# Privacy-enhanced Intelligent Automatic Form Filling for Context-aware Services on Mobile Devices

Enrico Rukzio, Albrecht Schmidt, Heinrich Hußmann
Ludwig-Maximilians-University Munich
80333 Munich, Germany
{enrico.rukzio, albrecht.schmidt, heinrich.hussmann}@ifi.lmu.de

## Abstract

*In this paper we describe our approach to intelligent automatic form filling in the context of mobile services based on user data. In a small study we show that manual form filling is slow and perceived a stressful task. And hence it is not surprising that most current mobile services provide primary information, more complex interactions such as ordering a specific product are often very complicate and time consuming tasks or are even not provided. In our approach the individual forms elements are preset based on the content and context of the form, existing rules and the user's preferences.*
*Our work is grounded in an analysis of requirements. We investigated popular current WAP based mobile services and their characteristics. Through an analysis of conventional HTML based services we concluded which data a user has to type in when filling out a form. Furthermore we analyzed the behavior of people during filling out forms. Based on these observations and additional interviews, we present an overall architecture. A prototype, taking into account these findings is outlined at the end of the paper.*

## 1. Introduction

In the late nineties during the omnipresent internet hype also the phrase *Mobile Commerce* and the corresponding technologies for mobile services such as the Wireless Application Protocol (WAP), Wireless Markup Language (WML), compact HTML (cHTML) and i-mode appeared. Until now these services are not widely accepted in Europe.

Many people have devices and service contracts so that they could use mobile services but they still do not use these services. When talking to people several general reasons often are given why these mobile services are not used e.g.

- users see no need for such services
- it often does not work at all or it is not clear whether or not it will work in the given context
- it is too expensive for the added value provided
- it is too complicate to use and setup
- it is complicate to input text with T9 or a stylus

In our research we wanted to find out more details about the last mentioned problem and how to build a system to overcome this problem. In this paper we outline a system through which the user can be unburdened from the input of text through an intelligent automatic form filling process for mobile services. Our work is also concerned with a further requirement: ensuring privacy. Particularly for e-commerce applications based on transactions such as ordering a product or reserve a special service this is essential for many customers.

This work was performed in the context of the EU-project Simplicity [1] and the depicted architecture and the prototype are part of a current implementation process which leads to a bigger prototype that represents the whole project. The key concept of this EU-project is the Simplicity Device which might be an enhanced SIM card that is worn by the user which stores the user data and preferences. Based on this data, terminals, services and networks will be adapted through the Simplicity framework.

The paper is organized as follows. The next section relates our work to existing approaches. Section 3 analysis existing mobile services and conventional web based services. Furthermore we describe an initial user test which investigates the input of text into forms with a stylus based mobile device. In Section 4 we present a generic architecture for our concept. Afterwards we depict a prototype which is currently under development. The paper is completed by a discussion and outline of our further work.

## 2. Related Work

In this section we analyze in the context of mobile devices and services, the state of the art of text input, automatic form filling, adaptive web services and the usage of artificial intelligence in rule- and policy-based systems that support these adaptations.

There are big differences between the text input on a desktop PC and on a mobile device. Basically we can

distinguish between three different text entry techniques for mobile devices: key-based, stylus-based and predictive input techniques [2]. There are big differences regarding the text entry speed which is normally measured in words per minute (wpm). A skilled touch typist using a conventional keyboard can enter an average of 72 wpm [3]. The text entry on a mobile device is slower as you can see in the following Table 1:

| Name | WPM | User skills | Ref. |
|---|---|---|---|
| **PDA** | | | |
| Graffiti | 21,5 | average user | [3] |
| QWERTY keyboard | 20,2 | novice user | [3] |
| **Mobile Phone** | | | |
| T9 | 41-46 | expert user | [4] |
| Multi-press method | 25-27 | expert user | [4] |

*Table 1. Text entry speed on mobile devices*

Graffiti which is popular in PALM-OS - based systems is a stylus based input technology that is based on handwritten letters. In mobile phones every button also represents three letters. The T9 system uses a predictive algorithm which is based on a dictionary where words have a probability associated. Thus in many cases the user has only to select the buttons which represents among others also the intended letter to write a word. When using the traditional multi-press method the user has to select the intended letter through multiple pressing a key until reaching the desired one.

