIS [14]), sensor networks, location-based service infrastructures from telecommunication providers and the EPC infrastructure [15].

3 Openness of Context Infrastructures

The design of an open context infrastructure has to consider some basic requirements that will be discussed in the following. Most of these requirements are not solely valid for open context infrastructure approaches, however, the aspect of openness is putting more emphasize on them. From a functional point of view a context infrastructure has to support context providers and context-aware applications by different means. As an overall requirement an open context infrastructure has to provide a variety of context services from different service providers in order to support multiple context-aware applications.

Context providers must be able to integrate their context services in a simple way without prior knowledge concerning context-aware applications that use context information or application-specific context models that have to be supported. Providers must be able to describe the context information offered by their services, which are then used for dynamic context service discovery. Also, the implementation of a context service must allow ad-hoc access to its functionalities by different applications.

As a consequence, the following elements of an open context infrastructure have to be provided:

- Architecture components that support dynamic context service publishing and discovery.
- A standardized infrastructure context model that defines the structure and semantics of context information that can be provided within the infrastructure, including the specification of entities and their attributes.
- An infrastructure component that manages context entity identities, e.g. persons, locations, …
- A context service description model that gives useful meta information about the context service including the description of the type of context information that can be delivered based on the infrastructure context model, the context entity instances that can be described and the quality of service meta information. This may also include service information including privacy and business rules.
- An identification of context service types according to their special functional scope and capabilities, thus allowing the definition of concrete interface specification for their access.

From the perspective of a context-aware application the focus is on the usage of available context information within an application specific context model. Context-aware applications have their own context models, thus context models of different applications may differ completely from each other depending on their functional focus, e.g. see figure 1.

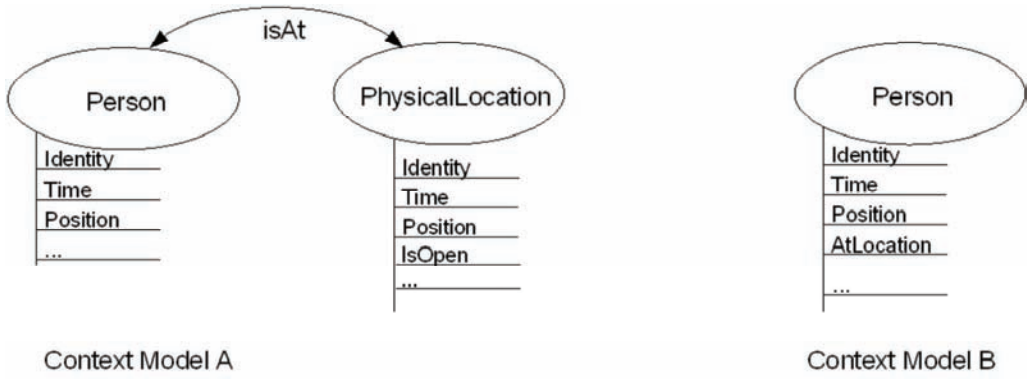


Fig. 1. Application-specific context models

An open infrastructure should support the usage of application specific context models in order to provide sufficient flexibility on the side of the application development. In order to allow application-specific context models the following elements have to be provided by an open context infrastructure:

- Provision of a schema for describing application-specific context models, including the requirements regarding the context information presentation and quality.

- An infrastructure component that is able to operate on the defined application specific context model, including the definition of a context query and definition language and optionally also allowing for reasoning on semantic knowledge on the context model.
- An infrastructure component that is able to match the application-specific context model on the available infrastructure context model.
- An infrastructure component that is able to find and integrate concrete context entity identities (e.g. persons, locations, …) into the instantiation of the application-specific context model.

Another relevant aspect is the type of operations supported on top of the context model. Disregarding the aspect of possible context query types [16] and complexities there is the special requirement that the infrastructure must not only provide ad hoc queries on context information, but also notification support on context changes. This must be covered through suitable infrastructure components, but also on the level of service type definitions.

4 Context Modeling

The context model is one important aspect of an open context infrastructure. Context models give an abstract description of parts of the real world that the system needs to react on. The task of a context model is to build a bridge between the physical world and context-aware applications [5]. The context model separates applications from the process of sensor processing and context fusion [17]. This allows sharing the same context infrastructure between a number of different context-aware applications. Because of its central relevance in this process a number of informal and formal context models have been proposed and are used in various context-aware systems. Until now, however, there is no standardized context model which can be used for any context-aware applications especially within an context infrastructure.

The requirements towards a context model from an infrastructure view differ from the ones of single specialized frameworks. Such a context model has two different purposes which imply different requirements towards its specification. The context model is needed for the integration of context services into the infrastructure. This we call the infrastructure context model. Moreover the application developers need to define a context model that represent the application's view on relevant context information and which enables context queries that are needed for the context-awareness of the application. Such a context model we call the application-specific context model.

4.1 Core Context Model

The core context model identifies those elements of a context model that are relevant both in the infrastructure context model and in the application specific context model. These are the model parts that need to correlate to each other and enable the mapping between those two. Such basic elements have already been identified in most other existing context models. For example in [3] the basic elements are context entities, properties of entities and relations. Refining this approach we define the following elements:

- Entity: An context entity is named and representing an physical or conceptual object, e.g. person, building, electronic device, etc.
- Dimensions: A context dimension represents potential properties of context entities and relations, e.g. time, status, position, etc.
- Attribute: A context attribute describes the concrete properties of an entity by relating it to a context dimension. Attributes can also describe the properties of relations.
- Relations: A context entity is linked to other context entities by uni-directional relations (entity relations). The relation originates at a single entity which we call owner of the relation. A relation that links an context entity or an entity relation with associated context dimensions is represented by a context attribute.

4.2 Infrastructure Context Model

The focus of the infrastructure context model is the integration of different context services from context providers within the infrastructure without consideration of concrete requirements from the applications. It must allow the description of the context information services. Examples for service description models are WSDL [18], ISO19119 [19], OWL-S [20], SWSO [21], WSMO [22] and WSDL-S [23]. Another approach that combines different of these approaches can be found in [24]. These provide the basic service registration information that is not exclusive to context services. Additionally context-related service meta information is needed to describe the content of the context services. This aspect should be addressed using the infrastructure context model. The following requirements towards the infrastructure context model can therefore be identified:

1. Context service providers must be able to describe the content of their context services without knowledge of concrete application usage.
2. Context service consumers must be able to find the appropriate context services that deliver the required context information.
3. Context service consumers must be able to use the provided context services.

4. The model must allow the extension of existing infrastructure context models, thus also allowing the definition of domain specific extensions.
5. The model must support the evolution of the infrastructure context model by allowing the definition of model versions and smooth transitions between different versions.

These requirements lead to the concepts of the infrastructure context model:

- The infrastructure context model basically serves as a common dictionary on context information types, which is used as a contract between context service providers and consumers (Req. 1, 2). Such type definitions can be provided using the basic concepts defined in the core context model.
- Since such a common infrastructure model can be very complex, appropriate extensions are needed that allow for better organizing the concrete context concepts within the model (Req. 1, 2). The definition of inheritance relations between model elements and the definition of abstract context entities are therefore additional model elements that can also be found in ontology-based approaches.
- Context services that provide context information may have different representation formats for a defined context dimension, e.g. different coordinate system for location information. Also context services may have a limited scope on a context dimension, e.g. only providing temperature information from 0°C to 30°C. Therefore, context dimension attributes ‘format’ and ‘scope’ are part of the infrastructure context model which are needed for its usage (Req. 3).
- The extensibility and evolution (Req. 4, 5) are supported by meta information on each model element giving its version and a reference to a previous version, the reference to the owner, an informal description of what it represents, and a semantic reference. Also each model element must have an ID that allows to link between context elements independently from its current version. In order to support the definition of domain-specific extensions the context model is extended by the element ‘domain’, which defines model domains and their sub domains. Concrete model elements can then be referenced by such a domain.

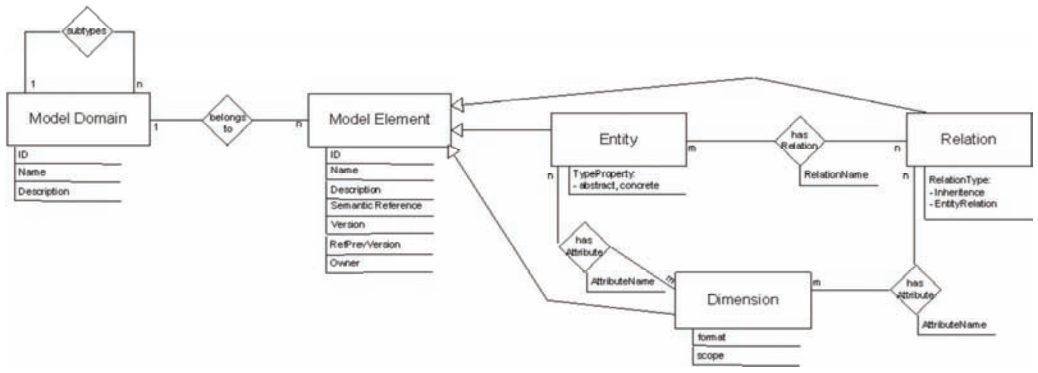


Fig. 2. Elements of the infrastructure context model schema

4.3 Application Specific Context Model

The application-specific context model should allow the application designer to specify the relevant aspects of his context model. The requirements towards the expressiveness of such a context model include the following aspects:

1. Support the definition of a context model that represents all aspects of the environment relevant for the application.
2. Enable flexible design decisions by the application developer driven by application-specific requirements independently from any infrastructure context model.
3. Support of complex context queries as required by the application.
4. Ensure a valid state of the context model despite of faulty and unreliable context information provided by context services.
5. Integrate domain knowledge that can be used for the deduction of additional context information.

These requirements have a different focus than the ones regarding the infrastructure context model. They lead to the concepts that are part of the application context model, which will be outlined roughly. A major concept is the distinction between the static and the dynamic view of the context model. The static view describes the structure of the context model while the dynamic view describes its valid behavior.

- The static view of the application specific context model can be defined using the basic concepts defined in the core context model independently from the infrastructure context model (Req. 1, 2).
- Part of the static view is the definition of the scope of the context entity instances, e.g. all shopping malls in Beijing (Req. 1).
- The static view may also need restrictions on context dimensions as part of the context model, e.g. only temperature between 0°C and 30°C (Req. 1).

- In order to support complex context queries (Req. 3) application specific context models may include attributes that are not provided by context services, e.g. the name or category of a location. Also context attributes can be of complex types, e.g. spatial types or taxonomies. Generalization and specialization relations between context entity or context relation definitions may be required which have to be provided by the application specific context model.
- The dynamic view supports the validation of the context model state and even the deduction of additional context information (Req. 4, 5). It includes the definition of transitional restrictions on single context attributes, e.g. an ordered transition for time. It also includes the definition of interdependencies between different context attributes of a single context entity or even between context attributes of different entities. This enables the definition of derived context dimensions.

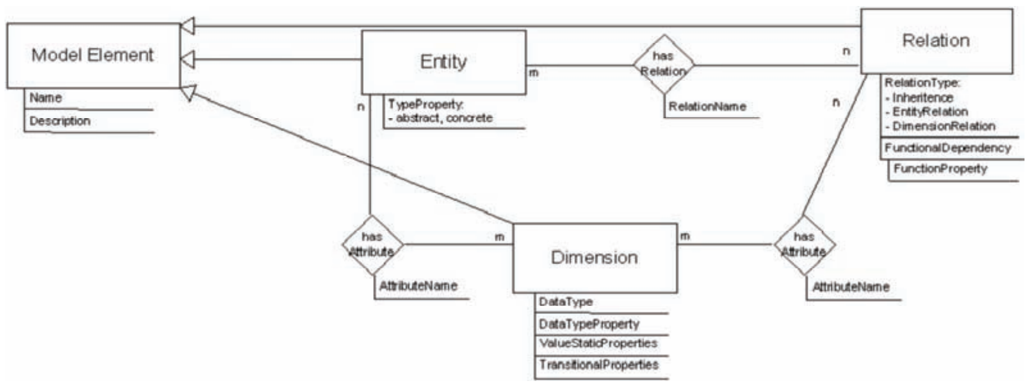


Fig. 3. Elements of the application-specific context model schema

4.4 Mapping between Models

A mapping between these two model types is essential for the application-specific usage of context information provided from within the infrastructure. Such a mapping should be done by a model matching component which is part of the infrastructure. Only the static view of the application-specific context model is relevant for that mapping process. Also those elements that enrich the static view with more complexity (Req. 3) are not needed for the match. Matching is basically performed on the model parts that are identified within the core context model:

- Concrete context entities defined on the application-specific context model must have their counterparts on the infrastructure side. The matching is based on the semantic reference given in the infrastructure model.
- Context dimensions that are not derived must have their equivalent on the infrastructure context model.

- Context relations might also have their counterparts on the infrastructure side. The matching is based on the semantic reference given in the infrastructure model.
- If attributes on context entities or relations defined on the application specific context model have their equivalent on the infrastructure context model then such a matching is rather simple, again using the semantic reference.
- If there is no such direct equivalent then a mapping rule has to be defined based on the elements of the core model. For example an application context model may define the name of the location where the user is located as an attribute of its representing context entity. A rule then has to map this attribute to the name of the location that matches to the current coordinates of the user.

Based on the model matching functionality a discovery component can be used to find those service which provide the required context information. Additionally quality of service requirements, business rules and also the restrictions on the context dimensions have to be considered within the context service discovery process.

5 Conclusion

In this paper we identified different aspects and architectural components that are essential for an open context infrastructure. One major challenge in this aspect is context modeling. There are different requirements regarding a context model depending on its purpose, whether it is used for the registration and retrieval of context services within the infrastructure or whether it has to represent an application-specific view on the environment. Therefore, we identified different schema elements for the infrastructure context model and the application-specific context model. A core context model that describes the common model elements is the basis for the model matching that is needed for the retrieval of appropriate context services from within the infrastructure.

References

1. G. Biegel, V. Cahill: A Framework for Developing Mobile, Context Aware Applications, 2nd IEEE Conference on Pervasive Computing and Communications, Mar. 2004
2. A. Dey, D. Salber, G. Abowd: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, Human-Computer Interaction (HCI) Journal, Volume 16, 2001

3. K. Henricksen, J. Indulska, A. Rakotonirainy: Modeling Context Information in Pervasive Computing Systems, Pervasive 2002, Springer Verlag, 2002
4. P. Nixon et al.: Engineering context-aware enterprise systems, Workshop on Engineering Context-Aware Object-Oriented Systems and Environments, Seattle, 2002
5. D. Salber, A. K. Dey, G. D. Abowd: The Context Toolkit: Aiding the Development of Context-Enabled Applications, CHI'99, 1999
6. C. Becker, D. Nicklas: Where do spatial context-models end and where do ontologies start? A proposal of a combined approach, Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management in conjunction with UbiComp 2004, Sept. 2004
7. J. Hightower, D. Fox, G. Borriello: The Location Stack The location stack: A layered model for location in ubiquitous computing. In Proceedings of the 4th IEEE Workshop on mobile Computing Systems & Applications (WMCSA 2002), pp. 22-28, Callicoon, NY, USA, June 2002. IEEE Computer Society Press
8. H. Ailisto, P. Alahuhta, V. Haataja, V. Kyllönen, M. Lindholm: Position Paper - Structuring Context Aware Applications: Five-Layer Model and Example Case, VTT Electonics, Aug. 2002
9. K. Henricksen, J. Indulska: A Software Engineering Framework for Context-Aware Pervasive Computing, Proc. of the Second IEEE International Conference on Pervasive Computing and Communications, Orlando, Florida, (PerCom'04), March 2004, IEEE Computer Society, pp. 77-86
10. S. Haseloff: Context Awareness in Information Logistics, PhD Thesis, TU Berlin, 2005
11. F. Dürr, N. Hönle, D. Nicklas, C. Becker. K. Rothermel: NEXUS – A Platform for Context-aware Applications, 1. Fachgespräch Ortsbezogene Anwendungen und Dienste der GI-Fachgruppe KuVS, Universität Stuttgart : Sonderforschungsbereich SFB 627 (Nexus: Umgebungsmodelle für mobile kontextbezogene Systeme), Informatik-Bericht der FernUniversität Hagen, June 2004, pp. 15-18
12. Markus C. Huebscher, Julie A. McCann, An adaptive middleware framework for context-aware applications, Personal and Ubiquitous Computing, Volume 10, Issue 1, Feb 2006, Pages 12 - 20, DOI 10.1007/s00779-005-0035-6
13. A. Schmidt, K. Asante Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, W. Van de Velde: Advanced Interaction in Context, In Proceedings of the 1st international Symposium on Handheld and Ubiquitous Computing (Karlsruhe, Germany, September 27 - 29, 1999), Ed. Lecture Notes In Computer Science, vol. 1707. Springer-Verlag, London, 89-101
14. Open GIS Consortium: The OpenGIS Abstract Specification, Topic 12: OpenGIS Service Architecture, Version 4.3, Jan. 2002
15. EPCglobal: The EPCglobal Architecture Framework, http://www.epcglobalinc.org/standards_technology/Final-epcglobal-arch-20050701.pdf, Jul. 2005
16. C. Becker, F. Dürr: On location models for ubiquitous computing, Personal Ubiquitous Comput. 9, 1, pp. 20-31, Jan. 2005

17. E. Chrisopoulou, C. Goumopoulos, I. Zaharakis, A. Kameas: An Ontology-based Conceptual Model for Composing Context-Aware Applications, Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management in conjunction with UbiComp 2004, Sept. 2004
18. W3C: Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl, 2001
19. Open Geospatial Consortium Inc: ISO19115/ISO19119 Application Profile for CSW 2.0, https://portal.opengeospatial.org/files/?artifact_id=8305, 2005
20. The OWL Service Coalition: OWL-S: Semantic Markup for Web Services, http://www.daml.org/services/owl-s/1.0/owl-s.pdf, 2004
21. Semantic Web Services Initiative: Semantic Web Services Ontology (SWSO), draft version 1.1, http://www.daml.org/services/swsf/1.1/swso/
22. WSMO.Org: Web Service Modeling Ontology, WSMO Final Draft, http://www.wsmo.org/TR/d2/v1.1/D2v1_20050210.pdf, 2/2005
23. J. Miller, K. Verma, P. Rajasekaran, A. Sheth, R. Aggarwal, K. Sivashanmugam: WSDL-S: A Proposal to W3C WSDL 2.0 Committee, http://lsdis.cs.uga.edu/Projects/METEOR-S/, 2004
24. N. Weißenberg, A. Voisard, R. Gartmann: Using Ontologies in Personalized Mobile Applications, Intl. ACM GIS Conference, ACM Press, D. Pfoser and I. Cruz (Eds.), 2004

BeTelGeuse: Tool for Context Data Gathering via Bluetooth

Patrik Floréen, Joonas Kukkonen, Eemil Lagerspetz,
Petteri Nurmi and Jukka Suomela

Helsinki Institute for Information Technology HIIT
Basic Research Unit
Department of Computer Science, P.O. Box 68
University of Helsinki, FI-00014 Finland
`{firstname.lastname}@cs.helsinki.fi`

Activity related information is often seen as a good indicator for human behaviour, see e.g. [1–5] and, as a consequence, context-aware applications and services need access to activity related context data. In mobile and wearable computing, the dominant approach has been to build customized sensor boards (e.g. Muffin [6]) and armbands or other fabrics (e.g. The Bodymedia SenseWear PRO$_2$ Armband [7]) which are equipped with sensors that provide physiological information (e.g. galvanic skin response, acceleration, heart rate). The decrease in the prices of Bluetooth sensor chips has made another alternative viable. Namely, sensors with Bluetooth capability can communicate their measurements via Bluetooth to mobile devices and the data can be analyzed directly on the device. However, currently generic tools that can be run on different platforms are not available and this hinders development and research in the fields of pervasive and mobile computing. To improve the situation we have developed a Java-based tool, BeTelGeuse, that can be used on any MIDP 2.0 [8] compliant device that supports the CLDC 1.1 profile [9] to which a JSR-82 compliant Bluetooth stack [10] is available. Our tool is freely available under the LGPL license and it has been tested with the following configurations: laptop running Windows XP, desktop computer running Windows 2000, desktop computer running Linux, Nokia 6680, Sony Ericsson W800i. In our demonstration we show how the tool works and give information on how to use and extend it.

