Policy Based Adaptive Services for Mobile Commerce

Enrico Rukzio, Sven Siorpaes, Oliver Falke, Heinrich Hussmann
Media Informatics Group, Institute of Computer Science, University of Munich
{Enrico.Rukzio, Sven.Siorpaes, Oliver.Falke, Heinrich.Hussmann}@ifi.lmu.de

Abstract

In this paper we describe a novel adaptation architecture as well as a process for the development of policy based adaptive services for mobile commerce. Our architecture is based on three basic requirements and defines the four core elements context, policies, policy decision point and policy enforcement point. The proposed approach is based on the reuse and adaptation of existing and matured standards, APIs and middleware for representing context information, usage of policies for reasoning and for the communication between the involved parties. Our aim is to present a simple architecture taking as much as possible available work and software into account to support the rapid development of context aware mobile services. Furthermore we present a novel methodology for the definition of context information and policies that is sup-ported by a new UML based diagram and a module pipeline. We show the feasibility of the architecture as well as of the process based on a prototype which implements a typical scenario for an adaptive mobile service.

1. Introduction

It is commonly agreed that mobile devices like mobile phones, smart phones or PDAs are already now ubiquitously available and that their pervasiveness will rise in the future. In addition to that, the capabilities of these devices such as processing power, memory size, display quality and the number of supported networks are improving rapidly. Also the number of mobile services that can be used with these devices in-creases quickly.

But what about the user? A very important key component for the acceptance of new services for mobile commerce is their usability and complexity. Is the user willing to select a supported encoding and an appropriate resolution before watching a video on a mobile phone, to configure a huge set of parameters of his mobile device for accessing mobile services or to define which network operators he or she wants to use for which service? Beside these technical aspects different users have different needs. If these are not satisfied, the user might well stop using the service and refrain from using it again.

The field of context-aware mobile services addresses these issues whereby different types of context information are acquired and used to adapt to technical requirements and the user's needs. So far we have seen a lot of academic and industrial projects which focus on the acquisition of context information, the composition of context to higher level context information, different levels of context, representation, structuring and managing of context and reasoning based on context information for the development of adaptive and personalized mobile applications.

This paper proposes a simple architecture and a corresponding process for building and providing policy-based adaptive services for mobile commerce. In our opinion in the fields of artificial intelligence, expert systems, semantic web and agent based mo-bile middleware a lot of very good work has already been done. Why not just combine and adapt them?

This paper presents a generic architecture for the development of policy based context-aware services. We define three basic requirements for such systems and present the core elements of our adaptation architecture. We use the W3C recommendation Resource Description Framework (RDF) [17] without any extensions to get an interoperable representation of context information that could be used by different systems and which can be directly processed by most inference systems. We are using policies which can be seen as sophisticated conditional rules for the definition of the adaptive behavior of the system. In this field several projects have developed their own policy languages and inference engines. We think that a better strategy is to use corresponding software from the field of artificial intelligence. Therefore we use the fully developed Java Expert System Shell (Jess) [2] as our policy language as well as our inference engine. Such context-aware systems require also a corresponding middleware. We define the requirements for that and based on them we choose the Java Agent Development Framework (Jade) [8]. It is an agent based middleware that sup-ports mobile devices. We think this combination of RDF, Jess and Jade according to our architecture can be used for rapid prototyping and this or similar approaches are appropriate for powerful, scalable and maintainable mobile services for the consumer market.

A further problem when developing such systems is the lack of methodology for the definition of context information and policies. Therefore we define a corresponding process based on an extended UML diagram notation for the development and documentation of the adaptation process as well as our concept of a module pipeline for defining and structuring policies.

The paper is organized as follows. The next section relates our work to existing approaches. Section 3 describes a typical scenario for a mobile service that shows which aspects are important to fulfill the technical requirements as well as the user's needs. This is followed by the presentation of our general architecture for policy based adaptive services for mobile commerce. In Section 5 we describe a novel process for defining context information and policies. Afterwards we depict a prototype which shows the feasibility of our architecture as well as the profile and context definition process. The paper is completed by a discussion and outlines our future work.