As one can see in Table 1 the T9 system is the fastest approach that is based on a predictive algorithm which takes the frequency or probability of specific words stored in a corresponding database into account. Unfortunately words that are used for format filling like name, address or e-mail often are not included in these databases. Because of this T9 is not very effective for form filling and often the traditional multi-press method on mobile phones is used that reaches 25-27 wpm when used by experts. Besides this it has to be taken into account that most users are not experts in T9 or Graffiti.

When using commercial websites the user very often has to input data in different kinds of forms. There are various proposals and concepts how this process can be automated. Chusho et al. [5] presented a system where an agent supports the automatic filling of forms in web applications. Therefore a corresponding architecture that is similar to modern AI architectures was developed that includes an inference engine, a learning facility and a knowledgebase. Furthermore there exists a W3C working draft Client Side Automated Form Entry [6] which includes among others an ontology for the description of identity, contact, postal, billing and organisational information. Barton et al. presented their XForms approach [7] that supports adaptive services through clients that fill forms with sensor data. Furthermore there are already some existing commercial applications like RoboForm [8] or iOpus Internet Macros [9] available that have functions for automatic form filling. In contrast to these approaches we concentrate on mobile devices and mobile services.

The exploration and development of context-aware services is currently a field that is considered by a lot of researchers and scientific projects. The basis for this is context information [10] like user data, device, location, surrounding devices, profiles, time, activity etc. This context information is used to adapt the services and contents. In the application area of this paper particularly personalized web applications for mobile devices have to be concerned that adapt web applications according the user and according the used device [11, 12, 13, 14].

The usage of rule- or policy-based systems that are based on concepts of the field of artificial intelligence are one standard approach when designing systems for context-aware services. Suryanarayana and Hjelm presented an architecture [15] that takes different profiles such as user profile, application profile and transport profile into account. Regarding the processing of this data they discuss possibilities that are based on rules languages such as RuleML and policies. They concern also the usage of XSL Transformations (XSLT) to adapt services according the context. A platform supporting coordinated adaptation in mobile systems that is based on policies is presented in [16] and [17]. They distinguish strictly between the monitored context information, the policies and the adaptation mechanisms. It is possible to use policies for different adaptations and the adaptation mechanisms are independent from the policies. Through this the mobile services can be adapted in a system-wide manner. *Rei*, a policy language for pervasive computing application was presented by Kagal et al. [18]. It is possible to express rules for rights, obligations, dispensations, and prohibitions. We restrict our approach to the domain of form filling for mobile services, which reduces the complexity to a great extend.

## 3. Analysis

After looking at related work this chapter analyses mobile services that are actually used and discussed actual problems regarding their usage. Afterwards we analyzed 20 different existing traditional web services and concentrated particularly on data users have to fill in for a transaction like e.g. order a book, rent a car or reserve a hotel room. We finish our analysis with an initial user test where we tested our concept through a mock-up.

### 3.1 Services

Currently the most services provided and used are based on WAP (Wireless Application Protocol) [19],

cHTML, HTML and XHTML. The following Table 2 shows the top 5 services of mobile services from three different German mobile network operators.

| | T-Mobile [20] | O2 Germany [21] | E-Plus Germany [22] |
|---|---|---|---|
| Top 5 Services | 1. Ring tones | 1. Live chat | 1. Ring tones |
| | 2. Download games | 2. eBay | 2. Playboy |
| | 3. Chat | 3. O2 E-Mail | 3. Sport news |
| | 4. Soccer | 4. O2 Ring tones | 4. Poptone |
| | 5. MMS-services | 5. O2 Games | 5. eBay |
| Format | WAP/WML | WAP/WML | cHTML, i-mode |
| Data from | 28/06/2004 | 28/06/2004 | 01/07/2004 |

*Table 2. Top 5 services of three German mobile operators*

As depicted in Table 2 mobile entertainment services like games, chat and ring tones are mostly used. Furthermore the mobile network operators provide often specific content that is accessed by a pre-configured mobile phone which has a corresponding soft-link to the corresponding portal of the mobile network operator.

