Context-Aware Service Discovery using Case-Based Reasoning Methods

Markus Weber[1], Volker Hudlet[2], Thomas Roth-Berghofer[1,2], Heiko Maus[1], and Andreas Dengel[1,2]

[1] Knowledge Management Department,
German Research Center for Artificial Intelligence DFKI GmbH
Trippstadter Straße 122, 67663 Kaiserslautern, Germany

[2] Knowledge-Based Systems Group, Department of Computer Science,
University of Kaiserslautern, P.O. Box 3049, 67653 Kaiserslautern

{firstname.lastname}@dfki.de,v_hudlet@cs.uni-kl.de

Abstract. In this paper we introduce an architecture for accessing distributed services with embedded systems using message oriented middleware. For the service discovery a recommendation system based on case-based reasoning methods is utilized. The main idea is to take the context of each user into consideration in order to suggest appropriate services. We define our context and discuss how its attributes are compared.
The presented prototype was implemented for Ricoh & Sun Developer Challenge, thus the client software was implemented for the Ricoh's Multi Functional Product (MFP). The similarity functions were designed and tested using myCBR, and the service recommender application is based on the jCOLIBRI CBR framework.

Key words: Case-Based Reasoning, context, service discovery, myCBR, jCOLIBRI

1 Introduction

The Ricoh & Sun Developer Challenge[3] is a programming contest where students invent and implement innovative applications for Ricoh's Multifunctional Product (MFP)[4]. The MFP is an office machine that incorporates the functionality of several devices in one. It is a combination of printer, scanner, photocopier, fax and e-mail systems. Furthermore it offers developers Ricoh's Embedded Software Architecture SDK (SDK/J)[1] for implementing and delivering customized Java-based solutions hosted on Ricoh MFPs. A large touch screen is the main interaction device.

As we see the need to decouple business logic from the MFP, we decided to come up with a distributed architecture to access services. Obviously a lot of applications for MFP devices focus on document handling. Choosing a centralized

[3] http://edu.ricoh-developer.com/contest/open/index.jsp
[4] http://www.ricoh.de/products/multifunction/mediumworkgroup/mpc2550.xhtml

approach seems to be a reasonable way to implement services, because a service implemented and running on a server machine is not as limited as one on an embedded system. Furthermore the services are independent of the programming language offered on the embedded system, as the business logic is implemented on a server machine.

Such a service-oriented architecture will grow quite fast if a suitable software development kit is available for a community or other vendors, as can be seen by such current trends as the Apple Appstore[5] for the iPhone[6]. So service discoverability is issued in such a service oriented architecture, there is a need for an intelligent way to recommend services, especially considering the end user at an MFP is confronted with limited time and interaction convenience.

Let us assume the following scenario: A business man is traveling on a business trip to a conference. At the airport he might be interested in information related to *his location, the airport*, such as offers of the duty free shop. Arriving at the hotel he, as *a hotel guest*, might be interested in information about the hotel services. *In the evening* he might be looking for a good restaurant and in the morning at the conference he may want to read news about the financial market.

As we can see in this scenario the context is a good indicator which service might be helpful to recommend services for a user. Hence service recommendation based on context seems to be a promising approach. For the Ricoh contest prototype we considered the context of the user comprised of *user role, location type, type of device and daytime*. Examples for user roles are tourist, business man, student or professor, and for location types a hotel lounge or a kiosk as a public place and an office environment as a private place.

Case-based reasoning methods are a promising approach to implement recommendation systems (see, for example, [2]). Thus, CBR methods are also adequate for recommending services. As context information is a good indicator of which service might be helpful, the context is stored as a case. One of the big advantages is that CBR systems are capable of learning from the users feedback, hence service recommendation is improved by taking user feedback into consideration. Besides the recommendation, the context could also be used to personalize services.

In this paper we discuss the design of our architecture and its implementation. Moreover we suggest a service recommender that takes the context of the user into consideration to suggest services using case based reasoning methods. Our prototype focuses on the MFP as a service consumer. In Section 2 we will discuss the context, and in Section 3 we present the suggested architecture. The service recommendation using a case-based reasoning approach will be shown in Section 4. The communication between the services and the device is the focus of Section 5 as well as the client software rendering the user interface. Finally, we introduce a service that has been implemented as a proof of concept together with an outlook on what still has to be done.