In previous research, data gathering tools have been developed for specific environments. For example, the ContextWatcher [11] and the ContextPhone [12] run on Nokia Series 60 [13] phones. The Context Watcher can gather GPS, heart rate, speed and distance information whereas the only physiological information that Context Phone currently supports is GPS information. Closest to our work is the IBM Mobile Health Toolkit [14], which contains a MIDP application that runs on a mobile phone and which uses a specific set of health sensors (part of the kit) to gather remote medical data. The main difference is that our tool is not restricted to a specific set of sensors, but it can be easily extended to use additional sensors.

The BeTelGeuse allows defining rules that can be used to define which parser to use to read data from a particular Bluetooth device. This way Bluetooth sensors can be discovered automatically and the correct parsers can be automatically initialized. At the moment we have parsers for a GPS device and for a Suunto ANT-2-BT module, which allows us to obtain heart rate, distance and speed information.

The BeTelGeuse runs periodically a Bluetooth device discovery and, if new devices are found, initializes the appropriate parsers. Thus, the tool can discover new devices while running. In addition, if a connection to an existing device is lost, the BeTelGeuse automatically tries to re-establish the connection. If the connection cannot be established on the first retry attempt, an exponential backoff is used so that, the longer time ago the connection died out, the rarer the retry attempts are. The architecture of the BeTelGeuse is modular so that new parsers, as well as other components, can be easily added to the tool later on.

References

1. Isbell, Jr, C.L., Omojukon, O., Pierce, J.S.: From devices to tasks: automatic task prediction for personalized appliance control. Personal Ubiquitous Computing 8 (2004) 146–153
2. Desai, N., Kaowthumrong, K., Lebsack, J., Shah, N., Han, R.: Automated selection of remote control user interfaces in pervasive smart spaces. In: Proceedings of HCIC Workshop on Pervasive Smart Spaces. (2002)
3. Moeslund, T.B., Granum, E.: A survey of computer vision-based human motion
   capture. Computer Vision and Image Understanding 81 (2001) 231–268
4. Philipose, M., Fishkin, K.P., Perkowitz, M., Patterson, D.J., Fox, D., Kautz, H., Hähnel, D.: Inferring activities from interactions with objects. IEEE Pervasive Computing 04 (2004) 10 – 17
5. Chen, J., Kam, A.H., Zhang, J., Liu, N., Shue, L.: Bathroom activity monitoring based on sound. In: Proceedings of the 3rd International Conference on Pervasive Computing (PERVASIVE). Volume 3468 of Lecture Notes in Computer Science, Springer-Verlag (2005) 47 – 61
6. Yamaba, T., Takagi, A., Nakajima, T.: Citron: A context information acquisition framework for personal devices. In: Proceedings of the 11th International Conference on Embedded and Real-Time Computing Systems and Applications, IEEE (2005)
7. : BodyMedia Health Monitoring System. http://www.bodymedia.com/main.jsp (2006)
8. : Mobile Information Device Profile.
   http://java.sun.com/products/midp/index.jsp (2006)

9. : Connected Limited Device Configuration (CLDC) 1.1. http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html (2006)
10. : Java API for Bluetooth. http://www.jcp.org/en/jsr/detail?id=82 (2006)
11. : Context Watcher. http://www.lab.telin.nl/ koolwaaij/showcase/crf/cw.html (2006)
12. Raento, M., Oulasvirta, A., Petit, R., Toivonen, H.: ContextPhone: A prototyping platform for context-aware mobile applications. IEEE Pervasive Computing 4 (2005) 51 – 59
13. : Nokia Series 60. http://nokia.com/series60 (2006)
14. : IBM Mobile Health Toolkit. http://www.zurich.ibm.com/mobilehealth/ (2006)

Context Awareness: Telco Perspectives

Nicoletta Salis, Carlo Alberto Licciardi

Telecom Italia Lab, Via G. Reiss Romoli, 274
10148 Turin, Italy
{nicoletta.salis, carloalberto.licciardi}@telecomitalia.it
http://www.telecomitalialab.com

Abstract. This article describes TILAB's view about the new services based on Context Awareness. After the latest multiradio devices spread, the interest towards the context aware services based on a unique device with different access technologies (Wi-Fi, GSM, Bluetooth) is increasing, even from the telcos, in the perspective of the current fixed-mobile convergence scenario. This paper describes TILAB's approach to these services, beginning from the definition of a context architecture, aimed not only at the user context calculation and management, but also at having a strong relationship with a proper Service Creation Environment. TILAB's experience about the Service Creation Layer for the implementation of Context Aware Services is shown with the example of the concept "My Movie", which helps the user to choose the best movie for his context and preferences. This is a starting point to optimize the exploitation of all the opportunities coming from these new categories of services.

1 Introduction

The Context Aware Services [1,2] (CAS) can offer personal services, based on the current context of the user and can help the user with contents, services and specific applications related to the situation he is really living.

The main advantage for the user is to be/work in an environment which "recognizes" his need, getting services and information useful for his current activities, in an easy, transparent and quick way.

Telcos, as service providers, may be interested in the implementation of CAS, in particular because they have a decisive advantage over the other service providers: they already have the network infrastructure (GSM/UMTS networks, wi-fi hotspots) necessary to offer/customize so services and with different infrastructures both for fixed (Wi-Fi, Bluetooth) and for mobile (GSM/UMTS) services, they also have the possibility to work independently of the type of network (CAS for Fixed-Mobile convergence [3] [15]) the user is connected to, using the network information as a context parameter useful to discriminate the best service for the user. At the same time, the "multi-radio" (GSM/GPRS, Wi-Fi, BlueTooth, GPS, …) devices spread is increasing in order to access both to the fixed and to the mobile network.

Another important advantage of telcos is the customer ownership: they can offer to their users customized services by the knowledge of their specific profiles; now this knowledge is often static and, however, not directly linked to the actual user situation. So, in order to migrate to CAS, it is necessary to integrate the user profile with further information, exploiting the available user devices to collect the context data.

2 An Architecture for CAS

The technical parameters useful to specify a context can be various and are, often, dependent on the service to provide; in any case, some of them are static or vary very slowly (such as the user profile or interests), others, instead, are very dynamic, such as spatial information (location, direction, speed), time, environmental information (temperature, noise) and, for the determination of a lot of them, a software intelligence is necessary in the user device.

The architectural scheme developed in 2005 for context awareness has been studied even on the basis of attending at international workshops/meetings [13] and from TILAB's direct involvement in IST Projects, such as Mobilife [14] (see Fig. 1).

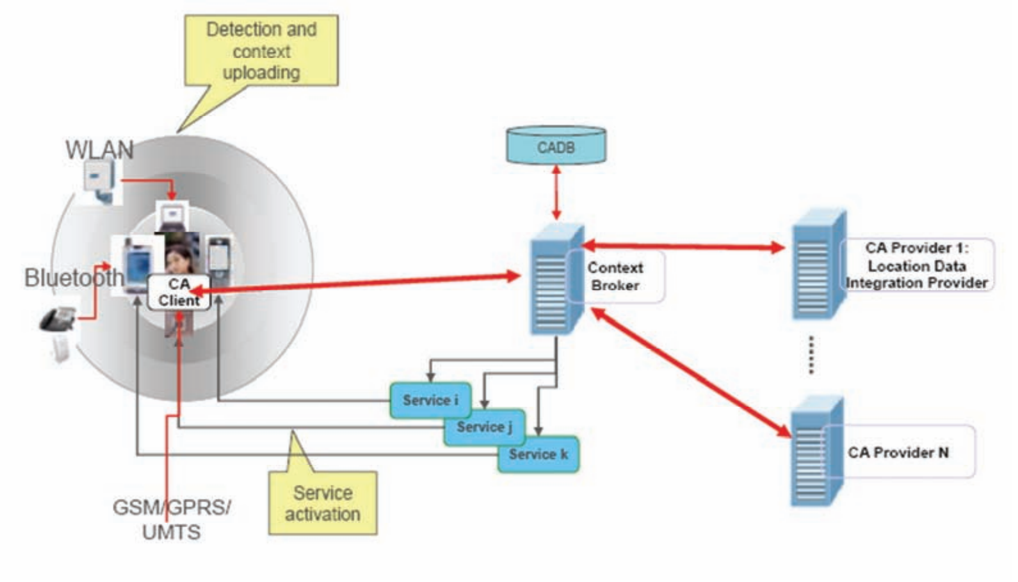


Fig. 1. Current TILAB's architecture for Context Awareness

In Fig. 1, the main elements are:

- the Context Broker, which is the central element of the architecture, as active to collect both the local user context (from the CAClient in the user device) and the context data from the external Context Provider

- the CA Client, as it is necessary a software intelligence on the user device too, in order to capture important context data such as position, device capability, available access networks, possible external triggers, …
- the CA Providers, which the Context Broker can ask context parameters to (e.g., the position, in an integrated way (see §2.1), from the Location Data Integration Provider)
- the CA DB, as a repository for the context data collected by the Context Broker, as they can be available for requesting third party services.

2.1 Location Data Integration Provider

One of the most important context parameters is the user location.

The localization mechanism used exploits the radio technologies of the user device to calculate the position; the basic idea is to locate a device through the "a priori" knowledge of the fixed position of the network points the device is "hooked up" to and, consequently, with an accuracy variable with the technology used.

The technologies studied in Telecom Italia Lab for this approach are GSM, GPS, Wi-Fi and Bluetooth, each with different spread and accuracy:

GPS: its accuracy (~some meters) is dependent on the satellites visibility, which grows worse in indoor and in urban environments [4].

GSM: it is the most spread technology among the mentioned ones, with consequent most service availability, but gives less accuracy in the localization (~some hundred meters).

Wi-fi: it lets the user be located with a good accuracy (~ten meters) in the limited areas where the Wi-Fi coverage is present (buildings, airports, stations, …) [6]; the Wi-Fi hotspot coverage is not continuous.

Bluetooth: it has an accuracy variable (by ten or one hundred meters) according to the class (emitted power) of both the mobile device and the Bluetooth object used as "hooking up" point [5].

The latest spread of multi-radio user devices, including different access technologies on a unique device, makes reasonable the idea of having a *location data integration*, which integrates the data from the different available localization techniques to obtain both better accuracy and a more seamless service, on the basis of the complementary characteristics of the different technologies besides their alternative network availability [see Fig.2].

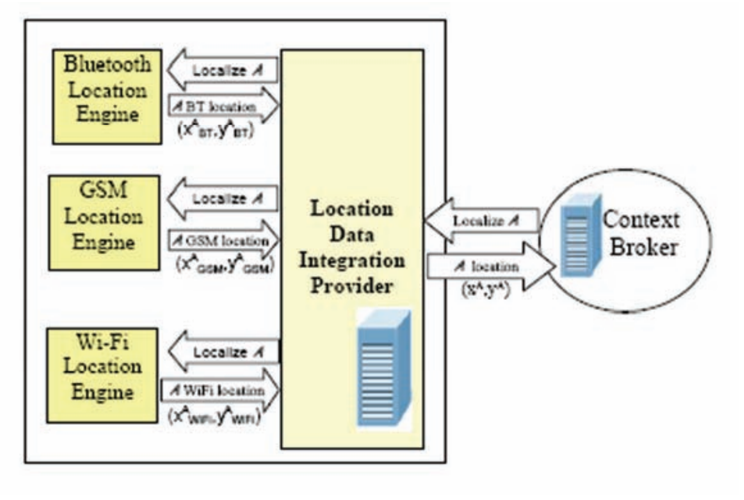


Fig. 2. Location data integration technique

This integration is reasonable only for localization techniques with comparable accuracy: so, with Bluetooth and Wi-Fi, it is possible to exploit both location data, with GSM and Wi-fi it is not. The location data can be seen as a circle where the center is the estimated location and the radius is the accuracy: the circle represents the uncertainty area and the real device location may be any of the points of the circle area. The integration lets the uncertainty area be reduced, by the intersection among the different circles obtained from the different localization techniques.

3 An Advanced Service Creation & Execution Environment for CAS

The CAS shown in Fig. 1 are the meeting point of the different elements of the architecture: the calculated context parameters are raw data which innovative services can exploit to give more and more functionality to their users according to their real-time needs.

Telcos, too, can have an important role in this innovation: as a matter of fact, the differentiation and the competition on the market is mainly based on the proposition of attractive and easy use of new services, as well as the ability to quickly adapt services to the new requirements of the customers. It's therefore really important to make use of tools that simplify and speed up the creation and deployment of services, to reduce time-to-market and to "tune" properly the services themselves by the customers feedback. As for CAS, in particular, as "disruptive" services, this process seems really useful and important.

In this perspective, a Service Creation Environment (SCE) must provide high level features that allow reusing efficiently the huge catalogue of existing high-level or more basic building-blocks existing either at the platform side or directly through the web. Since all these bricks provide a number of nice and rich capabilities, it is foreseen that providing new services will consist of combining, orchestrating or controlling elementary service components, network features and third party services, taking also into account other sources of information such as user's profile and preferences, user context etc.

3.1 Service Execution Models for the Service Platform

Service execution should ease integration of heterogeneous fixed and mobile resources (SIP Application Server, Messaging server, Conferencing server) by means of abstraction of resources and protocols. The execution model should allow an execution engine to host libraries of building components which hide the complexity of underlying resources.

Also, the advanced service execution model will require the service to be distributed among the service platform and the mobile terminal since the service will leverage on components available on both sides. Once designed, the service must be delivered in an easy and seamless way to the execution engines to become executable.

TILAB has developed an event-based Service Logic Execution Environment - named StarSLEE - and the corresponding SCE (StarSCE).

The StarSLEE engine supports the execution of event-based service logics that span different application platforms and networks. The engine is modeled following the emerging JAIN Service Logic Execution Environment (Jain SLEE) framework and can be easily connected to different application platforms or network elements via a "Resource Adaptor" mechanism. Fig. 3 shows some Resource Adaptors to different external platforms and elements, some of them already developed.

JAIN SLEE [8,9] provides a standard developer community. The programming model has been designed to simplify the work of the application developer, eliminate common programmer errors and ensure that robust services can be developed rapidly.

3.2 StarSCE

StarSCE is based on the industry standard Eclipse IDE and allows the development of advanced services by simply using graphic composition of a portfolio of atomic services, called "Service Building Blocks" (SBB).

The services developed using this tool are described by an XML language defined by TILAB (also known as service composition language). The designer tool is fully graphic and does not require any Java programming skills for the development of composed services starting from the portfolio of available service building blocks.

Using this tool, TILAB developed many SBB to enable the fast development of innovative services. The most important are: Call control (using SIP and OSA interfaces), Multipartycall control, Presence component, Instant messaging and presence (using SIP), SMS and MMS messaging, GSM, Wi-Fi, Bluetooth localization and the above-mentioned Location Data Integration Provider, Generic Web Services invocation (whether forecast, stock quote, yellow pages, ...) and non-functional blocks to control the service work-flow (for, loop, switch, timer, join...).

Now the challenge is the use of this instrument to develop CAS: in Fig. 3, a first hypothesis for the integration of CA architecture in StarSLEE is shown.