We have recognized that a lot of commercial websites do not provide forms because it is so complicate to fill them with the limited text input capabilities of mobile devices. For instance the WAP version of a German online book shop Booxtra [23] has only these three possibilities to order a book after it was selected by the user: call Booxtra with the phone, sending of a corresponding email to my personal email account or using an existing Booxtra account. It's not possible to order a book online without having an existing Booxtra account. In the WAP version of a German mobile phone distributor [24] the user has to input his phone number after finding the desired mobile phone, afterwards the distributor calls the potential customer.

Furthermore our survey showed that most mobile services provide information only that is accessible through simply navigating hyperlinks. More complex interactions that require text input or form filling are seldom found.

### 3.2 Needed Personal Data

Due to this fact that there are very few mobile services that ask for user information we looked at web pages designed for desktop systems to identify the data that is required for potential services. We analyzed 20 different existing commercial HTML services on the internet and looked at the data that the user has to fill in when he/she wants to order or reserve something. Based on this analysis we concluded that it is possible to implement such a system because most of these forms where quite similar. We recognized that most of the services asked for a basic set of very similar data. Furthermore we found out that there are fixed groups of labels on the fields and of variable names for that field. From this we concluded that the usage of synonym lists for attribute names is essential for the development of our system. Through this input fields that are related to an element of our context model could be addressed in a uniform manner.

In Table 3 specific sets of variable names are shown. When building a system for form filling this has to be taken into account.

| | Amazon.com | Sixt.com | Hilton.com |
|---|---|---|---|
| **Address** | | | |
| First Name | name | | firstName |
| Last Name | name | nam1 | lastName |
| Adress1 | adress1 | | adress1 |
| Adress2 | adress2 | | adress2 |
| E-Mail-Address | email | emai | email |
| City | city | | city |
| State/Province/Region | state | | state \|\| otherstate |
| ZIP/Postal Code | zip | | postalCode |
| Country | country name | | country |
| Phone Number | voice | tel | phoneNumber |
| **Payment** | | | |
| Payment Method | paymentMethod | zah | |
| Credit Card | newCreditCardIssuer | | |
| Credit Card No. | newCreditCardNumber | ccnr | CCNumber |
| Expiration Date | newCreditCardMonth, newCreditCardYear | | CCExpMonth ,CCExpYear |
| Cardholder's name | newCreditCardName | | |

*Table 3. Variable names in three different forms*

As one can see at the variable names at Amazon.com that begin with *new* these names has also to be concerned. This indicates that a system will require sub-string analysis of variable names. At Sixt.com system specific attributes and uncommon abbreviation are used. At Hilton.com an example of predefined values and alternatives is given. The variable *state* is represented by

a selection list and alternatively it is also possible to use a text input field (*otherstate*).

### 3.3 Initial User Test

In an initial user test we studied the behaviour and experiences of the potential users using a web service on a PDA. We used a P800 smartphone from Sony Ericsson which accessed the internet via a Bluetooth connection. We have built a mock-up HTML-based hotel reservation service that was visualized by "Opera for Smartphone/PDA" on the P800. We provided two different versions of the reservation service. In the first version the user has to fill out every form field by herself whereby in the other version the fields where already filled out with user data. In the second version we integrated two errors (wrong street name and credit card) that the tester has to identify and to correct. This mock-up can be found under [25] and can be used with any web browser.


*Figure 1. Screenshot of our mock-up web application and activated virtual keyboard*

In booth versions the first name, last name, address, city, ZIP, phone number, e-mail address, method of payment, card number and expiration date have to be filled in, accepted or corrected.

We tested this with 8 users (colleagues from our department) whereby all where familiar with web forms and the concept of mobile services but used a P800 for the first time. As you can see in the Picture 1 the virtual keyboard was used to fill out the forms.