[5] http://www.apple.com/de/iphone/appstore/
[6] http://www.apple.com/de/iphone/

2 Context

In our work we decided to use the context as an indicator to suggest services that fit to current needs of the user. In mobile computing context-awareness is an important research topic, as modern human mobile computing becomes more important. Modern mobile devices are capable of accessing online service via GPRS, UMTS, WLAN, and so forth, hence the need for personalised and adaptive information services is rapidly increasing.

Schmidt et al. [3] investigate in their work how to specify context in mobile computing. They introduced a working model for context and discuss mechanisms to acquire context beyond location, and application of context-awareness in ultra-mobile computing. Therefore they investigated the utility of sensors for context-awareness and present two prototypical implementations. In [4], Kofod-Petersen described an approach to facilitate the use of contextual information in order to improve the quality of service in a mobile ambient environment. So context-awareness seems to be a promising approach for the MFP, as information about the MFP's environment is easy to measure. Hence we need to define the term context and how our context looks like.

A popular definition of context is given by Dey [5]: "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 applications themselves."

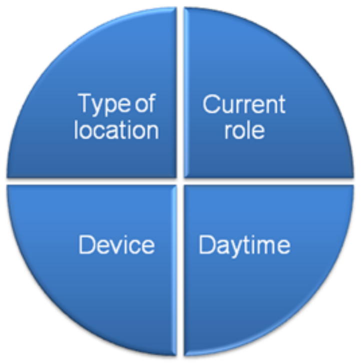


Fig. 1. Context of the MFP.

For our purposes we could take the definition quite literally. We looked at the context of an MFP and investigated which information could be automatically measured in order to characterize a situation and found that the attributes *user role, daytime, device,* and *type of location* were appropriate for defining our context (see Figure 1).

User Role Identification of the user role is not a trivial task. We need quite some knowledge about the user. For the prototype we assumed that the user is registered at a service provider and some initial knowledge about his role exists, such as his job or his hometown. By using simple assumptions like if someone uses an MFP in a city that is far away from his hometown, he might be there as a tourist, or on a business trip. To improve the result of this simple reasoning method, the registry can ask the user for feedback and propose possible roles to choose from.

Daytime Measuring daytime is a trivial task, as the devices have an internal clock. For our purpose? we divided a day into five intervals:

Morning	6:00am–11:00am
Noon	11:00am–2:00pm
Afternoon	2:00pm–6:00pm
Evening	6:00pm–11:00pm
Night	11:00pm–6:00am

During the day the needs of a user might change. For instance around noon or in the evening a restaurant guide might be more likely to be helpful as in the middle of the night.