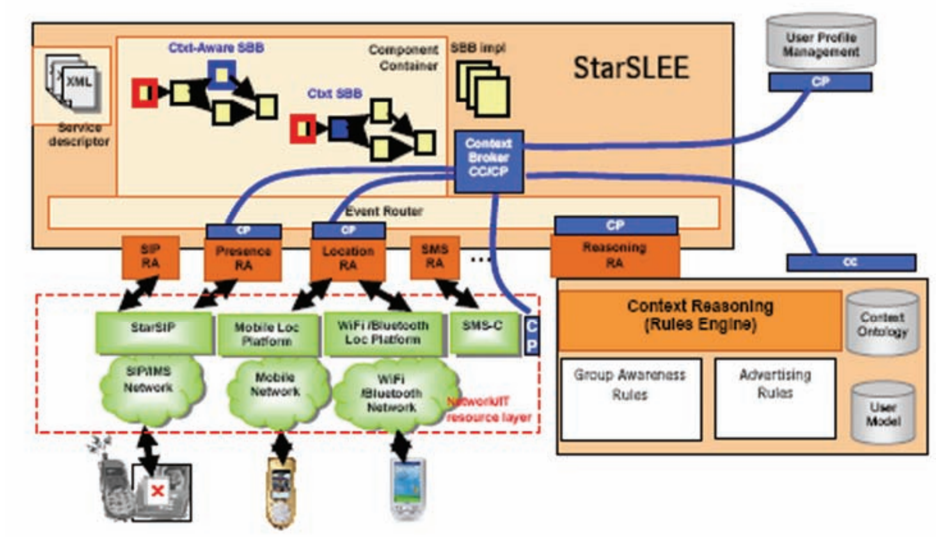


Fig. 3. First hypothesis for the integration of the CA architecture in StarSLEE

Context information will be retrieved from several context sources: network elements, user equipments (PC, mobile phones, sensors), local networks (printer, projector, enterprise data center).

The platform will provide a reasoning engine that will understand (semantically), compose and provide the application layer with a fully fledged context representation.

From the business perspective it is relevant to clearly define the following roles:

a) Context Provider (CP): it is either a network element or an end user device which provides context data
b) Context Consumer (CC): it is the application layer (either server or client side) which uses or elaborates context information. This role can be played also by the reasoning engine, which is an intermediate element between the context provider and the application.

c) Context Broker: it is a server element which collects context data (acting as context consumer) and distributes them (as a context provider) on request. Often the context broker adds value to the individual data which it collects.

All context sources must share the same context ontology, which will be used by the context data consumer and by the context broker.

4 Building a Context Aware Service with the StarSCE Graphical Designer: the Concept "MyMovie"

This section describes an example of a Context Aware Service developed by using the service creation designer.

In the service idea, the user can exploit his (possibly multi-radio) device to receive the updated information of the movie planning in his town, for example when he approaches an information point (service #1, see Fig. 4). He will receive only movies programmed from the current time (time) on and in the cinema closest to him (location), according to his preferences (user profile) and to the device capability (device).

Table 1. Link between user context and trailer quality received

Connectivity	Screen size	Trailer Quality
WI-FI	≥ 320 x 240	High (see Fig. 8)
Bluetooth	≥ 320 x 240	Low (see Fig. 5)
<any>	< 320 x 240	No trailer

Only if the user device has the proper screen size and type of connectivity, the poster and the trailer of the movie will be received (e.g., video and images will be low quality if the connectivity is Bluetooth and high quality if connectivity is Wi-Fi, see Table I).

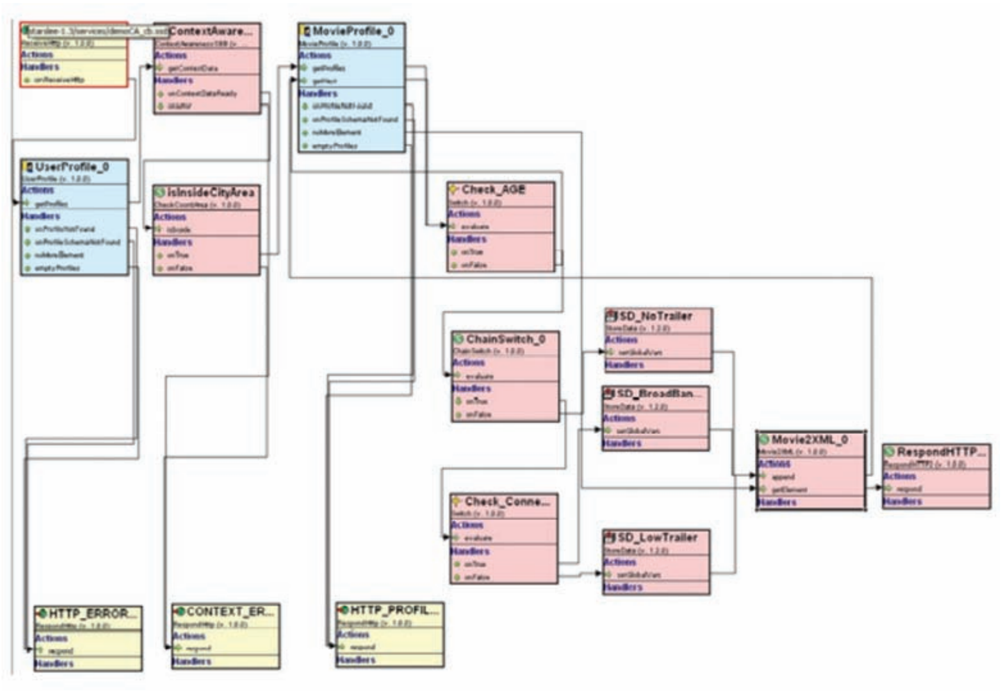


Fig. 4. StarSLEE Service #1: movie planning in the cinema closest to the user

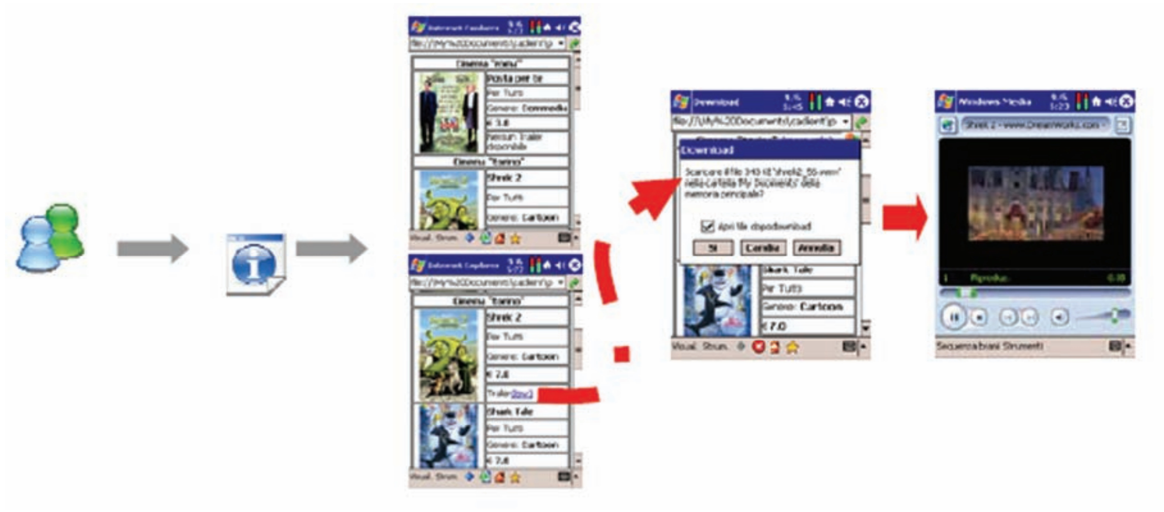


Fig. 5. PDA screen shots from Service #1: cinema closest to the user and download of a low quality trailer (Bluetooth connectivity)

In the next step of the service (service #2, see Fig. 6), the user is in the cinema chosen and he has the possibility to receive more detailed information, pushing the button "More Info" on his graphic interface.

According to his position, calculated indoor with the accuracy of some meters (Wi-Fi, Bluetooth positioning engine), it is possible to distinguish if the user is close to the refreshments area or to the cinematography hall and, consequently, to push him the proper contents (snack ads or movie wefts). If there is a wireless printer in the range of the device, the device can detect it and a pop-up will appear on the screen and will show the possibility to print information and/or tickets for the movie, without any particular configuration of the printer itself.

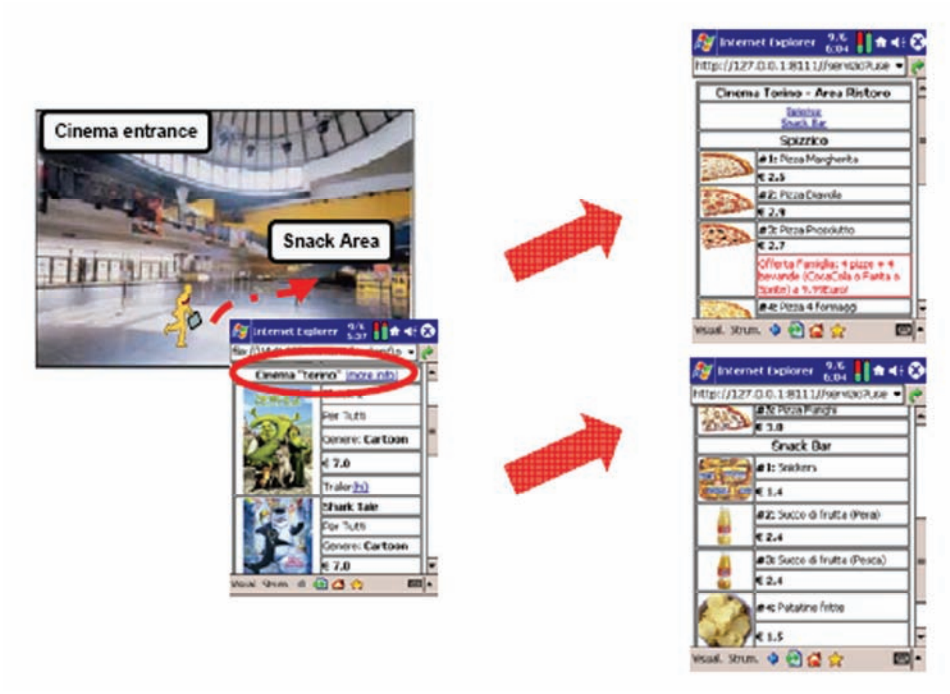


Fig. 6. StarSLEE Service #2 (inside the cinema): ads of the refreshments

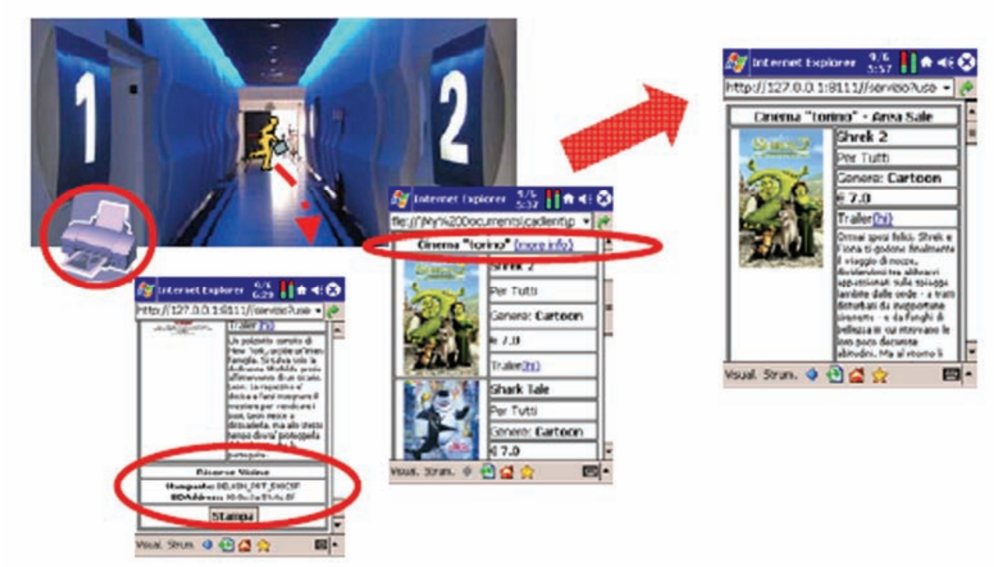


Fig. 7. StarSLEE Service #2 (inside the cinema): movie planning details and check of wireless printers in the range

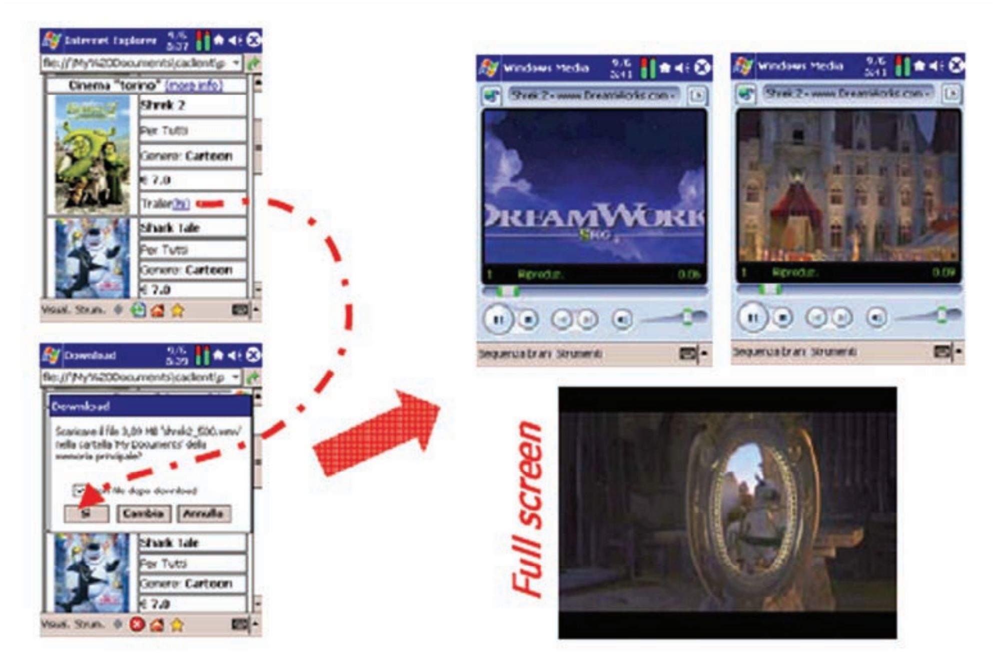


Fig. 8. StarSLEE Service #2 (inside the cinema): download of a high quality trailer (Wi-Fi connectivity)

It is clear that the context parameter of the user position is calculated in different ways during the two phases of this concept: as a matter of fact, in service #1 the user is outdoor and he is located by TILab's GSM location engine, while in service #2, inside the cinematography hall, the localization is short-range. In any case, each SCE/StarSLEE service asks the Context Broker for all the context parameters of the user: it will be the Context Broker itself to ask the Resource Adaptor of Location Data Integration Provider (see Fig. 4) for the user position, which will be, automatically, calculated exploiting all the radio links available at that moment.

Technically speaking, the service #1 is activated by a trigger event from the approaching of the user device to a Wi-Fi/Bluetooth Access Point. This triggering causes an http request from the device to the application server with the specification of the user_ID; this identifier will be used by the service to make queries to the user profile DB. Then the service asks the Context Broker to know all the available con-text information of that user (position, device capability, connectivity on and IP ad-dress, static profile, …):

- the position is useful for the interrogation of movies DB
- the user profile is useful for the comparison between the user age and the age permitted for the specific movie and in the user movie preferences
- the device screen size and the connectivity on real-time are useful for the choice of the quality (high/low) of the trailer to send

The reply of the service is sent to the user by XML.

The service #2 is similar to the service #1 in the first phase, but the check on the user position is important to distinguish the contents to send the user: detailed movie planning or available snacks with related prices and offers. In the former case, the previous information of the chosen cinema is enriched with the wefts; in the latter one, it is necessary to make a query to the DB of the refreshments and to create an XML as a reply for the client side of the service.

5 Conclusion

Context Awareness is a new challenge in the arena of innovative communication voice and data services. Telecom operators are, among the others, investigating in order to understand its potential.

TILAB is going to exploit its service creation environment as a powerful tool for a fast prototyping of these new services. The tool speeds up the whole creation process from the service idea to the actual service usage and it enables the service provider to deploy and modify several new services with low incremental investment (compared with full featured monolithic service platform). This approach is a strategic weapon for the operators, decreasing the risk of launching and customizing new services: the concept MyMovie described here is only a first example in this research activity, which will focus on both the service point of view and the architectural and platform one.