We took into consideration the usage of a mobile phone which supports the T9 system. We rejected this possibility because the potential testers were mostly familiar with keyboards but their experience with T9 differed greatly. It would be an interesting test to evaluate the speed of predictive systems like T9 when filling out forms because there are special requirements such as using special signs (e.g. +, @, -,.), input of numbers for ZIPs or credit card numbers and mixed input of text, special signs and numbers such as email addresses. But this was not the main focus of our investigation.

We show in the following Table 4 the durations for the filling of the empty forms and the completion time for pre-filled forms. Furthermore we also show the average times the testers needed in the first, the second and the third run.

| | Empty forms | Pre-filled forms |
|---|---|---|
| 1. run | 240 seconds | 60 seconds |
| 2. run | 170 seconds | 37 seconds |
| 3. run | 115 seconds | 33 seconds |

*Table 4. Average input times over all users, user were ask to perform several runs*

The most important result was that the testers need about four times longer to fill the empty form compared to the pre-filled form which needed corrections. Furthermore we recognized that the testers learned quit fast to use the virtual keyboard and the styles. But anyway they factor four exists also after three runs. From this we conclude that a form filling application would be extremely helpful and would if intensively used, support the further development of mobile services.

Beside this numeric results we recognized that most users where really frustrated when they used the stylus of the smartphone when they inserted text. Therefore we concluded it is very important that the user has to type in as few as possible.

At the beginning of every test we explained the tester the intension of the different forms. We explained them that in the second version there is an intelligent assistant which tries to fill out all fields based on the available user data. After this explanation many testers said they do not want to give their personal data away, e.g. as in the Microsoft .NET Passport [26]. From this we concluded the requirement that all data has to be stored on a physical device that is owned by the user himself. This supports the concept of the Simplicity Device as the key concept of

the Simplicity project where the user carries personal data stored on a private physical device. Furthermore the users liked being in control and want to see what data is filled in the different fields and so he/she has the possibility to delete or change the automatically inserted data. This approach provides the user an overview where and when data is transmitted and what data is given to which service.

To ease analysing the user tests we filmed the participants when using the mock-up web application with a video camera. Selected scenes can be found under [27].

## 4. The Overall Architecture

In this section we present an overall architecture that supports automatic form filling for mobile services that for instance are realized with WAP, X(HTML) or cHTML. This agent-based architecture is build on the evolving Simplicity framework, our analysis presented in Section 3 and on the work of Chusho et al. [5] that we adapted for mobile devices.

Figure 2 depicts the elementary parts of our overall architecture. The mobile device includes the four components proxy, web browser, personal assistant and user data. After the user requests a specific website which includes a form, it is transmitted from the server to the mobile device. To get access to the transmitted HTML-data a proxy is integrated in the mobile device. Before a website is transmitted to the web browser and before a website is transmitted to the server, the proxy has to be passed. The proxy provides an interface to access the actual web page. Such a proxy has the advantage that every existing web browser can be used by our system because we do not have to access the web browser through a special interface. The web browser interprets the different mobile services and shows them to the user.

The personal assistant uses the interface of the proxy to get the access to the current website. Based on this information and the user data the personal assistant performs the automatic form filling.