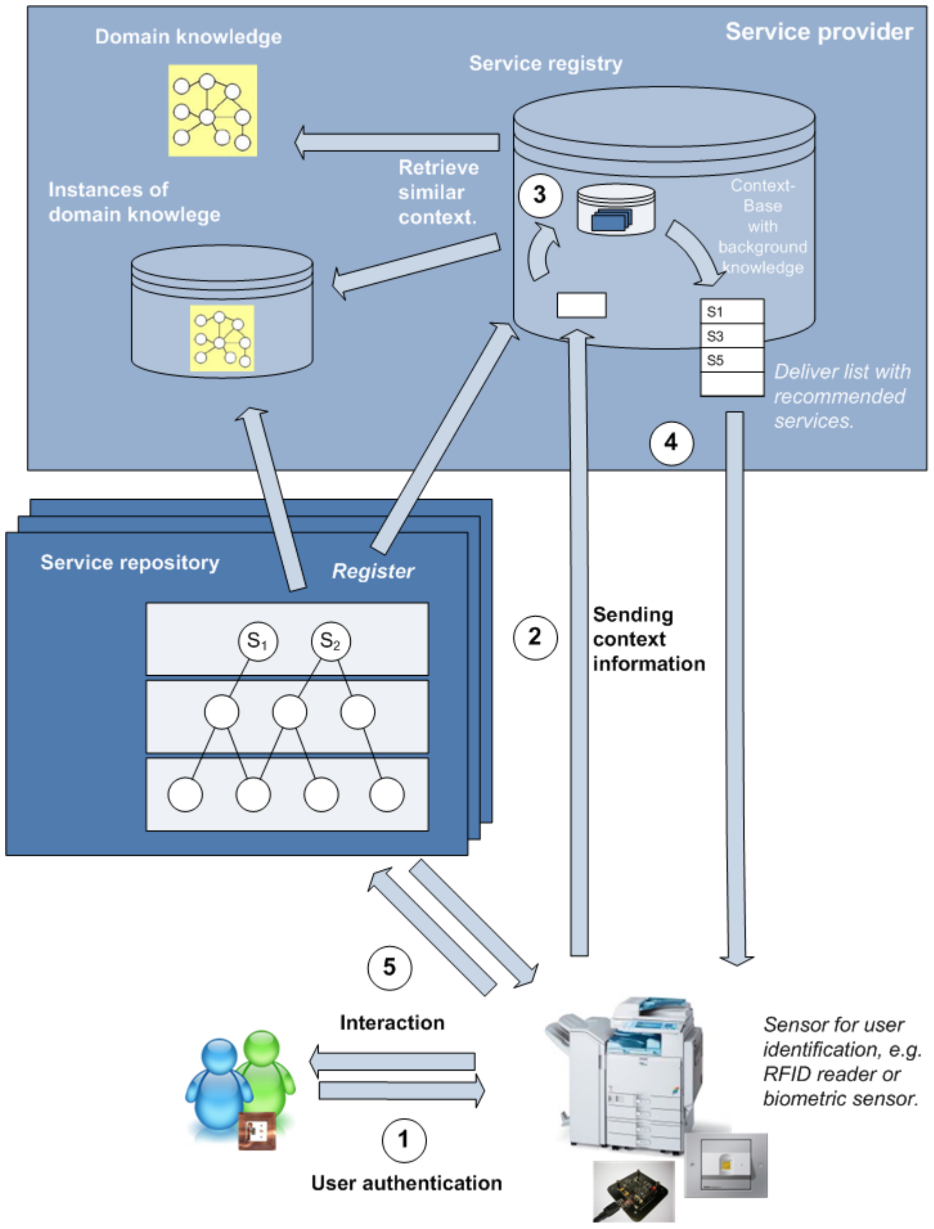
Device Even though our focus is on MFP devices, the services could also be used by mobile devices, as already addressed. Thus the type of the device is an important issue. Services producing larger documents are more interesting for e-readers or MFP devices as for smart phones, as reading larger texts on a smart phone can be unpleasant.

Type of location This attribute has to be set by the administrator of the device. Therefore a predefined set of possible types has to be available, such as public place, office or private household. According to the location several services could be excluded as the environment is not adequate, for instance a banking service on a public place. Another example could be a hotel, in which tourist services, restaurant and entertainment guides would be appropriate.

The following section will take a closer look at the architecture of system to give an overview where context and CBR methods can improve quality of the system. Further on the implementation of the CBR system and the similarity functions will be discussed.

3 Architecture overview

Our framework is a distributed architecture for accessing services with a mobile device or an MFP, respectively. The business logic is implemented in services running on server machines. The device just provides the user interface and communicates with the services. Figure 2 illustrates a schematic overview of the system and its components.



Service provider
Domain knowledge
Service registry
Retrieve similar context.
Instances of domain knowlege
3
Context-Base with background knowledge
S1
S3
S5
Deliver list with recommended services.
4
Service repository
Register
S1
S2
2
Sending context information
5
Interaction
1
User authentication
Sensor for user identification, e.g. RFID reader or biometric sensor.

Fig. 2. Technical overview of the whole system.

The *service provider* is the entity that offers services and the service registry. Users of the system have to be registered at the service provider as additional services can only be provided to registered users. Each user creates a profile with knowledge about his interests and some meta information about him, such as his job, his city and so on.

In order to authenticate users (1), we suggest, for example, to use RFID card systems (such solutions are widely available for MFPs) or biometric sensors on MFP devices, as no typing is needed for user password authentication. Context information, described in detail in Section 2, will be transmitted to the *service registry* (2).