References

1. P. Brown, W. Burleson, M. Lamming, G. Romano, J. Scholtz and D. Snowdon, ‘Context-awareness: some compelling applications’, In Proceedings the CH12000 Workshop on The What, Who, Where, When, Why and How of Context-Awareness, April 2000
2. Mitchell Keith, A Survey of Context-Awareness, 2002 Available: http://www.comp.lancs.ac.uk/~km/papers/ContextAwarenessSurvey.pdf (21 March, 2005)
3. Aslam, “Fixed Mobile Convergence – some considerations”, In LCS 2003. Available: http://www.ee.ucl.ac.uk/lcs/papers2003/138.pdf
4. GPS Reference, March 2005, http://www.gpsworld.com/gpsworld/
5. Specification of Bluetooth System. Core, Version 2.0. November 4, 20042003
6. N. Patwari et al., "Relative Location in Wireless Networks," Proc. IEEE Spring VTC 2001, pp. 1149-1153
7. www.omtp.org
8. JAIN SLEE specification Version 1.0, Java Community process, JSR 22, http://www.jcp.org/en/jsr/detail?id=22
9. JAIN SLEE specification Version 1.1 (under review), Java Community Process, JSR 240, http://www.jcp.org/en/jsr/detail?id=240

10. Falcarin, P., Licciardi, C.A., Analysis of NGN service creation technologies. In IEC Annual Review of communications, volume 56, June 2003
11. Lago, P., Licciardi, C.A et alii., An architecture for IN-Internet hybrid services. In Computer Networks Journal, Special Issue on Intelligent Networks and Internet Convergence, T. Magedanz, H. Rudin, I. Akyildiz (Eds.), Elsevier, Vol. 35(5), April 2001, pp. 537-549
12. Andreetto, A., Licciardi, C.A., Falcarin, P., Service opportunities for Next Generation Networks, In Proceedings of the Eurescom Summit 2001, Heidelberg, Germany, November 2001.
13. www.cs.helsinki.fi/hiit_bru/conferences/caps2005
14. www.ist-mobilife.org/
15. Licciardi, C.A, Milanese, I., Fixed Mobile Convergence: 3 Words, Many Perspectives. In Procedures of MATA’05

Controlling Services in a Mobile Context-Aware Infrastructure[1]

Patrícia Dockhorn Costa, Luís Ferreira Pires, Marten van Sinderen, Tom Broens

Centre for Telematics and Information Technology, University of Twente, The Netherlands
{dockhorn, pires, sinderen, broens}@cs.utwente.nl

Abstract. Context-aware application behaviors can be described as logic rules following the Event-Control-Action (ECA) pattern. In this pattern, an Event models an occurrence of interest (e.g., a change in context); Control specifies a condition that must hold prior to the execution of the action; and an Action represents the invocation of arbitrary services. We have defined a Controlling service aiming at facilitating the dynamic configuration of ECA rule specifications by means of a mobile rule engine and a mechanism that distributes context reasoning activities to a network of context processing nodes. In this paper we present a novel context modeling approach that provides application developers and users with more appropriate means to define context information and ECA rules. Our approach makes use of ontologies to model context information and has been developed on top of web services technology.

1 Introduction

The dynamic nature of context-aware applications, and the increasing integration of these applications into our daily tasks in a variety of domains (e.g., home, work and leisure), generate rapid changes in the requirements for the technology to support these applications. Although it is not possible to fully predict these changes, the supporting technology can be designed in such a manner that it can be configured to match changing requirements, preferably at runtime. This calls for a high level of flexibility. We aim at coping with these issues by means of a shared Context Handling Infrastructure to support context-aware applications. This infrastructure comprises, among others, reusable context processing and managing services, which facilitate context-aware application development. It provides building blocks that can be combined and specialized to satisfy application-specific requirements. A central building block in our Context Handling Infrastructure is the Controlling Service. This service takes application-specific rules and

[1] This work is part of the Freeband AWARENESS project (http://awareness.freeband.nl). Freeband is sponsored by the Dutch government under contract BSIK 03025.

information (context) models as input in order to carry out application-specific adaptation within the infrastructure, at runtime.

This paper aims at presenting our Controlling service, which facilitates the configuration of application-specific behaviors. Application requirements, expressed in terms of pieces of application behaviors, are written in a scripting format following the Event-Control-Action (ECA) pattern. In this pattern, an Event models an occurrence of interest (e.g., a change in context); Control specifies a condition that must hold prior to the execution of the action; and Actions represent the invocation of arbitrary services. The Controller component facilitates the configuration of the infrastructure by taking ECA rules and application-specific context models as input to (i) subscribe to context sources, (ii) perform context reasoning, and (iii) trigger actions on behalf of applications, in response to context changes. We have developed a scripting language for the purpose of writing context-aware ECA rules. This language is composed of an information part, defined by our context models, and a behavior part, defined by the language metamodel. Since ECA rules are written in a scripting format, application developers do not need to write programming code.

Furthermore, we propose a generic context model that incorporates a novel context categorization scheme that classifies context according to its nature, providing application developers and users with more appropriate means to define context information and ECA rules. Our approach provides a generic context model that captures general concepts and allows domain-specific and application-specific extensions.

We focus on mobile context-aware applications that are widely distributed and are typically offered by telecommunication providers. Examples of such applications are healthcare tele-monitoring applications, tourism applications and communication applications. We ignore sensory issues in this paper, but rather focus on the service infrastructure that leverages on the sensor network to provide appropriate context information to a large range of context-aware applications. Since the nature of applications is diverse, a rich set of context information is exploited by the infrastructure, including location of people and devices, vital signs and user activity, amongst others.

The remainder of this paper is structured as follows: Section 2 presents the Controlling service, and identifies the challenges to realize such services; Section 3 presents our context model; Section 4 discusses our language to describe ECA rules; Section 5 presents an overview of the infrastructure architecture and our prototyping activities; Section 6 discusses related work; and Section 7 gives final remarks and conclusions.

2 The Controlling Service

A Controlling Service accepts ECA rule specifications and activates them within the infrastructure. ECA rule activation occurs at infrastructure runtime, which requires runtime discovery and composition of context and action services. Context services aim at providing context information and action services implement the actions to be triggered when context conditions are satisfied. Fig. 1 depicts a typical usage flow of the Controlling Service.

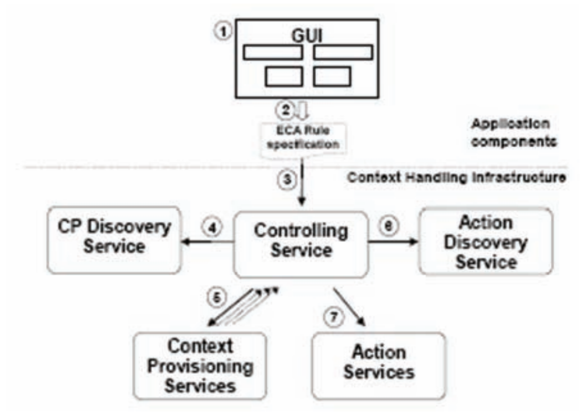


Fig. 1. Typical usage flow of the Controlling Service

The following phases are identified:

- Phase 1 initiates with end-users defining application behaviors by means of a graphical interface.
- Phase 2 consists of performing the mapping of an end-user rule specification to a less abstract specification to be provided to the infrastructure, in a scripting format (e.g., XML). The translation from users' inputs to a rule specification in some notation that can be accepted by the infrastructure is a responsibility of the application components. It is also possible that application developers specify application rules, as opposed to end-user rules. End-user and application rules are equally treated in this paper.
- Phase 3 consists of the actual invocation of the Controlling Service after the rule specification has been provided to the infrastructure. The Controlling Service verifies whether the specification is well-formed and separates it into events, conditions and actions.
- Phase 4 corresponds to the attempt of the Controlling Service to find event sources capable of providing context event notifications of interest. The Controlling Service decides whether or not to subscribe to one of more of these Context Provisioning services.

– Phase 5 consists of the exchange of a subscribe request and eventual event notifications. The Controlling Service determines whether the conditions are satisfied by the context event notifications being generated.
– Phase 6 is entered typically when a certain condition is satisfied. At this moment, an action should be triggered, and, therefore, its actual implementation needs to be found. For that purpose, the Controlling Service makes use of the Action Discovery Service.
– Phase 7 emcompasses the actual execution of an Action Service.

Although much study has been carried out in each of the topics mentioned above, the following research questions remain open: (i) how expressive should the ECA rule language be to accommodate user's and developer's requirements? (ii) what are the context abstractions needed to effectively compose application behaviors? (iii) what elements should be included in ECA rule specifications to provide enough information to perform infrastructure configuration? (iv) how to dynamically discover context provisioning services based on ECA rule specifications? and (v) how to invoke action services on behalf of application components? This paper provides answers for the three first questions in terms design solutions.

3 Context Modeling

A shared context model formally defines context information concepts and their relationships, such that context information can be distributed and unambiguously interpreted by interacting system parts. Our approach requires a context model to (i) provide application users and developers with appropriate means to describe context information and application behaviors; (ii) allow applications, infrastructure and third-party service providers to agree upon syntax and semantics of context information, thus enabling interoperation; and (iii) provide context processing components with proper means to perform context information reasoning. We have used ontologies to model context information in our infrastructure.

3.1 Characteristics of Context Information

We use the context modeling abstractions of *facts* and *situations* [4][5] to provide application developers and users with more appropriate means to define context information. A *Fact* defines a current "state of affairs" in the user's environment, such as "Bob has access to PDA and PC" and "Bob and his PDA are co-located in room A". The *situation* context abstraction allows application developers and users to leverage on the *fact* abstraction to derive high-level context information, such as `isOccupied`, derived from the fact that Bob is engaged in an activity and `isReachable`, derived from the fact that Bob is near to a device that supports a given communication channel. Situations may be built upon other situations, for example, `isAvailable` may be defined as Bob not being `isOccupied` and being `isReachable`. Application behaviors are defined at runtime as ECA rules, using our context models. Section 5 elaborates on ECA rules.

Since our service infrastructure supports a large number of mobile context-aware applications, a rich set of context information is exploited. However, it is not possible to define a complete context model that is accepted by all applications, since each application may define context information in a different way. For example, the context information `near` could mean *within 10 meters* in one application and mean *within 5 kilometers* in another. It may also be possible that certain context information types are domain-specific, rather than application-specific. For example, `heart-rate` and `body-temperature` are types of context information concepts shared among applications in the medical domain, but may be useless concepts in other domains. We suggest in our approach a general context model that contains concepts shared by applications we deal with. This model should be extended with application-specific concepts (facts and situations) on demand, at infrastructure runtime.

3.2 Context Models

Fig. 2 depicts some parts of our general context model. This model is a context ontology that captures general concepts and allows domain-specific and application-specific extensions. We have used OWL-DL [8] to define this ontology.

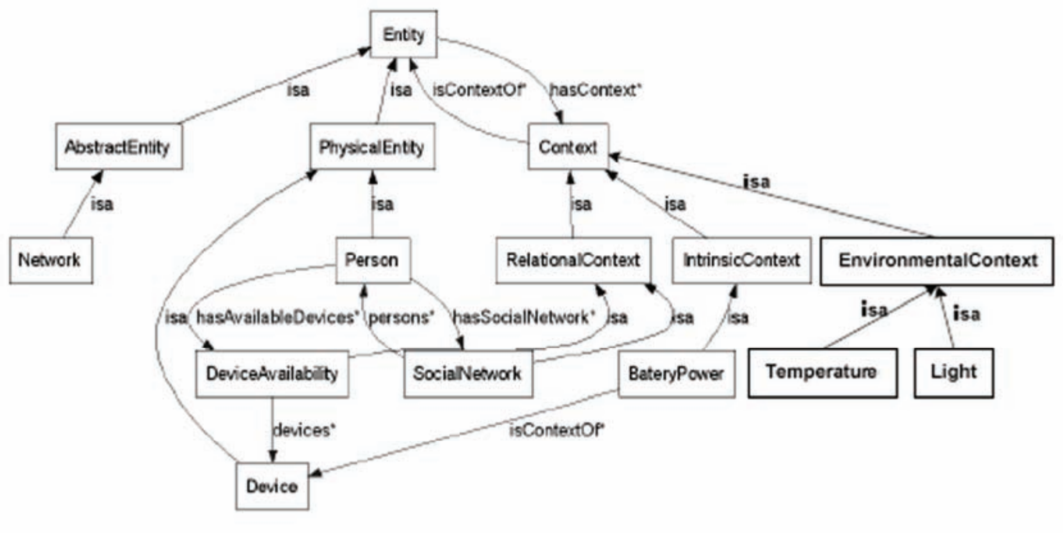


Fig. 2. Selected concepts of the general context model

We distinguish three fundamental categories of context information, namely intrinsic context (`IntrinsicContext`), relational context (`Relational-Context`) and environmental context (`EnvironmentalContext`):

- Intrinsic context defines a type of context information that belongs to the essential nature of an entity and does not depend on the relationship with other entities. An example of intrinsic context is the `location` of a `PhysicalEntity`, such as a `Person` or a `Building`.
- Relational context defines a type of context information that represents a relation between distinct entities. An example of relational context is `Containment`, which defines a containment relationship between entities, such as an entity `Building` contains a set of entity `Persons`. Another example of relational context is `Distance`, which represents the spatial distance between two physical entities.
- Environmental context defines a type of context information that belongs to the physical environment of an entity. Examples are `Light`, `NoiseLevel`, `Pressure` and `Temperature`.

Relational context may be used to relate an entity to the collection of entities that play a role in the entity's context. Examples of relational contexts that can be used for this purpose are `DeviceAvailability`, and `SocialNetwork` (Fig. 2). The `DeviceAvailability` of a person captures the person's current accessible devices and the `SocialNetwork` of a person captures the collection of persons interacting with that person by any communication channels.

The main benefit of this general categorization of context is that it explicitly separates the concepts of entity and context. The relational context type allows us to traverse from an entity to the entities that are related by the relational context. This has enabled us to recursively define the relationship between entity and context, facilitating navigation in the model.

We have extended the general context model to support a tele-monitoring application [2]. In this application, patients' vital signals are processed to detect abnormalities, such as the possibility of having an epileptic seizure, within seconds. Several actions may be taken upon an epileptic alarm, such as contacting volunteers capable of providing first aid and streaming patient's bio-signals to doctors at real time.

We have specialized entity person to `AidPerson`, `Patient` and `HealthCarePerson`. In addition, to accommodate the new types of intrinsic context information presented by the tele-monitoring scenario, we have included `BodyPressure`, `HeartRate`, `hasSeizure`, `isOccupied`, `isAvailable`, `isReachable`, `isAtHome` and `isAtWork`. The first two types of context information (`BodyPressure` and `HeartRate`) are facts, meaning that they are not derived information but can be sensed directly from the environment. The situation `hasSeizure` is derived from the persons `HeartRate` and `BodyPressure` by means of a rather complex algorithm. The situations `isOccupied`, `isReachable`, and `isAtHome` are defined in OWL-DL (using our ontology) as follows:

- `isOccupied` ≡ ∀ `isContextOf (Person` ⊓ `(`∃ `hasContext Activity))`
- `isReachable` ≡ ∀ `isContextOf (Person` ⊓ `(`∃ `hasContext (DeviceAvailability` ⊓ `((`∃ `hasContext ChannelAvailability)))))`
- `isAtHome` ≡ ∀ `isContextOf (Person` ⊓ `(`∃ `hasContext (Containment` ⊓ `(`∃ `container Home))))`

4 ECA Rules

The dynamic aspects of applications, i.e., application behaviors, are defined following the Event-Control-Action (ECA) pattern mentioned previously. We have developed an expressive language that enables the specification of ECA rules. These rules carry enough information to allow the Controller component to autonomously configure the infrastructure (composition of context and action) with no need for further intervention from the application developer. The behavior part of the ECA language we are proposing is based on the situation-based triggering approach presented in [4][5]. We have extended and adapted this approach to satisfy our infrastructure requirements. The information part of the ECA language is based on our context models, which have been presented in the previous sections.

Context changes are described as changes in situation states. There are three possible states (*true*, *false* and *unknown*) and six state transitions. The unknown state accommodates uncertainty of context information. Action invocations are enabled by sequences of transitions and the validation of pre-conditions. The condition part of ECA rules comprises two parts: an event part that defines a relevant situation change, and a pre-condition part that defines a logical expression that must hold following the event and prior to the execution of the action. Both events and pre-conditions are defined in terms of situations and facts. Each rule is associated with a lifetime, which can be `once, from <start> to <end>, to <end>, <n> times` and `frequency <n> times per <period>.`

Events, pre-conditions and actions are prefixed by the clauses `Upon`, `When` and `Do`, respectively. In our approach, we have included the clause `scope` to parameterize an ECA rule. A scope clause defines a collection of entities for which the rule should be applied. We have also included the clause `select`, which returns a collection of entities respecting a given filtering expression[2]. Consider the following ECA rule partially modeling the tele-monitoring scenario:

[2] For the sake of readability, the select clause is described in an OQL-like language, instead of OWL-QL

```
Scope (Select  (entity.patient.*, pat, isIncluded
               (pat.medConditions, epilepsy)))
{
     Upon EnterTrue(pat.hasSeizure)
     When pat.hasSeizure.Accuracy > 50%
     Do critical
          Foreach (Select (entity.patient.aidPersons, aidP,
                          aidP.isAvailable ^
                          aidP.isNear(pat)))
          Contact (aidP)
always
}
```

This ECA rule defines a scope that includes all patients suffering from epilepsy. The function `isIncluded` is part of the standard library to manipulate collections. The `Upon` clause defines a situation state transitions (`EnterTrue`) in which the action should be invoked. The `When` clause defines a pre-condition for the action to be invoked, which is the minimum accuracy required (50%) for the seizure alarm. The `Do` clause defines that the patient's designated aid persons who are near and available (not occupied and reachable) should be contacted on their preferred channel. The term `critical` indicates that the rule should be pre-fetched, meaning that no time should be spent with action lookup requests.

The scope clause defines a dynamic group of epileptic patients. Epileptic patients may enter and leave the system, and the scope clause maintains the actual list of patients, creating and removing rules for each patient that enters and leave the system, respectively.

Conditions are asserted based on information contained in the knowledge base, which is kept up-to-date with context information events originated from the network of Context Sources.

Other examples of notification ECA rules are (in a smart home setting):

Notify all family members (except Bob) that Bob is arriving home.

```
Scope (Select (person.*, person, person.isAtHome &
              person.name <> "Bob"); p))
{
     Upon EnterTrue (Bob.isAtHome)
     When True
     Do Notify (p, "Bob is home")
     Always
}
```

Notify Bob when the average temperature of the house goes beyond 30°C.

```
Upon EnterTrue(house.temperature > 30)
When True
Do Notify (Bob, "House temperature is beyond 30°C")
Always
```

Without the use of the Controlling service, context-aware application developers would have to write programming code to implement the behaviors described above with no automated support for (i) subscribing to context sources, (ii) receiving context event notifications, (iii) performing context reasoning, (iv) implementing a monitoring function to check conditions and (v) implementing action invocations. Our approach facilitates the development process by reducing development efforts and time. However, application developers using the Controlling service are limited to the expressiveness of our language when describing application behaviors.