2. Related Work

Context aware mobile services are currently one of the most interesting fields of re-search in the area of mobile computing. There is also a strong relationship to ubiquitous or pervasive computing and mobile communications. In this section we will give a compact overview of related work and state of the art in context modeling, context-aware mobile service platforms and policy-based systems.

There exists a huge set of approaches for gathering, describing and structuring con-text information, see [16], [3] and [6] for longer surveys. Not surprisingly, all these work deals with people, locations, devices, services, networks and their static and dynamic relationships. On the one hand, there are approaches that try to model the whole world which is in our opinion a wrong way because of the complexity of our world. On the other hand it is important to have standards that describe context information in an abstract manner like RDF or OWL and standards that define the structure of context information for a specific application area like the Core Information Model (CIM) [9] or UAProf [10]. A very important advantage of semantic knowledge representation is that existing ontologies can be reused by new applications.

Policies or rules that define the behavior of adaptive or context-aware mobile systems are used by different research projects because of the flexibility of this approach. The term *policy* is mostly used by people that have a background in policy-based networking [11] and the term *rules* by people that have a relationship to artificial intelligence or intelligent agents [12]. In both cases, ongoing work is mostly dealing with sophisticated conditional IF – THEN statements which allow the declarative description of systems behavior. The advantage of the usage of policies is that the policy decision point creates a custom program for each situation that arises and handles adaptations that the programmer might not have imagined. The policy decision point is the program which makes decisions based on context information and policies.

Schmidt et al present in [20] a WML-based application Context-Call which offers an interface that shows in which situation (e.g. meeting, working, busy) the receiver of the call currently is. Based on this information the caller can decide if he wants to leave a message or to continue or cancel the call.

Efstratiou et al [1] used the event calculus for policy driven adaptation on mobile systems. In their system they separated policies and adapted applications. By this, conflicts and suboptimal performance between multiple adapted applications could be prevented. Furthermore they developed a new policy language with explicit expressions of time dependencies. Chisel [13] is an open framework that supports unanticipated dynamic policy-driven adaptation based on contextual information. A policy-based approach was chosen to drive the adaptation mechanism by incorporating user and application specific semantic knowledge and intelligence, combined with low-level monitoring of the execution environment. A new human readable declarative policy language to adapt service objects based on a meta type based mechanism was developed. Another policy-based approach that was developed by Lago [14] supports users in defining their personalized perception of services on a high-level basis to define how, when and where to be contacted. A simple, flexible, context and application independent policy language was developed that is evaluated by a novel inference engine. Suryanarayana and Hjelm discuss in [15] a profiles view of the situated web architecture and the technologies useful in that respect. They describe four different profiles for describing the user, applications, devices, data transport aspects and how this information could be used by XSLT style sheets or rules.

3. Scenario

In this section we define a scenario for a context-aware mobile cinema information service which we used to evaluate our concept as well as to illustrate our architecture and methodology in this paper. The core concept of the scenario is a user who is standing in front of the cinema and he or she is not sure which movie is the most interesting one. In these situations, the cinema offers a mobile service where the user can get information about the current program with his or her own mobile phone, and as a special feature the user can download movie trailers.

Especially for the download or streaming of the videos a lot of context information should or must be taken into account. The most important parties involved in this process are the user, the device of the user, the offered videos and the available networks.

For the user we defined three different preferences which could be of interest and which could be changed by him or her. For every preference a weighting factor between 0 (“this is not important for me”) and 1 (“this is very important for me”) can be defined by the user. With the first preference *quality* the user can indirectly influence the visual quality of the video which is based on parameters like resolution or encoding. Via the parameter *speed* the duration of the transfer of the video from the server to the mobile phone can be defined, which is for instance based on the selected network type or the amount of data of the video. Adjusting the parameter *cost*, the user can define preferences regarding the costs for viewing the trailer. This aspect can for instance be influenced by the selection of the network provider.

The mobile device of the user has a specific screen resolution and we assume that it is possible to play videos with this resolution and also videos that have a smaller resolution. Furthermore the mobile device is characterized by a set of supported network types and a set of supported video encodings.

The different trailers for the movies that are currently showing at the cinema are available in different video encodings (e.g. MPEG-4, Real Media, H.263), different resolutions and amount of data per video. The user does not need to pay a fee to the cinema information service for downloading trailers. The user has only to pay the fees to the network provider for the transmission of data.