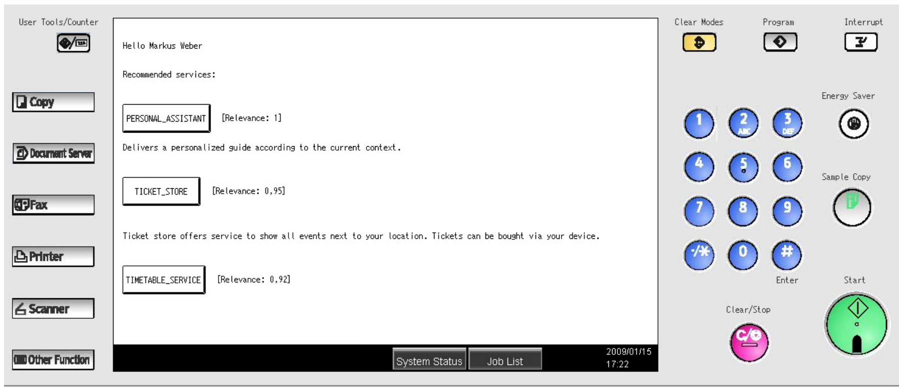
The *service registry* is the main component in the architecture as it manages the services and their recommendation. Each service, either of the service provider itself or other providers, has to register at the service registry. The recommendation is realized by using a case-based reasoning system (3). Services will be ranked according to their relevance, which, in turn, is calculated by comparing the context of the user with contexts in the case base. Each of the cases has an assigned service that suited the context best in a past situation. After calculating the similarity for all cases, a list of available services ordered according to their relevance is transmitted to the MFP (4).

The recommended services will be displayed on the MFP and the user is able to choose one of the suggested service (5). By selecting a service the communication process between service and user starts. The service sends a user interface form to the MFP which is rendered on the display. Each selection will be immediately transmitted to the service and triggers an action.

We distinguish between *push* and *pull services*. Push services use the capabilities of the device to scan documents or taking pictures. The information is "pushed" to the service and processed. Pull services retrieve documents from a service. In our prototype we just implemented a pull service, as a proof of concept (see Section 6). Further on several dummy services are implemented and added to the service repository and registered at the registry for demonstrating the recommendation of services. Nevertheless push services can be implemented as well as the MFP is capable of scanning documents or a modern mobile phone can take pictures.

In order to interact with a service a user interface must be provided (see Figure 3). The user interface should be generic enough to be feasible for all services, but must be flexible enough to be adaptable to each service. This dynamic requirement motivated us to build upon a user interface description language which is interpreted during runtime and rendered on the screen of the MFP. As an additional advantage of this approach other types of devices can also consume the services and adequately render their user interfaces.

For the user interface description language we use the open W3C standard XForms [6]. As the device does not support the complete bandwidth of user interface elements proposed by XForms, only a subset was implemented on the MFP. At the moment only text and buttons are supported. In later versions it is possible to increase the set of featured elements.


Hello Markus Weber

Recommended services:

PERSONAL_ASSISTANT [Relevance: 1]

Delivers a personalized guide according to the current context.

TICKET_STORE [Relevance: 0,95]

Ticket store offers service to show all events next to your location. Tickets can be bought via your device.

TIMETABLE_SERVICE [Relevance: 0,92]

Fig. 3. Screenshot of the recommended services running in the Ricoh SDK/J emulator environment. This screenshot is taken at an earlier development stage. Category browsing will be available in the next version.

After discussing the components from a top-level perspective, we will have a closer look at the system. As the service registry is the central component of the system it will be introduced in the following section.