5 Infrastructure Architecture and Prototyping Activities

Fig. 3 depicts an example configuration of the infrastructure services.

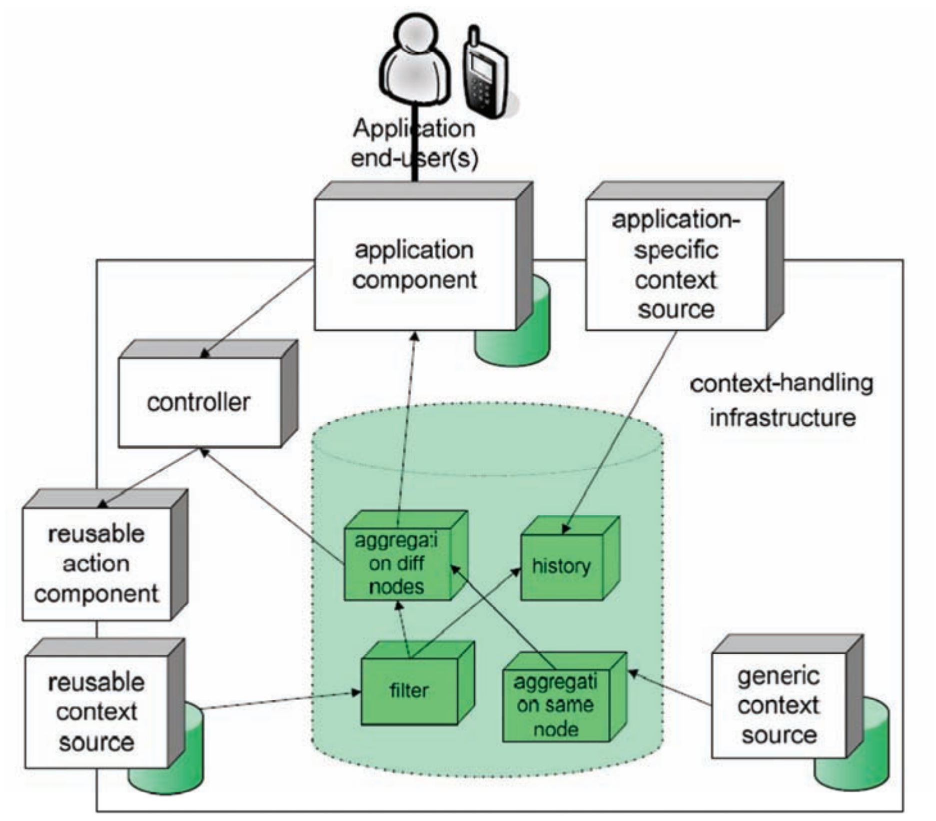


Fig. 3. Example configuration of the context handling infrastructure

Application components may either gather context information directly from the network of context sources or use the service offered by the Controller component. In the latter, applications provide the infrastructure with an application-specific context model and ECA rules, and the Controller makes sure action services and notifications are invoked and delivered appropriately through the reusable action components. Reusable context sources, application-specific context sources and generic context sources are components developed by third-party developers, application developers and the infrastructure, respectively. These components maintain their own specific extensions of the infrastructure general context model, and therefore their own knowledge bases. These particular knowledge bases are represented by database symbols in Fig. 3. The other context processing nodes are dynamically created, according to the layers of reasoning that are required to execute an ECA Rule.

We have prototyped a previous version of this architecture using Web Services technologies and the Java programming language. This prototype has been experimented in various scenarios in the tourism domain [3][11]. It includes a limited version of the Controller and a number of location-based Reusable Context Sources (GPS sensors). The Controller interface is offered as a web service end-point, allowing the operations to be remotely called by the application components. Application components have also been implemented as a web service end-point to allow callbacks from the Controller. We have defined an XML Schema that represents a limited version of the ECA rule language such that application ECA rules can be written as XML documents and validated using this WSL XML Schema. Our ECA rule parser reads application rules in XML format and maps them into Java classes, which are automatically compiled and executed at runtime. We have used an object-based context model and context reasoning is based on hard-coded algorithms. For service discovery, we have used the industry standard UDDI.

Our current prototyping activities include developing a Controller component using the Jess Rule Engine [9]. We are working on the mapping between the ECA language constructs and Jess constructs. In general, a simple ECA rule needs to be mapped to a set of fact assertions and rule definitions in Jess. The mapping of the Scope clause onto Jess rules is particularly complex, since the scope clause requires runtime rule definitions and deletions for each entity entering and leaving the scope, respectively.

Furthermore, we are also working on the runtime creation and distribution of context processing nodes. A newly defined ECA rule may require context reasoning activities that do not exist at the time the rule has been included. When this occurs, the Controller needs to create context processing nodes that are capable of performing the pieces of context reasoning that are required. In order to create such node, the Controller provides (i) the reasoning algorithm itself (derived from the context model); (ii) the type of context input(s) and where this information can be gathered from, and (iii) the type of the expected outputs.

Context processing nodes only exist when they are needed (limited lifetime). There are two types of context processing nodes, namely the stateless and stateful context processing nodes. Stateless context processing nodes perform a piece of reasoning that does not require persistence of context information, while stateful context processing nodes typically maintain history information. Once context processing nodes are running properly, the Controller component is regularly fed with (highlevel) context information provided by both pre-defined context sources and dynamically defined context processing nodes.

6 Related Work

Various approaches to address ECA services for context-aware systems have been proposed. The ECA rule matching service [6] proposes an extension to the standard CORBA Notification services with a composite Event Matching Engine, using CLIPS to implement event correlation and the aggregation process. This approach does not address context information representation, therefore being limited to a predefined set of context types. In addition, there is no support for context information management activities, such as gathering, processing and distribution.

The work presented in [12] discusses a structured framework to design and implement context-processing modules. This work uses ontologies for context information specification and composition. However, as opposed to our approach, no context categorization is provided. Application developers using this framework need to (i) provide a description of context in RDF/XML format and (ii) find and/or implement context processing components. Our approach differs from this work since we take a top-down strategy to perform configuration of context processing modules, which is based on ECA rule descriptions. Application developers provide ECA rules and context models as input, and the Controlling service takes care of dynamically finding and/or implementing context processing components that are required.

The framework presented in [1] proposes a rule-based sentient object model to facilitate context-aware development in an ad-hoc environment. The main functionality is offered in a tool that facilitates the development process by offering graphical means to specify context aggregation services and rules. Although this approach introduces useful ideas on how to easily configure rules and aggregation services on a sentient object, it is based upon a simple model of context that is both informal and lacks expressive power.

The Context Management framework presented in [7] defines a framework and a tool for facilitating end-user customization of context-aware features. This work concentrates on a tool that allows users to combine context and actions in order to define ECA rules. The context and action options provided by the tool reflect the concepts defined in a context ontology. Differently from our approach, there is no use of discovery mechanism to find and match context sources and action providers. Furthermore, there is no strong support for application rules (as opposed to end-user rules) and parameterization of rules. The mechanism presented in this paper focuses on application rules that may be applied to a collection of users. Therefore, as shown in this paper, the use of parameterization through the *scope clause* is important. However, although we present an alternative mechanism to gather context, process rules and invoke actions, our approach would benefit from a user friendly tool such as the one presented in [7].

7 Conclusions

We have discussed in this paper our current efforts towards a flexible context handling infrastructure. A central element of this infrastructure is the Controller component, which takes application-specific rules and context models as input in order to carry out runtime application-specific adaptation within the infrastructure. We have discussed (i) important aspects on context modeling, (ii) a mechanism to define ECA rules, and (iii) a general overview of the infrastructure services. As opposed to various related works [4][10][12], our proposal is based on a top-down approach towards automatic configuration of ECA rules. Based on ECA rules and models of context, the Controller is capable of autonomously configuring the infrastructure accordingly. This approach facilitates the development process, since application developers do not need to write programming code to (i) activate rules; (ii) find and compose context sources; (iii) implement context reasoning activities; and (iv) invoke actions.

We have discussed a generic context model that can be specialized with domain and application specific concepts. This creates application-specific virtual knowledge bases, permitting specific requirements to be addressed in a general Controller architecture. In addition, our context categorization allows us to define context information recursively, which conforms to the recursive nature of context information. For example, it is possible to define that a device is part of a person's context, and that communication channels are part of the device's context, and so forth. Our context model allows us to traverse from an entity to the entities that are related to this entity by the relational context. This has enabled us to recursively define the relationship between entity and context, facilitating navigation in the model.

We have defined an ECA language to specify rules that are used by the Controller to create and compose context processing nodes. This language allows us to specify ECA rules that consider a scope in which these rules should be applied, as opposed to cumbersomely defining an individual rule for each entity instance.

By using the Event-Control-Action pattern we have decoupled context concerns from action concerns, under the control of application-specific rules, enabling effective distribution of responsibilities among various parties within the infrastructure. This approach has greatly improved extensibility and flexibility of the infrastructure's generic functionality, since rules, actions and context information can be added on demand, at infrastructure runtime.

As part of our ongoing research, we are developing additional context-aware applications with the support of the infrastructure's Controlling service. We are also investigating effective approaches to distribute Controller components, while tackling synchronization of rules and potentially conflicting issues.

References

1. Biegel, G., and Cahill, V.: A Framework for Developing Mobile, Context-Aware Applications. In: Proc. of the 2nd IEEE Conference on Pervasive Computing and Communications (Percom2004). USA (2004) 361-365
2. Dockhorn Costa, P., Ferreira Pires, L., van Sinderen, M.: Architectural Patterns for Context-Aware Services Platforms. In: Proc. of the Second International Workshop on Ubiquitous Computing (IWUC 2005 at ICEIS 2005). USA (2005) 3-19
3. Dockhorn Costa, P., Ferreira Pires, L., van Sinderen, M.: Designing a Configurable Services Platform for Mobile Context-Aware Applications. In: International Journal of Pervasive Computing and Communications (JPCC), 2005, Troubador Publishing
4. Henricksen, K., and Indulska, I.: A Software Engineering Framework for Context-Aware Pervasive Computing. In: Proc. of the 2nd IEEE Conference on Pervasive Computing and Communications (Percom2004). USA (2004) 77-86
5. Henricksen, K.: A Framework for Context-Aware Pervasive Computing Applications. PhD thesis, School of Information Technology and Electrical Engineering, The University of Queensland (2003)
6. Ipina, D., and Katsiri, E.: An ECA Rule-Matching Service for Simpler Development of Reactive Applications. In: Proc. of Middleware 2001 at IEEE Distributed Systems Online, Vol. 2, No. 7, November 2001
7. Korpipää, P., Malm, E., Salminen I., Rantakokko, T.: Context Management for End User Development of Context-Aware Applications. In: Proc. of the 6th International Conference on Mobile Data Management. Cyprus (2005) 304-308
8. McGuinness, D., and van Harmelen, F.: OWL Web Ontology Language – Overview, W3C Recommendation (2004). Available at http://www.w3.org/TR/owl-features/
9. Jess – the Rule Engine for the Java Platform. Available at herzberg.ca.sandia.gov/jess/
10. Przybilski, M.: Distributed Context Reasoning for Proactive Systems. In: Floreen, P., et al. (eds.): Proc. of the Workshop on Context Awareness for Proactive Systems (CAPS 2005). Finland (2005) 43-54

11. Pokraev, S., Koolwaaij, J., van Setten, M., Broens T., Dockhorn Costa, P., Wibbels, M., Ebben, P., Strating, P.: Service Platform for Rapid Development and Deployment of Context-Aware, Mobile Applications. In: Proc. of International Conference on Webservices (ICWS'05). USA (2005)
12. Sbodio, M. and Thronicke, W.: Specification and Design of Framework-Based Context Processing Modules. In: Floreen, P., et al. (eds.): Proc. of the Workshop on Context Awareness for Proactive Systems (CAPS 2005). Finland (2005) 79-92

Proactive Hoarding in Location-Based Systems

Susanne Bürklen, Pedro José Marrón, and Kurt Rothermel

University of Stuttgart
Institute of Parallel and Distributed Systems
Universitätsstr. 38
70569 Stuttgart, Germany
{buerklen, marron, rothermel}@ipvs.uni-stuttgart.de

Abstract. The proliferation of mobile devices and the fact that high-bandwidth connectivity is not available everywhere, has led to the development of hoarding algorithms. The aim of these algorithms is to select and prefetch data, mobile users might access when they are weakly connected in the future. The data is selected based on context information of users, such as their location. In this paper, we present a hoarding approach for semi-structured information items and introduce our *Bounded Path Search* algorithm (BPS) for the computation of the hoard list. We show by means of experimental evaluation that BPS outperforms existing hoarding techniques that use standard graph search algorithms by a factor of 2.7 in terms of hoard cache hit ratio.

1 Introduction

With the increasing popularity of mobile devices, users want to be able to access information anytime and anywhere. Wireless communication technology such as 2G and 3G cellular networks can provide this in large parts. However, even 3G networks like UMTS provide only moderate data rates. Moreover, communication channels may be subject to substantial error rates, high latency and frequent disconnections. A complementary technology is based on wireless LAN (WLAN) hotspots, which are isolated ”islands” of connectivity. They provide comparatively cheap or sometimes even free high-speed Internet access to mobile users within a limited area.

In order to optimize mobile data access, both technologies can be combined. More precisely, mobile devices can use the available connectivity at a given hotspot to prefetch and locally cache data wherever possible. This approach, called hoarding, is a proactive prediction technique that aims at minimizing wireless WAN usage by prefetching all the data items a user is going to request next. Moreover, it also supports disconnected operation since (a portion of) the data requests can be served from the cache. For the purpose of this paper, we use the term *infostations* to refer to hotspots that provide hoarding functionality. In contrast to caching, the goal of hoarding is to fetch a data item into the cache *before* the first access occurs. Therefore, it is necessary to predict which data items will be needed next. This prediction, called hoarding decision, may be done by the user himself telling the system which items to hoard, or it may be done automatically by the system exploiting different types of information, such as user interests or location-dependency and syntactic/semantic relationship of data. In this paper, we will present an automatic hoarding mechanism that assumes data items to be location-dependent. Moreover, the algorithm takes into account the semantic distance [8] of data items, which is derived from the users' navigation behavior in a semi-structured information space. With this analysis we can leverage relationships among data items. Such a relationship can be a (logical) link, which indicates that the linked items were accessed subsequently in a user session. In [1], we showed that considering link popularity improves hoard cache hit ratio up to 25% compared to the unstructured scheme presented in [2], where only data popularity was considered. The hoarding approach proposed in this paper is based on our Bounded Path Search algorithm (BPS) and the computation of a relevance value for each data item by taking into account its access probability and size. Although this approach has been designed for mobile Web access, it is generic enough to be also used for other semi-structured information spaces. In our evaluations, we compare the results of BPS with the results when using standard graph search algorithms. We show that our approach outperforms these schemes by a factor of up to 2.7 when considering hit ratio as the performance metric.

The research reported in this paper was conducted as part of Nexus [3, 4], a Center of Excellence at University of Stuttgart that deals with the design and implementation of a platform for mobile context aware applications.

The remainder of this paper is structured as follows. After discussing related work in Sect. 2, we give a short overview of our hoarding approach in Sect. 3. We introduce the concept of an information graph in Sect. 4, which is used by infostations to represent the access behavior of users. In Sect. 5, we discuss BPS and the steps involved in computing the data to hoard. In Sect. 6, we discuss the experimental results of our evaluations. Finally, we conclude with a short summary and an outlook on future work in Sect. 7.

2 Related Work

The goal of hoarding is to prefetch data before it is accessed the first time, which differs from what caching schemes like [5] aim at. Hoarding is not an issue in those approaches. Broadcast disk approaches like [6, 7] also analyze the access behavior of users. If the mobile devices involved in the system are location-aware, users are provided with the information items they need at their current location. However, location-aware broadcast-based dissemination systems do not predict the information items users will need after leaving for another location away from the coverage area of the current cell. Furthermore, these systems use this information to define broadcast schedules that minimize response time.

The SEER filesystem [8] incorporates a hoarding mechanism, which automatically selects the files to hoard. The authors of [9] present the architecture for a mobile file service for ubiquitous computing, which also supports automatic file hoarding. However, these approaches are based on the characteristics of filesystems and their operations, which do not fit the needs for semi-structured data such as Web pages.

In [2] a hoarding mechanism for location-dependent data is proposed. However, this scheme was designed for unstructured information spaces. Our previous hoarding scheme proposed in [1] supports semi-structured data, such as the Web, but uses standard graph search algorithms for building the hoard list without computing a relevance value. In our evaluations, we will compare our work with this scheme.

Approaches such as [10] combine the benefits of hoarding and cooperative caching methods. The idea is to increase the hoarding space by letting mobile devices share their hoard caches. This research focuses more on distributed cache management issues rather than the processing of the hoarding decision.

3 Assumptions and Overview

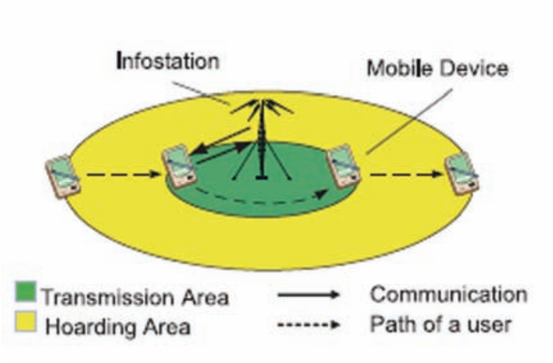


Fig. 1. System Model

As depicted in Fig. 1, our system model comprises *infostations* and *mobile devices*. An infostation is a stationary server connected to the Internet and equipped with a WLAN access point. We use the term *transmission area* to refer to the transmission range of the access point and *hoarding area* to the geographic region the infostation is responsible for in terms of analyzing user access patterns. Typically the hoarding area is much larger than the transmission area. We assume that a mobile device, such as a PDA or smart phone, is equipped with WLAN capabilities and possibly with wireless WAN connectivity.