Furthermore the user in our scenario can easily switch between different network providers who have different pricing models and offer different network types. Every network type (e.g. GPRS, UMTS, WLAN) is characterized by its transmission speed. Regarding the price there is no difference for using e.g. GPRS or WLAN. The user pays only a fixed price per transmitted amount of data which is defined by the network provider.

In this scenario for the download of the video the policy-based decision process must lead to a result that defines which video (e.g. resolution, encoding, size), which network provider and network type (e.g. GPRS, UMTS, WLAN) should be selected based on the capabilities of the mobile device (e.g. supported network types, resolution, encoding) and the preferences (quality, speed, cost) of the user.

4. The Overall Adaptation Architecture

In this section we present an overall architecture for policy based adaptive mobile services. We specify three basic requirements for such systems and define the adaptation architecture which represents the core elements of our system in a compact way. This is followed by subsections that describe how context information is represented and which policy language and policy decision point we use. Afterwards we present the physical view of the architecture that shows which distributed elements have which functions. This section is completed by a discussion about the required communication middleware.

4.1. Requirements

For the development of our overall adaptation architecture we define the following three basic requirements:

- **Uniformity** in the different adaptation areas

The elements of this architecture will be distributed over different servers (e.g. for service provisioning, billing, network provider) and different mobile devices. For the flexibility, compatibility, extensibility and adaptability it is very important that the representation of context information, the definition of policies and the reasoning take place in a uniform way.

- **Separation** of context, policies, policy decision point and policy enforcement point

In some systems the mentioned elements are woven into a single adaptation application. This could lead to unintentional adaptations and hysteresis effects if two applications run in a single system and do not share common context information or make fully independent decisions. Through the separation of the different elements it is possible to build systems that act consistently in a global way. Furthermore it is easily possible to change the context information, to modify policies and to integrate new adaptations.

- Policy language should be **generic** regarding the range of adapted services

The policy language should not be specialized for a specific adaptation area. This allows the integration of arbitrary adaptations or decisions requested by different entities. Through this it is also possible to support a system wide adaptation process.

4.2. Core Elements

The core elements of our basic adaptation architecture are shown in Figure 1. The architecture is based on the principles of rule- or policy-driven systems which are used by a lot of adaptive systems [1, 2]. All of them take context information into account, have a policy language, a policy decision point and a policy enforcement point, even if they use another naming or some elements are combined or split. As already mentioned the representation of *context* information such as user preferences, device capabilities and available services is one key component for such systems.

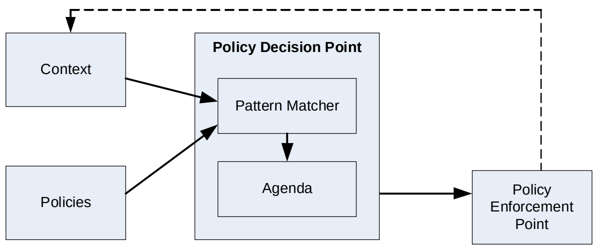


Figure 1. The core elements of the adaptation architecture

Policies are rules that can be seen as sophisticated conditional IF – THEN statements that define how the system reacts in a specific context. The *policy decision point* (PDP) which applies policies to context information consists of a *pattern matcher* and an *agenda*. The policy decision point works in cycles and in every cycle a policy can be fired. As the first step in a cycle the pattern matcher compares all policies with the context information and generates an unordered list of policies that should be activated in the current cycle. These and the policies which have been activated in a previous cycle form a conflict set. The agenda is an ordered list of activated policies which is generated by a conflict resolution algorithm from the conflict set. As the next step the first policy of the agenda will be fired and the action part (*Policy Enforcement Point*) will be executed which could lead to a change of the context information or to any other action that influences the surrounding system.

4.3. Context

As already mentioned in the section about related work, different approaches for acquiring, describing, representing, structuring and querying context are currently discussed in research. Some projects try to establish generic ontologies for the description of context information in a homogenous way. This approach is not feasible because it is not possible to describe all aspects of the whole world and often adaptations are interested in different aspects of an entity. Furthermore some people think that structuring context information, e.g., into high-, middle- and low-level is a suitable solution for managing complex context information.