4 Service Recommender

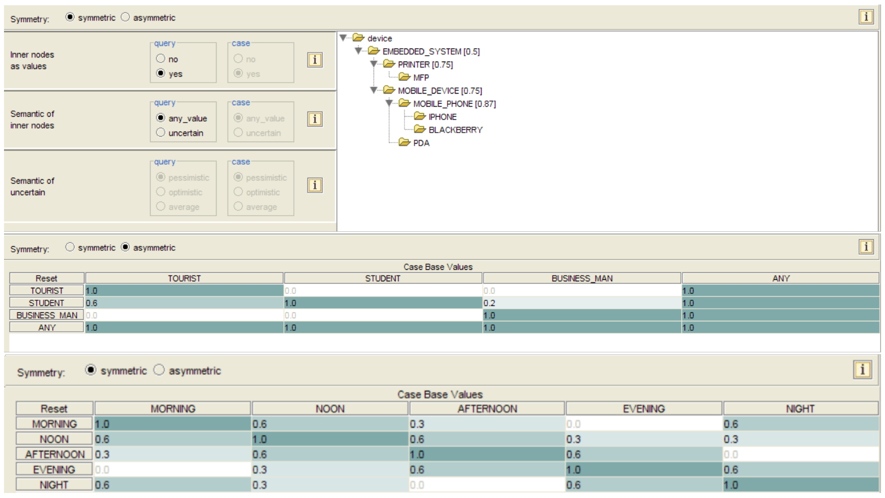
The main function of the service registry is the discovery of appropriate services. We interpret the context of the user as problem and the service as solution. By taking the current context into consideration the system calculates a relevance for the cases, stored in the case base. As the main idea of CBR is to learn from experience we see the service registry as an experienced advisor that knows about several contexts and can tell the user which services are available in order to help serving his needs.

Firstly each service has to register at the service registry and provide information, such as:

Service name Name of the respective service
Service identifier Service identifier used to address the service
Description Short text that describes the functionality of the service
Category Category of the respective service, e.g., tourist services.
Initial context Context that describes a situation in which the service is appropriate
Filter set [optional] Filters restricting the usage of the service in a certain context

The initial context is needed to solve the bootstrapping problem, as we need cases for our case base. So developers of the service have to think about a reasonable context, where the service is helpful. This context is a prototypical for this service and will be relevant to find the service in its initial state. For instance a restaurant guide could be helpful at a public place at noon or in the evening. This common case will be added to the case base.

During the retrieval process and the assembly of the service list, we also have the opportunity to apply filters. For instance there could be certain limitations for the services, e.g., a service might not be allowed to be accessed in a public place, because confidential data is accessed. Another example would be that only for certain cities local information is available. Thus services can be excluded by the application of a set of filters.


Symmetry: [✓ symmetric] asymmetric

Inner nodes as values — query: no [✓ yes]; case: no [✓ yes]

Semantic of inner nodes — query: [✓ any_value] uncertain; case: [✓ any_value] uncertain

Semantic of uncertain — query: [✓ pessimistic] optimistic average; case: [✓ pessimistic] optimistic average

```
device
  EMBEDDED_SYSTEM [0.5]
    PRINTER [0.75]
      MFP
    MOBILE_DEVICE [0.75]
      MOBILE_PHONE [0.87]
        IPHONE
        BLACKBERRY
      PDA
```

Symmetry: symmetric [✓ asymmetric]

Case Base Values

Reset	TOURIST	STUDENT	BUSINESS_MAN	ANY
TOURIST	1.0	0.0	0.0	1.0
STUDENT	0.6	1.0	0.2	1.0
BUSINESS_MAN	0.0	0.0	1.0	1.0
ANY	1.0	1.0	1.0	1.0

Symmetry: [✓ symmetric] asymmetric

Case Base Values

Reset	MORNING	NOON	AFTERNOON	EVENING	NIGHT
MORNING	1.0	0.6	0.3	0.0	0.6
NOON	0.6	1.0	0.6	0.3	0.3
AFTERNOON	0.3	0.6	1.0	0.6	0.0
EVENING	0.0	0.3	0.6	1.0	0.6
NIGHT	0.6	0.3	0.0	0.6	1.0

Fig. 4. Screenshots of the similarity functions designed with myCBR

As we have initial cases in our case base we needed to design our similarity functions that compare context information. In order to compare the user role we decided to use an asymmetric table similarity (cf. Figure 4, middle). For instance we need to find a measure how similar a student is to a professor or a business man to a tourist. The asymmetric character becomes obvious if we compare a student and a tourist. In our example, the student is more similar to a tourist, as they might have the same interests. If we compare a tourist to a student, the similarity might be even 0.0, as a tourist is not interested in services offered by a university.

For daytime we defined time intervals and gave them a symbolic value, such as MORNING, NOON and so on. Thus we are able to compare them with a symmetric table similarity where MORNING is more similar to NOON as to AFTERNOON, and vice versa (cf. Figure 4, top).