An infostation keeps track of the aggregated Web access behavior of users within its hoarding area. Based on this information, the infostation predicts which pages are most relevant for current users and updates their hoard cache as explained in the next sections. This approach is based on the assumption that the requested data is location-dependent, that is, the location of the user has an influence on the preference for specific data items. The main idea is then to use the gathered information about access behavior to supply users with those pages they might be interested in while they are in the hoarding area.

Typically there will be many infostations spread over a region like a town. To ensure complete support for hoarding, each hoarding area should overlap with at least one transmission area of another infostation. In this paper, we concentrate on the operation of a single infostation.

Figure 2 depicts the overall process of our hoarding algorithm. A mobile device runs a browser allowing the user to navigate the Web. In addition, it maintains a *hoard cache*, which stores the Web pages downloaded from the most recently visited infostation. When a user requests a page, the *hoard cache manager* checks whether or not this page is stored in the hoard cache. If the page is cached, it can immediately be returned to the user. In case of a hoard miss, the page may be accessed over the wireless WAN, provided the device is connected. Otherwise, the miss is signalled to the application.

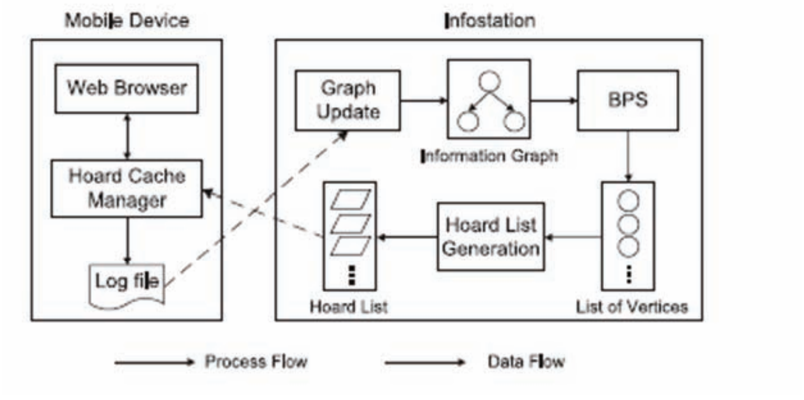


Fig. 2. Components of our Hoarding Approach

In order to record user demands, a mobile device maintains a *log file*, which is used to keep track of the locally requested Web pages. Whenever a page is requested, a record is added to the log file containing the URL and size of the page, as well as the time and location of the request. In order to determine the location of Web page requests, we assume that each mobile device can determine its position. The location information associated with a request should allow an infostation to determine in which hoarding area the request was issued. Therefore, GPS accuracy, for example, is fully sufficient. Even cell IDs can be used if a hoarding area is defined as a set of cells. Of course, in this case cellular WAN connectivity is mandatory.

When a mobile device moves into the transmission area of an infostation, it uploads its log file and hoard cache size to the corresponding infostation, which uses the log file to update its *information graph*. An information graph models the aggregated browsing behavior of all the users in the hoarding area and includes information about the frequency of requests, which pages have been requested, how they relate to each other and the location of requests. As part of the *graph update* procedure the infostation divides the uploaded log file into *browsing sessions*, defined as sequences of temporally related page requests. After traversing the graph using BPS, the *hoard list generator* assigns a relevance value to each of the vertices that depends on the probability of the corresponding page being accessed and its size. Finally, based on this relevance, we compute an ordered list of Web pages, which we call a *hoard list*.

Graph traversal and hoard list generation is done only periodically due to performance reasons, while graph update and hoard list download is performed whenever a log file is uploaded.

Let us now discuss each of the algorithms in the infostation in more detail.

4 Graph Update

The process of updating the information graph involves two steps: (1) the analysis of the log file, which includes the computation of session boundaries and (2), the actual information graph modification.

Table 1. Sample User Log File with two Sessions and Derived Information

User Log File				**Derived Information**	
URL	**loc**	**time**	**size (Bytes)**	**period (s)**	**session**
A	(1.0,2.0)	2005-06-20/12:00:00	1512	10	Session 1
B	(1.0,2.0)	2005-06-20/12:00:10	1432	20	
D	(5.0,7.0)	2005-06-20/12:00:30	1510	960	
E	(1.0,2.0)	2005-06-20/12:16:30	2312	10	Session 2
F	(1.0,2.0)	2005-06-20/12:16:40	1460	75	
T	(1.0,2.0)	2005-06-20/12:17:55	1643	15	
F	(1.0,2.0)	2005-06-20/12:18:10	1856	10	
J	(1.0,2.0)	2005-06-20/12:18:20	1746	15	
I	(1.0,2.0)	2005-06-20/12:18:35	2684	–	

4.1 User Log File Analysis

For the first step, we consider a user *log file* $L = \{l_1, \ldots, l_{i-1}, l_i, l_{i+1}, \ldots, l_n\}$ to be a totally ordered set of log entries sorted with respect to the individual request time using the order relation $l_i < l_{i+1} \Leftrightarrow l_i.\mathrm{time} < l_{i+1}.\mathrm{time}$.

A user log file is composed of log entries, which are defined as follows. A *log entry* is a tuple $l = \langle \mathrm{URL}, \mathrm{loc}, \mathrm{time}, \mathrm{size} \rangle$, where $l.\mathrm{URL}$ is the URL of the requested Web page; $l.\mathrm{loc}$ is the geographical location where the entry was recorded (e.g., coordinates x and y, defined by a geographical coordinate system); $l.\mathrm{time}$ is the date and time of the request; and $l.\mathrm{size}$ is the size of the requested page in bytes.

Table 1 shows an example of a log file with Web pages requested within a particular hoarding area. The first entry is the request of page A of size 1512 bytes, requested at time 12:00:00 on June 20th, 2005 at location (1.0, 2.0).

Using the information contained in this log file, as part of the log file analysis step, we derive the visit period and the beginning and end of sessions.

The *visit period* is defined as follows. Let l_{i-1} and l_i be two consecutive log entries in the log file. Then $l_{i-1}.\mathrm{period} = l_i.\mathrm{time} - l_{i-1}.\mathrm{time}$ is the visit period of the page requested in l_{i-1}. In our example of Table 1, data item A has a visit period of 10s.

We then detect the boundaries of sessions contained in the log file. A session is a subset of consecutive log entries in a log file, where the visit period of each page falls below a threshold value. For our experiments and following the suggestions in [11], we have set the threshold value $t_{\mathrm{threshold}}$ to 15 minutes for the maximum visit period of a page within a session.

In our example of Table 1, using the aforementioned values, we obtain two sessions separated by the double horizontal line. Session 1 contains pages A, B and D and Session 2 contains the remaining pages.

The location $l_1.\mathrm{loc}$ associated with the Web page starting a session determines which hoarding area(s) the session belongs to. Each session belonging to the same or a different hoarding area is forwarded by the analyzing infostation to the responsible one(s). Due to space constraints, we do not describe the collaboration among infostations further in this paper.

4.2 Information Graph Modification

Each infostation maintains an information graph that encodes the information received as part of the user log file into a graph where the vertices represent the individual Web pages and the edges contain information about the navigation patterns of users.

More formally, the information graph IG is a tuple $IG = (V, E, g, B, z)$, where V is the set of vertices representing Web pages; E is the set of edges; g

is the root vertex; $B \subseteq V$ is the set of session start vertices; and z is the accumulative vertex used for bookkeeping of sessions and user revisit patterns.

Each vertex $v \in V$ has attributes v.URL, and v.size, that contain the URL and size of the page. Each edge $e \in E$ contains a counter e.traversal that keeps track of how many times its source and sink vertices have been requested consecutively over the lifetime of the information graph.

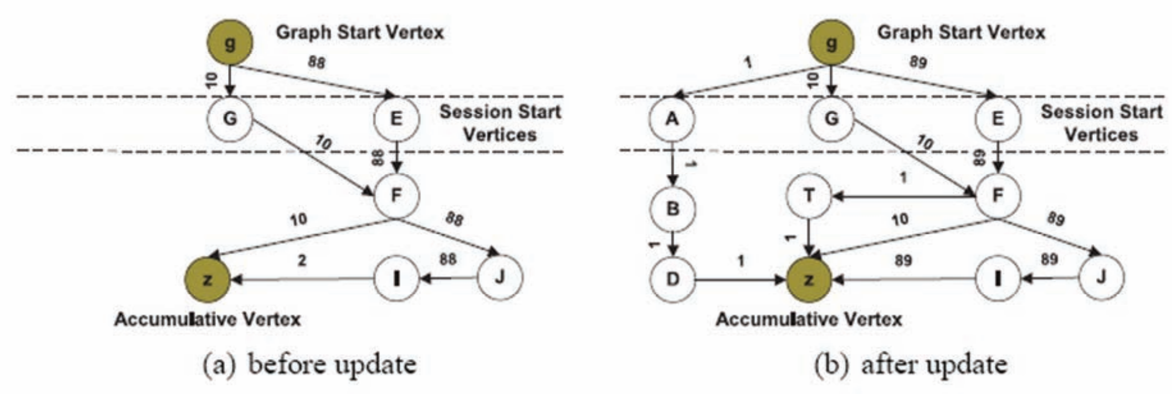


Fig. 3. Instances of Information Graph

Figure 3(a) contains an example of an information graph where the two special vertices g and z as well as 5 other vertices can be seen. The numbers at the edges represent the values of the e.traversal counter. For example, the number 10 associated with edge (G, F) indicates that vertex F has been requested ten times immediately after G.

Using the information graph and the uploaded log file, for each session $L_s = \{l_1, . . . , l_n\}$ identified during the log analysis step, the update algorithm increments the traversal counter of an edge e according to the following cases. For the ease of exposition, we assume that a log entry l_i has a corresponding vertex v_i in the information graph.

1. If l_i is a session start page, update the edge $e = (g, v_i)$.
2. If l_i is a session end page, update the edge $e = (v_i, z)$.
3. For all other entries l_i with predecessor l_{i-1}, update $e = (v_{i-1}, v_i)$ if l_i appears in the session for the first time. Otherwise, update $e = (v_{i-1}, z)$.

Following our running example, consider Session 2 in the log file of Table 1 composed of the requests E, F, T, F, J, I. Then, our algorithm performs the following changes. Based on the three cases described above, we update the traversal counter of edge $e = (g, v_E)$ from 88 to 89, since E is a session start page, and the counter of edge $e = (v_I, z)$, since I is a session end page. Other edges such as $e = (v_E, v_F)$ are also updated accordingly and for vertices where there is still no edge available, such as $e = (v_F, v_T)$ or $e = (v_T, z)$, we create the edge and initialize its traversal counter to 1. The final result of the update algorithm is depicted in Fig. 3(b).

Following the algorithm we have just described, the information graph adapts slowly to changing user behavior. In fact, the ability to adapt to new input becomes even worse over time since past and current page accesses are counted equally. Therefore, we integrate an *aging mechanism*, which divides time into epochs with a fixed length Δt. Whenever an epoch ends, we compute a smoothed value for each traversal counter using an exponentially weighted moving average function.

Therefore, we additionally maintain an auxiliary counter which takes care of recording the number of requests and traversals in the current epoch. When an epoch ends, we compute the value of the traversal counter using the following formula and then reset the auxiliary counter:

$$e.\text{traversal} = \delta \cdot e.\text{traversal} + (1 - \delta) \cdot e.\text{aux\_traversal} \qquad (1)$$

The actual implementation of our algorithm integrates this aging mechanism with the search algorithm described below so that no extra graph traversal is required.

5 Bounded Path Search Algorithm (BPS) and Hoard List Generator

At the end of an epoch, the infostation traverses the information graph using BPS. The result is a list of vertices with statistical information such as their access probability and size. The hoard list generator ranks the vertices and builds a hoard list containing the Web pages associated with the sorted vertices.

BPS is a modified version of the Bounded Depth First Search (B-DFS) algorithm. It extends B-DFS in two aspects. Firstly, vertices may be visited multiple times to potentially traverse all paths in the information graph. Secondly, the traversal is bounded in terms of the access probabilities associated with the individual paths.

Before describing BPS, we have to introduce the terms edge probability and path probability. Let $e = (u, v)$ be the edge connecting vertices u and v. The edge probability $p(e)$ represents the probability that a user will request $v.URL$ immediately after $u.URL$. More formally, the metric is defined as follows. Let $E_{out(u)}$ be the set of edges outgoing from u. Then,

$$p(e) = \frac{e.traversal}{\sum_{e' \in E_{out(u)}} e'.traversal} \qquad (2)$$

For a path from vertex $v \in V$ to the root vertex g, we define its probability $p(path_v)$ as the product of the probability of each individual edge that forms the path as

$$p(path_v) = \prod_{e \in path_v} p(e) \tag{3}$$

The path probability $p(path_v)$ corresponds to the access probability of the page represented by v.

As mentioned above, BPS potentially traverses all paths. In order to avoid cycles, each vertex maintains a mark which is set the first time this vertex is visited in a rootto-leaf direction. The mark is removed during backtracking, which is initiated in three cases: (1) The probability of the path to the following vertex is smaller than a threshold value. (2) The accumulative vertex z is reached. In this case, a session ended or a user revisited pages, respectively. (3) A marked vertex is reached.

Note that the *path probability threshold* is an important parameter for limiting the computation overhead. BPS returns the list of vertices (i.e., URLs), whose access probability is above this threshold. Each vertex is associated with statistical information, such as its access probability and size.

The final step involves the *generation of the hoard list*. The aim of hoarding is to maximize the number of cache hits when a mobile user requests Web pages. Since the hoard cache is limited, we consider the access probability per byte rather than per page when computing the relevance $r(v)$ of vertex v as follows

$$r(v) = \frac{p(path_v)}{v.size} \tag{4}$$

In our running example, $r(E) = \frac{0.89}{2312} \approx 3.8 \cdot 10^{-4}$ and $r(J) = \frac{0.89 \cdot 0.89}{1746} \approx 4{,}5 \cdot 10^{-4}$. Even though $p(path_J)$ is smaller than $p(path_E)$, we rank J higher than E since J occupies less hoard cache than E.

The hoard list generator ranks the vertices according to their relevance. For hoard list download, the high-ranked portion of the hoard list fitting in the client's hoard cache is selected for download.

6 Evaluation

In this section, we describe the methodology of our evaluation, in particular the log files we have used and our metric. We then discuss our experimental results.

6.1 Methodology and Experimental Setup

Our evaluation is based on an implementation of BPS and a large volume of synthesized log data. Extracting the needed information from real-world (proxy) server log files does not usually contain all the information needed to properly create hoarding log files, especially if they have been made anonymous due to privacy concerns. Therefore, we decided to generate synthetic log files based on the Web Browsing Model described in [12].

We have generated 20.000 user log files, from which we analyzed 15.000 to update the information graph and used the remaining 5.000 for evaluation purposes. The information graph represents the Web pages maintained by an infostation. Since the requested Web pages are assumed to be location-dependent, the infostation maintains only a small portion of the Web. The resulting information graph consists of about 10.000 vertices, 47.000 edges, and 5.600 session start vertices. The size of all Web pages is about 175 MBytes, in the range between 1.8 KBytes and 20 MBytes.

We set the value of parameter δ for the ageing mechanism in (1) to 0.5 after having performed extensive experiments to determine this value. The evaluation of these experiments is out of the scope of this paper.

6.2 Experimental Results

From a user's perspective, the major performance parameter of a hoarding algorithm is the hit ratio h, which is defined as $h = \frac{|C|}{|L|}$, where $|L|$ is the number of entries in the log file and $|C|$ is the cardinality of a set C that contains those Web pages found in the hoard-cache. Obviously, the hit ratio strongly depends on the available hoard cache size. From the infostation's perspective, the major performance parameter is the path probability threshold to reduce runtime.

We have compared the performance of BPS with approaches using standard breadth first search (BFS), standard depth first search (DFS) and bounded DFS modified to traverse the edges based on the edge probability as defined in (2). Furthermore, we have compared the hit ratios of BPS achieved using different values for the path probability threshold w_{pth}.

Figure 4 depicts the hit ratios comparing BPS, BFS, DFS, and B-DFS with boundaries 5, 10, and 20, in the chart denoted as DFS-5, DFS-10, and DFS-20, respectively. As we can see, BPS outperforms the remaining search algorithms in all cases. The hit ratio increases rapidly to 64.8% for hoard cache sizes up to 10 MBytes and increases steadily for hoard cache sizes greater than 30 Mbytes (which is about 16% of the graph size) from 88.2% up to a value of 96.8%. This is due to the fact that we hoard those pages first that have the highest relevance as defined in (4). Therefore, the slope of this curve decreases steadily, since we include the page size in the relevance, as opposed to the remaining search algorithms.