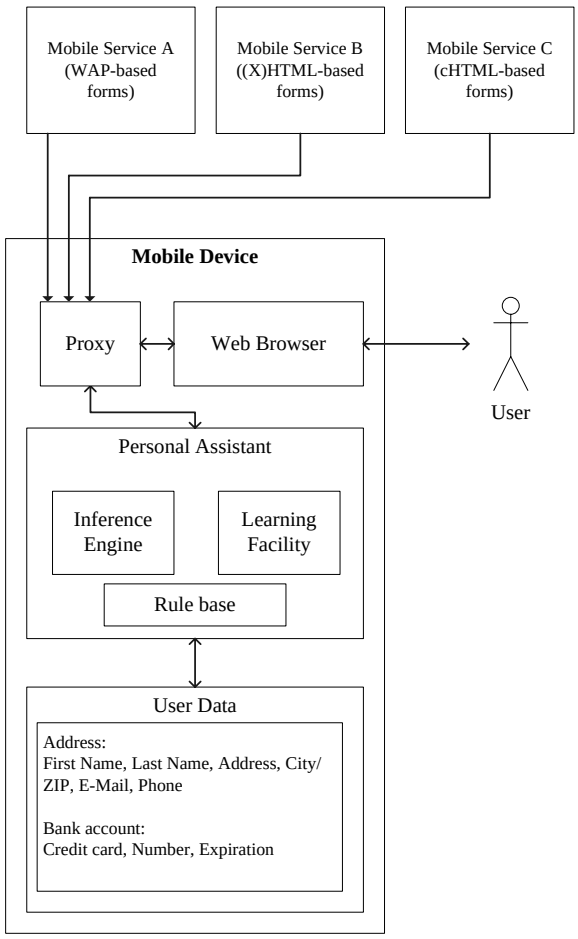


*Figure 2. Overall architecture*

### 4.1 Available Data

There are two different types of data which are the basis for the decision that has to be made by the personal assistant.

One type is data related to the user, such as address, bank account or preferences (e.g. non-smoker, “I like window seats”, etc.). This data can be initially defined by the user or could be learned through the observation of the user interaction. For the context model existing ontologies and infrastructures such as the 3GPP Generic User Profile [28] and the W3C working draft *Client Side Automated Form Entry* [6] are used.

The other type is data that is already existing in the web form. This includes a lot of information that is taken into account by the personal assistant. Examples are caption of the field (e.g. Name), definition of the field, the context of the field (e.g. fields before and afterwards, title of and heading in webpage), variable names, and context of the user (e.g. he/she is in an airport). By the field definition (e.g. *<input name="Name" type="text" size="20" maxlength="30">*) it is possible to get a lot of information about the desired input such as attribute name

(*Name*), data type (*text*) and length (between 20 and 30 characters). But there is of course ambiguity as the services developers choose the names of attributes. This depends on the methodology of the developer or the conventions of the used web authoring software. This problem might become less important through the further usage of XML or Resource Description Framework (RDF) in the development of mobile services.

## 4.2 Personal Assistant

The personal assistant is an active entity on the mobile device which works in the background and which is not directly visible to the user. After a new website of a mobile service (e.g. WAP-, (X)HTML- or cHTML-side) is requested the personal assistant analyses this and tries to fill out the fields of the included forms.

The personal assistant is a rule based agent whereby the inference engine that uses a rule base is responsible for the decisions which field is filled out with which content. Furthermore there is a learning facility included that enhances the rule base and the synonym lists.

Based on the approach from Chusho et al. [5] we will also use sets of synonyms that are related to our own context model and rules that concern the context of a field. Our ontology for user data includes for instance the variable *LastName*. It is possible to predefine a set of synonyms that can be used also as alternative variable names such as *FamilyName*, *surename* or *name2*. As rules we use *IF-THEN* statements which have the following structure: *IF* 'constraints' *THEN* 'insert value'. In the constraint statement we address the input field trough the synonym set rather then a concrete attribute name. The different attributes that can be considered in the constraint statement have been already described in subsection 4.1 *Available Data*.

The learning facility is able to extend the existing synonyms and to generate new rules. If there is e.g. a form including a field with variable name *name_2* this might be not filled out by the personal assistant because this one is not part of the existing synonyms. So the user has to fill out this field with his family name by himself. This filled field is transmitted through the proxy to server. The personal assistant recognizes that there is a field whose value corresponds to the family name of the person. So the existing synonym list (e.g. *FamilyName*, *surename* or *name2*) is extended by *name_2*.

Furthermore the personal assistant includes a function which receives and sends new rules and synonyms from and to a central server. If for instance the personal assistant of user A has identified a new synonym because he/she was one of the first users filling out the new form of a specific service, users B's form is already correctly filled because of the updated synonym list.

## 4.3 Prototype

Based on the overall architecture we are currently developing a corresponding prototype in the context of the EU-Project Simplicity [1]. As a first step there will be two different web based services available which can be used by a user through a mobile device. The personal assistant on the mobile device will try to fill out the transmitted forms based on the available information.