The location itself is just matched exactly. A more sophisticated solution would be to compare cities by using an ontology. Cities can be clustered according to special characteristics and arranged in a taxonomy. But as the location is just important for services, if they provide location based information, we just implemented the simple approach. The usage of location based information is shown in Section 6, where we introduce context-aware service. Currently the more important attribute is the type of the location as it describes the location in a more common sense. A hotel lounge in New York is quite similar to a hotel lounge in Rome. Thus this information is more important for service recommendation. We chose different symbols such as HOTEL-LOUNGE, KIOSK, OFFICE, and so forth and arranged them in a taxonomy using abstract terms to group them such as PUBLIC-PLACE.

In our prototype the device attribute is less important, as we only implemented a client for a specific MFP device. Nevertheless the basic idea of the architecture is to take different devices into consideration. A service that might be appropriate for MFP devices, can be inappropriate for smart phones, because the generated document is too large to read it on a small display. In order to group similar types of devices a taxonomy is defined, with abstract terms like MOBILE-DEVICE (cf. Figure 4, bottom).

The global similarity is a weighted average of local similarities. After a similarity value for the retrieved cases is calculated and ordered according their relevance, a set of services has to be transfered to the requesting device.

If none of the suggested service is appropriate for the user, he is able to browse a category based listing of the services. Even though the calculated relevance might be low the service might be appropriate in the users current context. Thus in the revise step of the CBR cycle [7] the user decides by selecting a service that it fits to his current context. As the selected service and the current context are transmitted to the service registry, the system can learn additional contexts where the service might be applicable in the retain step. Thus the relevance of the service will be higher if a similar context comes up.

For the implementation of the prototype we choose myCBR[7], as it supports rapid prototyping to design and test the similarity functions [8]. Furthermore these functions can be exported and used within the jCOLIBRI[8] CBR Framework [9] quite easily, which was used to implement the service recommender.

As we are using a distributed environment communication is quite important and is discussed in the subsequent section.

5 Communication

In our system we are faced with several issues. Services should be able to run on different machines, thus we need a mechanism to discover the recommended services. Furthermore an asynchronous communication is better suited as devices will not block while accessing service functionality.

All these issues can be handled by using a message oriented middleware such as the Extensible Messaging and Presence Protocol (XMPP) [10]. We assign a unique identifier (JID) to each device, service and the service registry to provide

[7] http://mycbr-project.net/

[8] http://gaia.fdi.ucm.es/projects/jcolibri/

a way to address each other. The communication between each component in the architecture illustrated in Figure 2 is realized by sending messages.

The device starts the communication process by sending the context to the service registry. This triggers the reasoning process and a list incorporating the recommended service and their respective JID is replied to the device. The user now selects one of the recommended services. By using the JID the device can address the service and directly start the communication process with the service, without knowing the actual host where it is running. As the routing is handled by the XMPP protocol even using a service running behind a firewall is possible. Encryption of the communication channels is easy to realize, as it is already offered by the protocol. Further on the protocol provides presence information. Using this information the service registry always knows which services are running.

For the implementation of the prototype we choose the Smack API[9] on the server side. On the MFP, we had to use the Beep API[10] as the Smack API is not supporting J2ME. As XMPP is an open protocol based on XML there are several libraries for various programming languages[11] available. Accordingly the implementation of the services is not limited to a specific programming language.