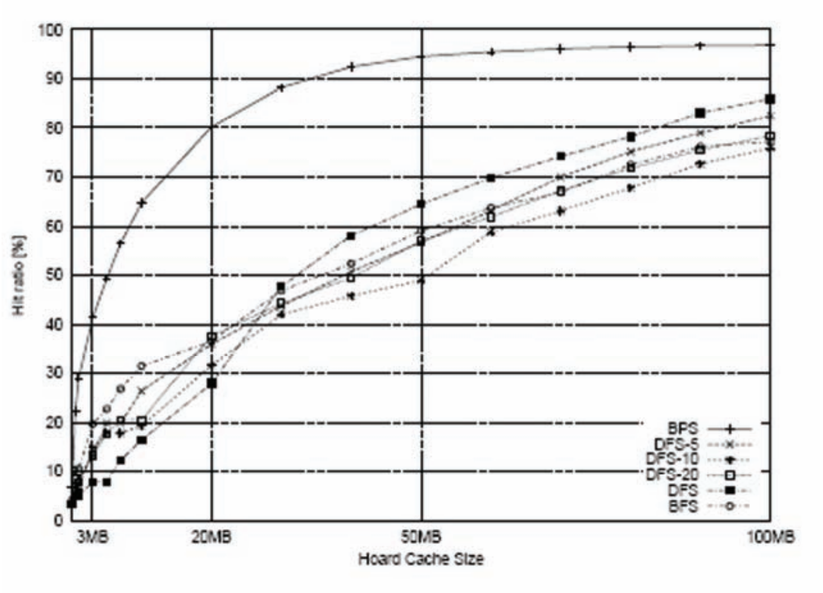


Fig. 4. Comparing BPS to Standard Graph Search Algorithms

For hoard cache sizes up to 30 MBytes, BFS performs best, whereas the hit ratio decreases with increasing boundaries for B-DFS. This is due to the fact BFS and DFS-5 provide more shorter sessions. For hoard cache sizes greater than 30 MBytes, DFS performs best, since now popular paths are traversed completely. In this case, it is not reasonable to limit DFS, since traversal is stopped unnecessarily.

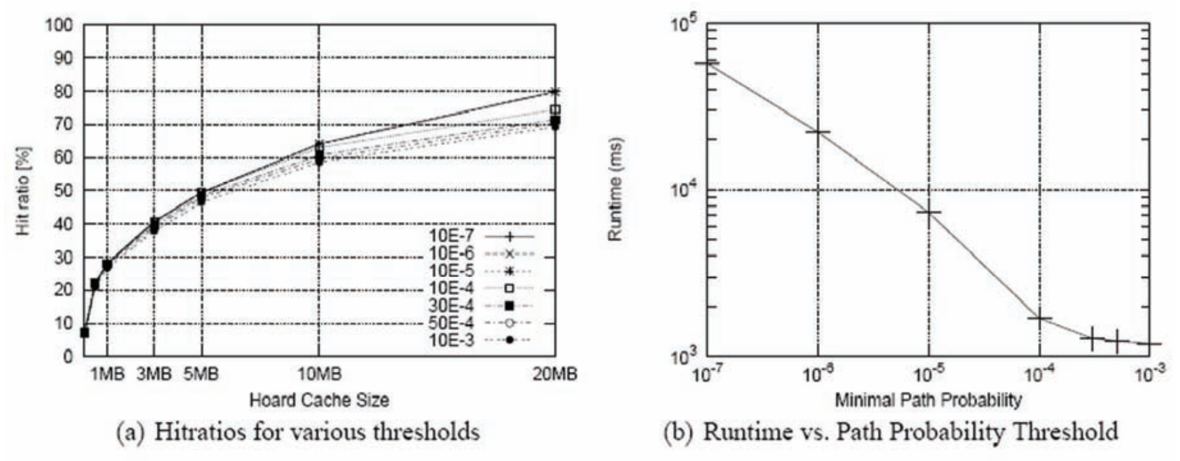


Fig. 5. Runtime Performance using Various Path Probability Thresholds

Figure 5(a) depicts the comparison of hit ratios depending on the hoard cache size using threshold values w_{pth} varying from 10^{-7} to 10^{-3}. As we can see, the curves representing the hit ratios for w_{Pth} from 10^{-7} to 10^{-5} overlap. At the same time, the execution time for these values decreases rapidly from about 58 seconds to about 7.3 seconds, as depicted in Fig. 5(b).

In the near future, we expect a considerable growth of cache sizes. However, we expect that in the same way the sizes of information items increase (e.g., video or music files). We assume that in this case we can achieve similar results for the same ratio between the size of data available at the infostation and hoard cache sizes.

7 Conclusion and Future Work

In this paper we have introduced a novel approach for hoarding location-dependent semi-structured information items, based on our *Bounded Path Search Algorithm* (BPS). We have shown by means of experimental evaluation that BPS works better than approaches that use standard graph search algorithms for deciding which items to hoard. Finally, we have evaluated BPS with several path probability thresholds, in order to determine the values that provide us with the best results and, at the same time, with an acceptable runtime performance.

Next, we plan on providing a greater level of granularity for each infostation by dividing the hoarding area into zones, as done in [2]. For this purpose, we will not only use Web pages, but also other type of structured information whose location-dependency plays a more important role than that of Web page data.

References

1. Bürklen, S., Marrón, P.J., Rothermel, K.: An enhanced hoarding approach based on graph analysis. In: Proceedings of the IEEE International Conference onMobile Data Management (MDM 2004), Berkeley, CA, USA. (2004) 358–369
2. Kubach, U., Rothermel, K.: Exploiting location information for infostation-based hoarding. In: Proc. of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom 2001), ACM Press (2001) 15–27
3. Nicklas, D., Großmann, M., Schwarz, T., Volz, S., Mitschang, B.: A model-based, open architecture for mobile, spatially aware applications. In Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J., eds.: Proceedings of the 7th International Symposium on Spatial and Temporal Databases (SSTD 2001), Redondo Beach, CA (2001)
4. Hohl, F., Kubach, U., Leonhardi, A., Rothermel, K., Schwehm, M.: Next century challenges: Nexus – an open global infrastructure for spatial-aware applications. In Imielinski, T., Steenstrup, M., eds.: Proceedings of the Fifth

Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1999), Seattle, Washington, USA (1999) 249–255
5. Ren, Q., Dunham, M.H.: Using semantic caching to manage location dependent data in mobile computing. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom 2000), ACM Press (2000) 210–221
6. Davies, N., Cheverst, K., Mitchell, K., Friday, A.: Caches in the air: Disseminating information in the guide system. In: Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications. (1999) 1–19
7. Tan, K., Ooi, B.: Data Dissemination in Wireless Computing Environments. Kluwer Academic Publishers (2000)
8. Kuenning, G.H., Popek, G.J.: Automated hoarding for mobile computers. In: Proceedings of the sixteenth ACM Symposium on Operating Systems Principles, ACM Press (1997) 264–275
9. Zhang, J., Helal, A.S., Hammer, J.: Ubidata: ubiquitous mobile file service. In: Proceedings of the 2003 ACM Symposium on Applied Computing, ACM Press (2003) 893–900
10. Kwong Lai and Zahir Tari and Peter Bertok: Supporting Disconnected Operations Through Cooperative Hoarding. In: Proc. of International Conference on Computer Communications and Networks (ICCCN). (2005)
11. Arlitt, M., Krishnamurthy, D., Rolia, J.: Characterizing the scalability of a large web-based shopping system. ACM Transactions on Internet Technology 1(1) (2001) 44–69
12. Bürklen, S., Marrón, P.J., Fritsch, S., Rothermel, K.: User centric walk: An integrated approach for modeling the browsing behavior of users on the web. In: Proceedings of the 38th IEEE Annual Simulation Symposium (ANSS'05), San Diego, CA, USA. (2005)

Context-Agnostic Learning for Contextual Recommendations

Christian Räck, Bernd Mrohs, Stefan Arbanowski, Stephan Steglich

Fraunhofer Institute for Open Communication Systems (FOKUS)
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
`<firstname>.<lastname>@fokus.fraunhofer.de`

Abstract. Identifying correlations between context data, user behavior, and semantic information can lead to new services that are able to adapt to different situations. This “personalization” process can be based on recommendations on content. To better support service developers in focusing mainly on the creation of their service logic, these recommendations should be provided by a generic multipurpose recommender. Therefore, this paper proposes a generic framework that delivers “contextual recommendations”, which are based on the combination of previously gathered user feedback data (i.e. ratings and clickstream history), context data, and ontology-based content categorization schemes. This paper provides a detailed overview of the specification, a short description of a possible usage scenario, and a discussion of the results.

1 Introduction

The World-Wide Web (WWW) provides access to more than 70 million web sites in the beginning of the year 2006 [1]. On a wide range of topics the WWW has become the primary source of information. At the same time it has become increasingly difficult to find what we are looking for [2]. Today, search engines are used by most of us to address this issue.

These search engines allow us to find all information such as web pages, products, or news that matches explicit queries rather easily. Whereas queries, which are based on fuzzy keywords, lead to large lists of mostly irrelevant search results. If we do not know much about the information we expect to find in advance, most of us will have difficulties specifying the required queries.

Search engines start to be accompanied by other technological means to address the above mentioned shortcomings. Personal characteristics, which have been learnt over time, help to improve search results by eliminating irrelevant results based on individual interests. These recommender systems attempt to find web pages, products, or news that are relevant for us. Their recommendations are based on data about our past behavior or statistical models. The resulting data, which is used to make recommendations, is stored in a user profile. Kobsa [3] and Stewart [4] provide a good survey of user modeling techniques and De Roure [5] provides a review of recommender systems.

The learning algorithms and prediction methods used by today's recommender systems are strictly tailored to the service for which the system was designed. Moreover, these systems abstract from the prevailing situation during the learning process. This means that they could not provide reasonable recommendations, if the user employs the same service in different situations (such as '*being at home*' or '*being at work*') and acts differently.

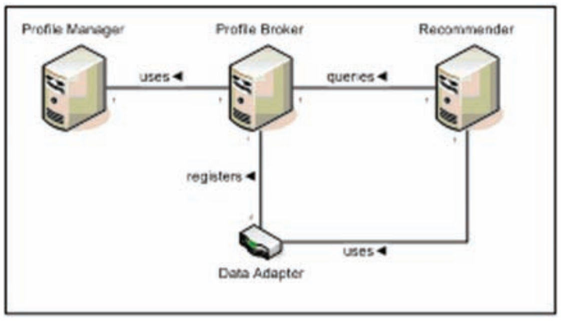


Fig. 1. AMAYA recommender system

This paper presents a current overview of the on-going work on the AMAYA recommender system. At the current stage of our work the focus lies on the analysis and refacturing of the applied approach, i.e. that there are only preliminary results on the quality of the used learning algorithms and prediction methods available.

2 Related Work

Content-based recommender systems base their recommendations on the similarity between new items and items that a user has liked before (user-to-item). Examples of content-based recommender systems are Fab [6] and ELFI [7]. Fab recommends web pages and ELFI recommends funding information from a database.

Collaborative recommender systems derive their recommendations on the basis of content ratings from people with similar interests (either user-to-user or item-to-item). PHOAKS [8] and Group Lens [9] are examples of such recommender systems. PHOAKS recommends web links found in newsgroups. Whereas the Group Lens system recommends single newsgroup postings.

Calvin [10] is a *personal information agent recommender* that monitors the user's browsing behavior over time. It learns to identify broader contexts by relating documents that tend to be accessed together. Calvin extracts information about how the user tends to access groups of documents over time, and predicts document relevance based on similarity of the extended sessions within which various documents are accessed.

3 A System for Context-Agnostic Learning

The AMAYA recommender system is built on two basic requirements: The first requirement is that the system must **provide recommendations for all situations** that might arise in a user's life. And the second requirement is that the system must **support any number of services** a user might want to use. The first requirement will be met by mapping all personalization data to specific situations in order to be processed by the system. The second requirement will be met by providing a generic interface to a number of learning algorithms and prediction methods as well as introducing an ontology-based content categorization scheme.

AMAYA consists of four functional components: the DATA ADAPTER subsystem, the PROFILE MANAGER subsystem, the PROFILE BROKER subsystem, and the actual RECOMMENDER subsystem.

A DATA ADAPTER enables the retrieval and processing of distributed personalization data by providing an interface that abstracts from the actual technology used to store the data. The PROFILE MANAGER groups the personalization data in terms of profiles, which provide a mapping to one or more specific situations. The PROFILE BROKER allows querying of all personalization data that is needed in a specific situation. The RECOMMENDER supports the learning of contextual preferences by providing a generic interface to multiple learning algorithms and prediction methods. These algorithms still prevail from the situation during the learning process. However, all learnt models are linked to the current situation during the learning process. The contextual retrieval of these models enables us to provide recommendations for specific situations. Together these four components provide the required functionality enabling **contextual preferences management** and **contextual recommendations**. Räck [11] provides a detailed overview of the AMAYA approach. Figure 1 shows the basic architecture of the AMAYA recommender system.

4 System Design: Data Model

The AMAYA data model consists of three major objects: ENTRY, ENTITY, and PROFILE. It shows how any kind of personalization data can be described.

4.1 Class: ENTITY

ENTITY objects manage all basic information about users or groups. Each ENTITY is identified by a URI. In addition to that, aliases can be defined in order to enable users to address the record in a more convenient way. This record contains descriptive information (e.g. full name, a description) about that entity and acts as a starting point for the retrieval of all other profile information. Figure 2 shows the corresponding class diagram.

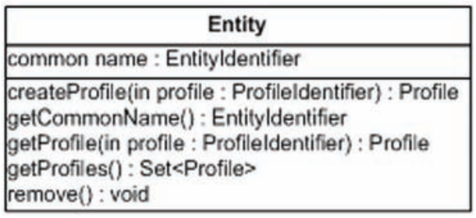

Entity
common name : EntityIdentifier
createProfile(in profile : ProfileIdentifier) : Profile
getCommonName() : EntityIdentifier
getProfile(in profile : ProfileIdentifier) : Profile
getProfiles() : Set<Profile>
remove() : void

Fig. 2. 'Entity' class diagram

4.2 Class: Entry

A single unit of personalization data is called ENTRY. An ENTRY object is uniquely identified by its distinguished name [12]. The distinguished name is stored as a URI. In addition to its common name, each ENTRY object stores a value or a list of values. Figure 3 shows the ENTRY class diagram. The distinction between distinguished names and common names allows the grouping of different ENTRY objects by introducing additional hierarchy levels. Such a group is called component according to the CC/PP recommendation [13] and the GUP specification [14]. The key difference between this data model and the previously mentioned approaches lies in the fact that AMAYA provides contextual profiles (i.e. here: more than one profile) whereas the other approaches provide only one profile per user (or per device). A set of ENTRY objects will be provided by a single DATA ADAPTER and consists of multiple generic entity preferences as well as a set of service-specific entity preferences.

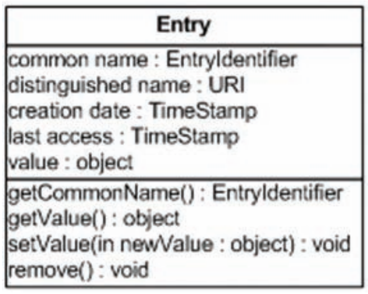

Entry
common name : EntryIdentifier
distinguished name : URI
creation date : TimeStamp
last access : TimeStamp
value : object
getCommonName() : EntryIdentifier
getValue() : object
setValue(in newValue : object) : void
remove() : void

Fig. 3. 'Entry' class diagram

4.3 Class: Profile

A PROFILE groups multiple units of personalization data and links them to a specific situation that describes a user or a group in a **precise interaction context**. Several PROFILE objects can be attached to an ENTITY. Usually, a PROFILE describes just one subset of all possible situations in which a user or a group can be encountered. There are situations like '*Bob being at home*', '*Bob using certain terminals*', '*Bob accessing information at a certain time*'. A PROFILE may also contain any reasonable combination of the situations described above. Each PROFILE is uniquely identified by a distinguished name. Figure 4 shows the PROFILE class diagram. Mapping of ENTRY objects to specific situations is done by a set of qualifiers, which specify all conditions that have to be met to consider a PROFILE valid for a specific situation. Besides these qualifiers, all profiles have to be ordered so that the PROFILE BROKER is able to identify a single PROFILE (or none at all), which is returned as the result of a query in a specific interaction context. A PROFILE may be enhanced with additional functionality, which can be plugged-in separately, e.g. Middleton [15] considered simple time-decay functions as an easy to use profiling technique that can be applied to content and rating-based profile representations. A PROFILE object could take care of the automatic aging of ENTRY objects in this way.

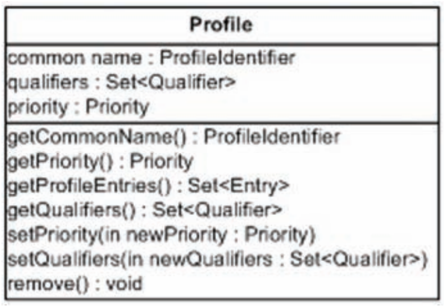

Profile
common name : ProfileIdentifier
qualifiers : Set<Qualifier>
priority : Priority
getCommonName() : ProfileIdentifier
getPriority() : Priority
getProfileEntries() : Set<Entry>
getQualifiers() : Set<Qualifier>
setPriority(in newPriority : Priority)
setQualifiers(in newQualifiers : Set<Qualifier>)
remove() : void

Fig. 4. 'Profile' class diagram

5 System Design: Interfaces

AMAYA consists of four functional components with distinct interfaces: the DATA ADAPTER subsystem, the PROFILE MANAGER subsystem, the PROFILE BROKER subsystem, and the actual RECOMMENDER subsystem.

5.1 Interface: DATA ADAPTER

A DATA ADAPTER wraps several personalization data units by providing a uniform data access interface for retrieval and storage of this information. Each DATA ADAPTER registers at the PROFILE BROKER once its data sources are ready to provide the data. These data sources can be accessed, e.g. via SQL / JDBC / ODBC in case of a full-fledged database or via REST[1] [16] in case of an embedded database running on a device with limited resources such as a mobile phone or PDA. This means that, if the personal preferences are stored on a mobile device, they are also available when the device is not connected to a network.