As you can see in Figure 3 our prototype will consist of a mobile device (Nokia 6600) and a server (Laptop Sony Vaio PCG-Z1XMP).

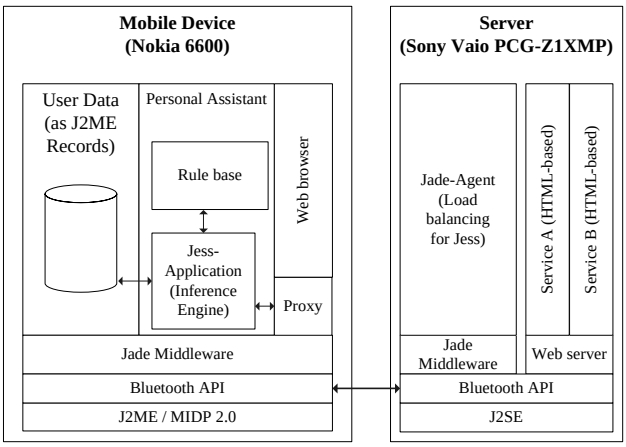


*Figure 3. Components of the prototype to evaluate the approach for automated form filling on mobile devices.*

We use Java as the implementation language. As our platforms we use on the mobile device J2ME (Java 2 Platform, Micro Edition) and on the server J2SE (Java 2 Platform, Standard Edition). The communication between mobile device and server is done via a Bluetooth connection. Based on the Java runtime environments we use the middleware Jade (Java Agent Development Framework) [29] which runs on J2SE as well as on J2ME. So we do not have to concern the special aspects of programming applications for mobile devices (e.g. asynchronous communication, unreliable connections, and limitations of mobile devices) because Jade provides an abstraction layer which hides these aspects.

Furthermore the server will run a web server and the two already mentioned web based services. We plan to provide as a first step a hotel reservation service and a car rental service.

The user data conforms to a special ontology that is currently developed in the Simplicity project and consists of different profiles such as user, device, services, etc. For the storage of these data we use J2ME records.

We will use the Jess (Java Expert System Shell) [30] as the basis for the implementation of the personal assistant. The personal assistant observes all received web pages which it gets form the proxy. Based on this, the

user data and the rule base of the personal assistant try to fill in the correct data. Afterwards the personal assistant sends the changed web pages and transfers it to the web browser. The user can now control the filled data and maybe he/she has to correct some of them or has to fill in leftover empty fields.

To reduce the processing power required on the mobile device we do not run the interference engine locally. We used a function of the Jade middleware which can split an agent in a front end and a backend where only the front end is running on the mobile device and the backend is running on the server. Therefore it is possible to migrate the inference engine to the server. The corresponding Jade agent is depicted in the server of Figure 3.

In the next step we will use existing mobile services to test our approach with this prototype.

## 5. Conclusion

In this paper we presented a conceptual system which supports the automatic filling of forms in web based mobile services. The basic idea is that there is a personal assistant running in the background of the mobile devices taking the user data and the content and context of the form into account. Based on this information and corresponding rules the forms are filled. The user has only to control, correct and confirm this pre filled forms.

## 6. Acknowledgements

This work was performed in the context of the framework of IST Project Simplicity (Secure, Internet-able, Mobile Platforms LeadIng CItizens Towards simplicitY) funded by the EU. The authors wish to express their gratitude to the other members of the Simplicity Consortium [1] for valuable discussions.

We thank our colleagues Paul Holleis and Matthias Kranz for contributing interesting ideas and comments to the project.

## 7. References

[1] Simplicity Project, http://www.ist-simplicity.org

[2] I. MacKenzie and R. Soukoreff, „Text entry for mobile computing: Models and methods, theory and practice", *Human-Computer Interaction*, 17, 147-198. 2002.

[3] J. Pierce and H. Mahaney, "Opportunistic Annexing for Handheld Devices: Opportunities and Challenges", *Human-Computer Interface Consortium*, 2004.