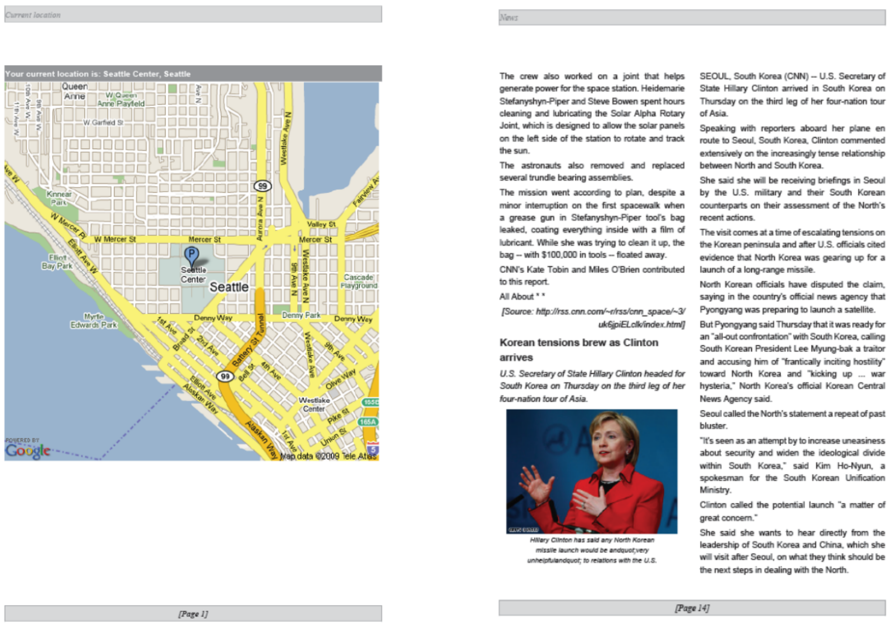
6 Context-Aware Service

After discussing the architecture and the service recommendation system, we will finally introduce a context-aware service. As a proof of concept, where context-awareness seems to be reasonable, the idea of a personal assistant came up. The concept envisions an assistant that knows where we are and which interests we have. According to this information the assistant collects information about the current location on a map and points of interests next to the current location. Furthermore it should compose interesting news articles that might be relevant according to the interest of the user.

Thus we need context information and some kind of user profile to achieve the goals of such a service. As already mentioned in Section 3 each user has to register at the service registry. With the registration the user creates a personal profile where he defines his personal interests. We decided to store this information using an ontology, the domain knowledge of the service provider. In our prototype the domain knowledge consists of simple profile information about the user, stored in the database with the instances of the domain knowledge (cf. Figure 2).

Information about the location of the device is transmitted by the device itself. An MFP will send the location that is configured by the administrator of the device. As not all mobile devices are equipped with GPS receivers, an automatic transmission can not be guaranteed or can be approximated, using Cell-ID of a base station in the GSM network.

[9] http://www.igniterealtime.org/projects/smack/index.jsp [Last access: 2009-02-09]
[10] http://sourceforge.net/projects/beep/ [Last access: 2009-02-09]
[11] http://en.wikipedia.org/wiki/List_of_XMPP_library_software [Last access: 2009-02-20]



Current location

Your current location is: Seattle Center, Seattle

[Page 1]

News

The crew also worked on a joint that helps generate power for the space station. Heidemarie Stefanyshyn-Piper and Steve Bowen spent hours cleaning and lubricating the Solar Alpha Rotary Joint, which is designed to allow the solar panels on the left side of the station to rotate and track the sun.

The astronauts also removed and replaced several trundle bearing assemblies.

The mission went according to plan, despite a minor interruption on the first spacewalk when a grease gun in Stefanyshyn-Piper tool's bag leaked, coating everything inside with a film of lubricant. While she was trying to clean it up, the bag -- with $100,000 in tools -- floated away.

CNN's Kate Tobin and Miles O'Brien contributed to this report.

All About * *

[Source: http://rss.cnn.com/~r/rss/cnn_space/~3/uk6jpiELclk/index.html]

Korean tensions brew as Clinton arrives

U.S. Secretary of State Hillary Clinton headed for South Korea on Thursday on the third leg of her four-nation tour of Asia.

Hillary Clinton has said any North Korean missile launch would be andquot;very unhelpfulandquot; to relations with the U.S.

SEOUL, South Korea (CNN) -- U.S. Secretary of State Hillary Clinton arrived in South Korea on Thursday on the third leg of her four-nation tour of Asia.

Speaking with reporters aboard her plane en route to Seoul, South Korea, Clinton commented extensively on the increasingly tense relationship between North and South Korea.

She said she will be receiving briefings in Seoul by the U.S. military and their South Korean counterparts on their assessment of the North's recent actions.

The visit comes at a time of escalating tensions on the Korean peninsula and after U.S. officials cited evidence that North Korea was gearing up for a launch of a long-range missile.

North Korean officials have disputed the claim, saying in the country's official news agency that Pyongyang was preparing to launch a satellite.