5.2 Interface: PROFILE MANAGER

The PROFILE MANAGER governs all entities that require contextual personalization data. All internal data regarding users and groups including the mapping of their preferences to specific situations is stored here. For each user or group there is a record called ENTITY and a set of PROFILE objects. The PROFILE MANAGER provides an interface that can be used by a service or by the PROFILE BROKER in order to process an ENTITY and all PROFILE objects that belong to the record. The personalization data itself is accessed via a DATA ADAPTER, which is returned by the PROFILE BROKER along with a list of ENTRY ids. It is also possible for an entity to decide to bypass the PROFILE BROKER's decision making process by manually specifying a PROFILE, which should be used for all queries that may follow. In addition to that, each PROFILE can also be excluded from future queries by manually disabling it.

5.3 Interface: PROFILE BROKER

The PROFILE BROKER retrieves and processes all queries for contextual personalization data. In order to process these queries the PROFILE BROKER has to check whether there is a proper match (i.e. PROFILE) to this query or not (i.e. that the resulting PROFILE has to match the situation description that has been specified in the query). If a PROFILE is found, the PROFILE BROKER returns a set of ENTRY objects together with the required DATA ADAPTER objects to retrieve the data. The next sections describe the matchmaking algorithm that has been implemented. A service that wants to personalize itself specifies the present situation of a user and requests from the PROFILE BROKER all preferences, which are relevant. The PROFILE BROKER uses the PROFILE MANAGER to get a list of

[1] Representational State Transfer

PROFILE objects, which belong to that entity, and tries to find a matching PROFILE given the description of the entity's situation. If a PROFILE is found, the broker returns a set of ENTRY objects, which are identified by the PROFILE, together with their DATA ADAPTER objects. The service has to use the provided DATA ADAPTER objects to retrieve the preferences values directly. It has to use the DATA ADAPTER objects to update the previously received preferences values, too. A set of privacy enforcement rules can be applied by a PEP[2] so that a service gets only preferences that it is allowed to receive (as stated by the user and his / her definition of appropriate policies). Figure 5 provides an overview of the preferences retrieval process. Please note that all DATA ADAPTER objects have been removed from the sequence diagram for a better readability.

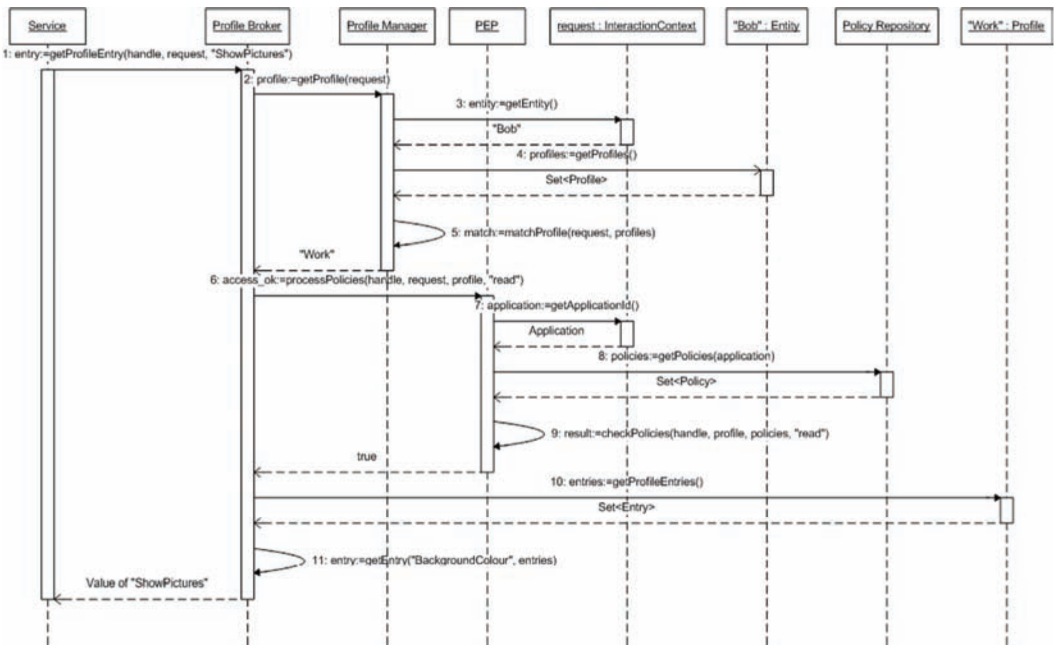

Service
Profile Broker
Profile Manager
PEP
request : InteractionContext
"Bob" : Entity
Policy Repository
"Work" : Profile
1: entry:=getProfileEntry(handle, request, "ShowPictures")
2: profile:=getProfile(request)
3: entity:=getEntity()
"Bob"
4: profiles:=getProfiles()
Set<Profile>
5: match:=matchProfile(request, profiles)
"Work"
6: access_ok:=processPolicies(handle, request, profile, "read")
7: application:=getApplicationId()
Application
8: policies:=getPolicies(application)
Set<Policy>
9: result:=checkPolicies(handle, profile, policies, "read")
true
10: entries:=getProfileEntries()
Set<Entry>
11: entry:=getEntry("BackgroundColour", entries)
Value of "ShowPictures"

Fig. 5. 'Retrieval of user preferences' sequence diagram

The following matchmaking algorithm is used by the PROFILE BROKER to process all queries: Each PROFILE is given a priority by the PROFILE MANAGER. This priority is used to sort all profiles from the highest priority to the lowest priority. This search order is used to determine the most appropriate PROFILE for a given interaction context. This means that the first PROFILE that matches the provided description is returned. The PROFILE qualifiers are used to determine whether a specific PROFILE matches the description or not. The sort sequence guarantees that the most relevant PROFILE is found, if any. This search algorithm also ensures that all conflicts (i.e. several profiles match the provided interaction context) will be

[2] Policy Enforcement Point

solved. There are two ways of providing the required description of an entity's situation. On the one hand the service can provide a simple description of the situation itself. On the other hand the description can be provided by an external component that manages all context data [17], which characterizes the present situation. This frees the service from the need to gather and to infer all context data itself. In the context of a seamless and context-aware computing environment, the second possibility has to be preferred. It enables a more complex characterization of the situation.

The RECOMMENDER employs the PROFILE BROKER to request a set of service-specific data that is updated or processed by learning algorithms and predictions methods, too.

5.4 Interface: RECOMMENDER

The RECOMMENDER enables the learning of user and group interests (e.g. 'Bob likes sports news.') and the prediction of their behavior (e.g. 'Since Alice likes Chinese food and since her interests are very similar to Bob's interests, it's very likely that Bob likes Chinese food, too.').

All recommendations are provided on the basis of a **two-step process**: At first a user model has to be learnt. On this basis a prediction / recommendation can be made. Figure 6 shows the learning of user preferences.

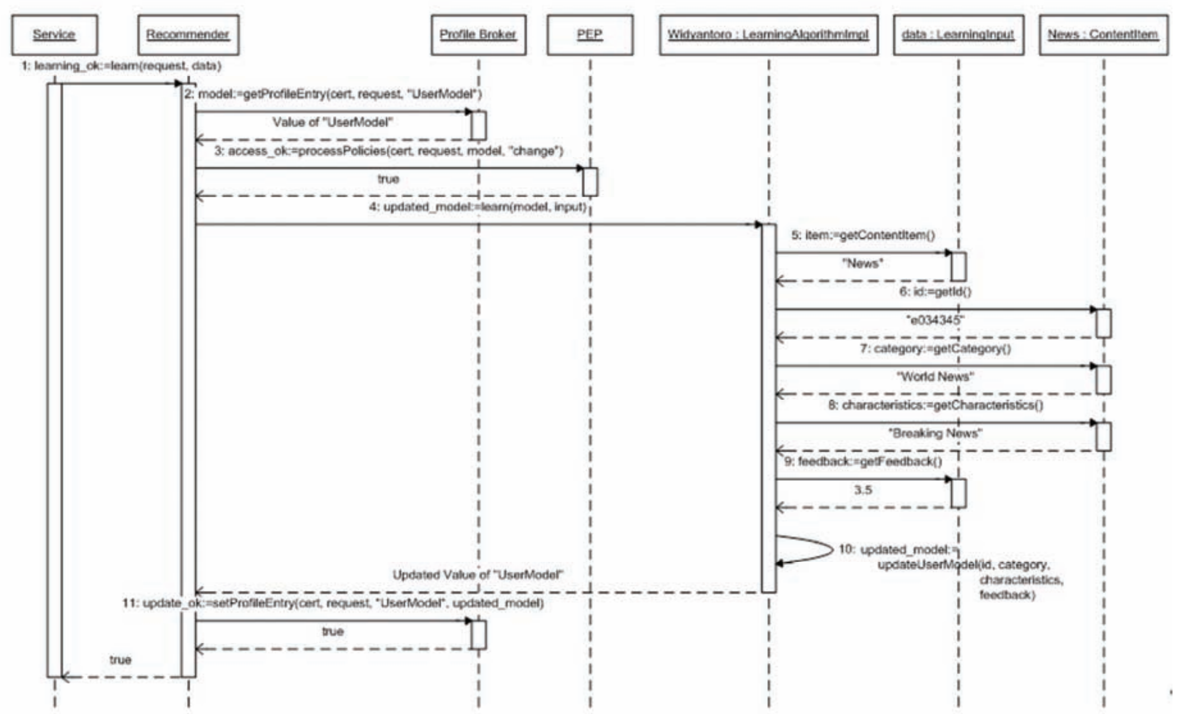

Service
Recommender
Profile Broker
PEP
Widyantoro : LearningAlgorithmImpl
data : LearningInput
News : ContentItem
1: learning_ok:=learn(request, data)
2: model:=getProfileEntry(cert, request, "UserModel")
Value of "UserModel"
3: access_ok:=processPolicies(cert, request, model, "change")
true
4: updated_model:=learn(model, input)
5: item:=getContentItem()
"News"
6: id:=getId()
"e034345"
7: category:=getCategory()
"World News"
8: characteristics:=getCharacteristics()
"Breaking News"
9: feedback:=getFeedback()
3.5
10: updated_model:= updateUserModel(id, category, characteristics, feedback)
Updated Value of "UserModel"
11: update_ok:=setProfileEntry(cert, request, "UserModel", updated_model)
true
true

Fig. 6. 'Learning of user preferences' sequence diagram

Content-based filtering algorithms and collaborative filtering algorithms allow the learning of service-specific interests based on either implicitly or explicitly gathered feedback. Explicitly gathered feedback is provided by means of positive or negative ratings, which is directly received from an entity (i.e. every time Bob rates a recipe received by a cookbook service, this rating is sent to the RECOMMENDER by the service in behalf of Bob). The service appends a description of the entity's present situation to the received feedback (e.g. 'Bob is driving in his car.'), so that the RECOMMENDER can request the appropriate entity preferences before updating them. The request is carried out with the help of the PROFILE BROKER. AMAYA requires implicitly gathered feedback (e.g. clickstreams) to be expressed in terms of a positive or negative rating by the service (instead of the user). In order to provide a service-independent interface to a number of learning algorithms and prediction methods the RECOMMENDER introduces the concept of a CONTENT ITEM, which describes an atomic unit of information (i.e. content) that belongs to a service.

A CONTENT ITEM abstracts from the actual information (e.g. '*Google, NASA Partner on Research*' or *Alicia Keys' CD 'Songs in A minor'*) and provides a description that can be used for learning purposes. This description consists of a uniform resource identifier and a set of distinct features such as service-specific categories and other content-related characteristics (e.g. further key words). The service-specific categories are based on an ontology-based content categorization scheme. This scheme allows expressing different content categories (e.g. '*sport news*' and '*business news*') as well as all relations between different content categories (e.g. '*soccer news is a sub-category of sport news*').

Basically, the interface of the RECOMMENDER provides two methods that can be used by services: The first method is used to provide input on a per service basis. The input is used during the learning process. It is received as either positive or negative feedback on specific CONTENT ITEM objects. The second method is used to answer the question whether a specific CONTENT ITEM is relevant to an entity in a specific situation or not. All learning algorithms and prediction methods have to implement the same abstract interface. With the help of this interface all learning algorithms and prediction methods provide their functionality transparently to a service. A service developer doesn't have to care about any algorithm-specific and implementation-specific details. All what is required is to choose the algorithm that suits a service best.

6 Discussion

The AMAYA RECOMMENDER subsystem does not care about the mapping of preferences to specific situations The approach described in this paper is what we call **context-agnostic learning for contextual recommendations**. Advantages as well as disadvantages have to be carefully considered when examining the first results of the proposed recommender system.

An advantage of this approach is that well-known learning algorithms and prediction methods can be used without any modifications. Rocchio's relevance feedback algorithm [18] can now be used to infer the relevance of information according to an entity's situation.

A drawback of this approach is that all situations that are available for learning purposes have to explicitly be defined by PROFILE qualifiers. Complex coherences between preferences and situations, which are normally not directly perceived by an entity, are learnt with a great effort only or not at all (i.e. as long as Bob is not aware[3] of the fact that he acts differently when '*being at home*' or '*being at his office*', the RECOMMENDER cannot learn the differences between both situations).

7 Evaluation and Conclusion

A first trial of AMAYA and a web-based news service showed good results so far. The implemented learning algorithm utilized a three-descriptor vector to learn changes in user interests over time [19]. Different user interests in predefined situations such as '*being at work*' or '*being at home*' had been learnt successfully. Based on the recommendations provided by AMAYA our news service delivered convenient information for each user.

The context-agnostic learning for contextual recommendations approach provides an easy to implement and easy to use way of how user models can be learnt and then applied to different situations by using well-known learning algorithms and without the need of "reinventing the wheel". Nevertheless, it remains to be seen how the initial results that have gathered so far can be improved.

8 Future Work

The next steps for this work are to run more trials based on several popular service scenarios and to implement different learning algorithms and prediction methods. The future evolution path of our contextual recommender system approach leads to the question on how the quality of the provided

[3] Bob can express his awareness by specifying different PROFILE objects for both situations.

personalization data can be improved in order to allow better personalization results. To answer this question it has to be evaluated how additional information / more detailed user models can be inferred from the readily available data. The idea of using the readily available entity preferences to infer additional personalization data could improve service scenarios where new or unused services suffer from the cold-start problem [20]. This could allow delivering of a better overall quality of the personalized service to the user. The usage of further ontological models as a basis for this process has to be analyzed.

References

1. Hobbes' Internet Timeline [Online]. Available: http://www.zakon.org/robert/internet/timeline/
2. D. Shenk, "Data Smog: Surviving the Information Glut", Technology Review, May 1997
3. Kobsa, "Generic User Modeling System", User Modeling and User-Adapted Interaction, issue 11, pp. 49 – 63, Kluwer Academic Publishers, 2001
4. Stewart, C. Niederée and B. Mehta, "State of the Art in User Modelling for Personalization in Content, Service and Interaction", NSF/DELOS Report on Personalization, 2004
5. D. De Roure, W. Hall, S. Reich, G. Hill, A. Pikrakis, M. Stairmand, "MEMOIR – an open framework for enhanced navigation of distributed information", Information Processing and Management, issue 37, pp. 53 – 74, 2001
6. M. Balabanovi, Y. Shoham, "Fab: Content-based, collaborative recommendation", Communications of the ACM, volume 40, no. 3, 1997
7. Schwab, W. Pohl, I. Koychev, "Learning to Recommend from Positive Evidence", Proceedings of Intelligent User Interfaces, pp. 241 – 247, ACM Press, 2000
8. L. Terveen, W. Hill, B. Amento, D. McDonald, J. Creter, "PHOAKS: a system for sharing recommendations", Communications of the ACM, volume 40, no. 3, 1997
9. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, J. Riedl, "GroupLens: applying collaborative filtering to Usenet news", Communications of the ACM, vol. 40, no. 3, 1997
10. T. Bauer, D. Leake, "Exploiting information access patterns for context-based retrieval", Proceedings of the 7th international conference on Intelligent user interfaces, San Francisco, California, USA, pages 176-177, ACM Press, 2002
11. Räck, S. Steglich, S. Arbanowski, "AMAYA: A Recommender System for Ambient-Aware Recommendations", Proceedings of the 2005 International Conference on Internet Computing (ICOMP), Las Vegas, Nevada, USA, June 27 – 30, 2005
12. ITU Telecommunication Standardization Sector (ITU-T) X.500, "The Directory: Overview of concepts, models and services", ISO/IEC 9594-1

13. World Wide Web Consortium (W3C), "Composite Capability / Preference Profiles (CC/PP): Structure and Vocabularies", W3C Recommendation, January 2004
14. 3rd Generation Partnership Project (3GPP) TS 23.240, "3GPP Generic User Profile (GUP) Architecture", version 6.6.0, December 2004
15. S. Middleton's Dissertation: [Online], chapter 3, available: http://www.ecs.soton.ac.uk/ ~sem99r/
16. R. Fielding's Dissertation: [Online], chapter 5, available: http://www.ics.uci.edu/~fielding/ pubs/dissertation/rest_arch_style.htm
17. S. Steglich, C. Räck, and S. Arbanowski: "Ambient-Aware Information Delivery for Beyond 3G Systems", 4th Workshop on Applications and Services in Wireless Networks (ASWN), Boston, Massachusetts, USA, August 9 - 11, 2004
18. Rocchio, "Relevance Feedback in Information Retrieval", The SMART Retrieval System: Experiments in Automatic Document Processing, pp. 313 - 323. Prentice-Hall, Englewood Cliffs, NJ, 1971
19. Widyantoro, T. Ioerger, J. Yen, "Learning User Interest Dynamics with a Three-Descriptor Representation", Journal of the American Society for Information Science (JASIS), 2000
20. J. Herlocker, J. Konstan, L. Terveen and J. Riedl, "Evaluating collaborative filtering recommender systems", ACM Transactions in Information Systems, volume 22, issue 1, pages 5-53, ACM Press, 2004