[4] M. Silfverberg, I. MacKenzie and P. Korhonen, „Predicting Text Entry Speed on Mobile Phones", *Proceedings of the SIGCHI conference on Human factors in computing systems*, The Hague, The Netherlands, ISBN 1-58113-216-6, pp. 9-16, 2000.

[5] T. Chusho, K. Fujiwara and K. Minamitani, "Automatic Filling in a Form by an Agent for Web Applications", *Asia-Pacific Software Engineering Conference 2002*, IEEE Computer Society, pp.239-247, 2002.

[6] P. Hallam-Baker, "Client Side Automated Form Entry", W3C Working Draft WD-form-filling-960416 http://www.w3.org/TR/WD-form-filling.html

[7] J. Barton, T. Kindberg, H. Dai, N. Priyantha and F. Al-bin-ali, „Sensor-enhanced Mobile Web Clients: an XForms Approach", *Proceedings of the twelfth international conference on World Wide Web*, ISBN 1-58113-680-3, Budapest, Hungary, pp. 80-89, 2003.

[8] RoboForm, http://www.roboform.com/

[9] iOpus Internet Macros, http://www.iopus.com

[10] G. Abowd and A. Dey, "Towards a Better Understanding of Context and Context-Awareness", in: *Technical Report GIT-GVU-99-22*, College of Computing, Georgia Institute of Technology, pp. 12, 1999.

[11] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Keränen and E-J Malm, "Managing context information in mobile devices". *IEEE Pervasive Computing* 2(3):42-51. 2003.

[12] G. Rossi, D. Schwabe and R. Guimar, "Designing Personalized Web Applications", *Proceedings of the tenth international conference on World Wide Web*, Hong Kong, ISBN 1-58113-348-0, pp. 275-284, 2001.

[13] D. Billsus, C. Brunk, C. Evans, B. Gladish and M. Pazzani, "Adaptive interfaces for ubiquitous web access", *Communications of the ACM* 45/5, pp. 34-38, 2002.

[14] Z. Fiala, M. Hinz, K. Meißner and F. Wehner, „A Component-based Approach for Adaptive, Dynamic Web Documents", *Journal of Web Engineering*, Vol.2 No.1&2, pp. 58-73, Rinton Press, September, 2003

[15] L. Suryanarayana and J. Hjelm, "Profiles for the situated web", *Proceedings of the eleventh international conference on World Wide Web*, Honolulu, Hawaii, USA ISBN 1-58113-449-5, pp. 200-209, 2002.

[16] C. Efstratiou, A. Friday, N. Davies and K. Cheverst, "A Platform Supporting Coordinated Adaptation in Mobile Systems", Proceedings of the 4th {IEEE} Workshop on Mobile Computing Systems and Applications (WMCSA) 2002, pp 128-137, 2002.

[17] C. Efstratiou, A. Friday, N. Davies and K. Cheverst, “Utilising the Event Calculus for Policy Driven Adaptation in Mobile Systems”, *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY 2002)*, 2002.

[18] L. Kagal, T. Finin and A. Joshi, “ A Policy Language for a Pervasive Computing Environment”, IEEE 4th International Workshop on Policies for Distributed Systems and Networks, Lake Como, Italy, pp. 63. 2003.

[19] WAP Forum, http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html

[20] T-Zones, T-Mobile Germany, http://www.t-zones.de/de/Getting_started/Wap/50simulator.html

[21] Wap Portal O2 online, wap.o2online.de

[22] E-Plus Germany, www.eplus-imode.de

[23] Booxtra, http://wap.booxtra.de

[24] Getmobile, http://wap.getmobile.de/

[25] Web service for initial user test http://www.rukzio.de/formfilling/

[26] Microsoft .NET Passport, http://www.passport.net

[27] Will be provided as a Real Video stream if the paper is accepted

[28] 3GPP Generic User Profile, Different documents under www.3gpp.org

[29] Java Agent Development Framework (JADE) http://jade.tilab.com/

[30] Java Expert System Shell (Jess) http://herzberg.ca.sandia.gov/jess/