But Pyongyang said Thursday that it was ready for an "all-out confrontation" with South Korea, calling South Korean President Lee Myung-bak a traitor and accusing him of "frantically inciting hostility" toward North Korea and "kicking up ... war hysteria," North Korea's official Korean Central News Agency said.

Seoul called the North's statement a repeat of past bluster.

"It's seen as an attempt by to increase uneasiness about security and widen the ideological divide within South Korea," said Kim Ho-Nyun, a spokesman for the South Korean Unification Ministry.

Clinton called the potential launch "a matter of great concern."

She said she wants to hear directly from the leadership of South Korea and China, which she will visit after Seoul, on what they think should be the next steps in dealing with the North.

[Page 14]

Fig. 5. Personal document generated by the personal assistant service.

After transmission of the available context information the service asks the user some questions in a dialog-based manner to acquire which kind information and which amount of information the user would like to have. The selection of the news articles is realized by ranking of the news categories according to the interests of the user. Finally the result of the service, a personal document as illustrated in Figure 5, is sent to the MFP and can then be printed.

Besides the personal assistant service, services such as restaurant guides, timetable services, or event recommenders, which suggest and print events next to the users location and match his preferences, are examples for possible services. For the contest, we provided them as dummy services with appropriate initial context.

7 Conclusion and Outlook

In this paper we have shown an application framework which has been developed within the scope of the The Ricoh & Sun Developer Challenge programming contest. This framework enables the creation and deployment of context-aware services that can be consumed by devices, especially the MFP.

As the architecture and utilized technologies are not limited to the MFP, the architecture could be extended to access services with mobile devices. So instead of printing a document, it could also be shown on a mobile device, such as the e-reader iRex Iliad[12] or on a smart phone such as Apple's iPhone.

By offering a service development kit to other vendors or a community an application market platform can grow quite fast, ideally like the Apple App Store. Thus an intelligent way to recommend services could be helpful to find appropriate services. Crawling through categories is time-consuming and an appropriate service might even not be found. Therefore using CBR methods seems to be a promising approach in this scenario, as it is an intuitive way to use experience with usage of this service. Furthermore the system is able to learn from user feedback, as mentioned in Section 4.

A further development which supports other (mobile) devices is possible, as we build upon a programming language independent communication stack and user interface, but this remains a future task. As we have shown the core feature, the service recommender takes advantages of case based reasoning methods and frameworks. By the use of these and also by the use of the current surrounding context the recommender can propose suitable services. In future work the recommender will be refined to offer even better service matches.

References

1. Ricoh: White paper: Embedded software architecture sdk (2004)
2. Bridge, D.: Product recommendation systems: A new direction. In: Workshop on CBR in Electronic Commerce at The International Conference on Case-Based Reasoning (ICCBR-01. (2001) 79–86
3. Schmidt, A., Beigl, M., Gellersen, H.W.: There is more to Context than Location. Computers & Graphics Journal, Elsevier 23 **23** (1999) 893–901
4. Kofod-Petersen, A., Mikalsen, M.: An Architecture Supporting implementation of Context-Aware Services. In Floréen, P., Lindén, G., Niklander, T., Raatikainen, K., eds.: Workshop on Context Awareness for Proactive Systems (CAPS 2005), Helsinki, Finland, HIIT Publications (June 2005) 31–42
5. Dey, A.K.: Understanding and using context. Personal and Ubiquitous Computing **5** (2001) 4–7
6. W3C: Xforms 1.0 (third edition) (2007) http://www.w3.org/TR/xforms [Last accessed: 2009-02-09].
7. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AICom - Artificial Intelligence Communications, IOS Press **7**(1) (1994) 39–59
8. Stahl, A., Roth-Berghofer, T.: Rapid prototyping of cbr applications with the open source tool mycbr. In Althoff, K.D., Bergmann, R., Minor, M., Hanft, A., eds.: Advances in Case-Based Reasoning. Volume 5239 of Lecture Notes in Computer Science., Springer (2008) 615–629
9. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: Prototyping recommender systems in jcolibri. In: ACM Conference On Recommender Systems RecSys'08. (2008) 243–250
10. Saint-Andre, P.: Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920 (Proposed Standard) (October 2004)

[12] http://www.irextechnologies.com/products/iliad