We believe that a concentration on the basic results that the fields of artificial intelligence and semantic web have produced in the last decades is the most suitable solution for describing context information. Here context is always a triple consisting of a *resource*, a *property* and a *literal*. We use the terminology of the Resource Description Framework (RDF) because we use this standard to describe context information in a way that is understandable by policy decision points which again are often based on rule engines. We think that the functionalities that are given by RDF are sufficient for the development of policy-based adaptive mobile services and that high-level concepts might be useful but most of them are complicate, too specialized for a specific application area and often not compact and understandable and thus their acceptance gets questionable.

4.4. Policies and Policy Decision Point

The development of a new policy language or of a policy decision point which is practically useful needs several years of development effort because of the high complexity of such systems. Therefore we think that reuse and adaptation of an existing language and inference engine is the best solution for the development of prototypes as well as sophisticated products.

As already mentioned in related work different approaches exist which can be reused and adapted. There are some candidates in the field of semantic web research for a policy language and inference engine such as DAML Rules [4] or the generic rule reasoner of Jena [5]. But all of them are not yet in a mature state und therefore not yet useable for our purposes. In the field of artificial intelligence, however, much effort has been put into the development of fully-fledged languages. In our architecture we use a LISP-based rule language and the rule engine Java Expert System Shell (Jess) [2] because of their function range, development status, extensibility, availability and excellent documentation. Jess is also the basis for the Java Specification Request (JSR) 94, the Java Rule Engine API [18].

The Jess language consists of three basic elements: *templates* for the definition of the type of a fact, *fact* which is a piece of information and *rule* for the definition of IF – THEN statements, i.e., policies.

```
1  (deftemplate mobilePhone
2      (slot encoding))
3
4  (deffacts example
5      (mobilePhone
6              (encoding mpeg)))
7
8  (defrule supports
9      (mobilePhone
10             (encoding mpeg))
11     =>
12  (printout t "Phone supports mpeg
     encoding." crlf))
```

Figure 2. Example of the definition of templates, facts and rules

Figure 2 includes a small example of a Jess program. First a template describing the encoding is defined (lines 1-2), then a fact which represents a concrete mobile phone with a specific encoding is defined (lines 4-6) and then the rule *supports* (line 8-12) is defined that will be fired if there is a *mobilePhone* with the *encoding mpeg*. The => sign could be interpreted as a logical implication; operationally it is equivalent to a THEN statement.

In our architecture *facts* are represented by RDF, *templates* are represented by RDF Schema and Jess *rules* are called policies. Because of the similarity of *facts* and RDF as well as *templates* and RDF schema it is easily possible to transform them with XSL transformations (XSLT).

4.5. Physical View

The physical view addresses the distribution of the core elements of our basic adaptation architecture to different entities in the network such as sensors, services that are provided by servers, or mobile devices.

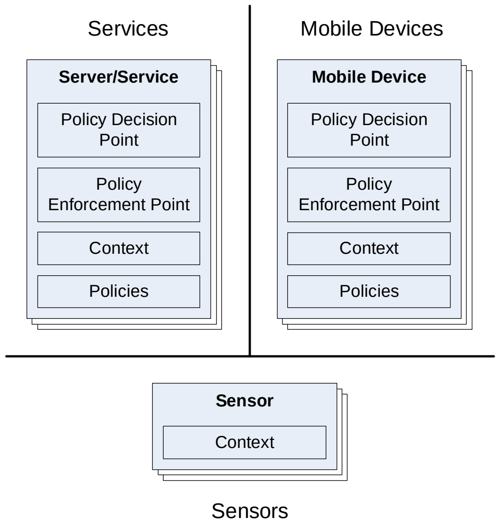


Figure 3. Entities in the physical view

Figure 3 shows the three different kinds of entities in our physical architecture: services, mobile devices and sensors whereby every entity could exists 0..n times. Every entity is connected with the network so that each entity could talk with every other entity. A *sensor* could be used to acquire context information such as weather, noise, social contexts or proximity of augmented objects or people. A server provides *services* that could be directly or indirectly used by the user. A mobile device is for instance a mobile phone, Smartphone or PDA. Services and mobile devices could include a policy decision point and a policy enforcement point. Furthermore they can provide and manage context information and policies. Every service or mobile device could integrate all of the mentioned core elements or just three, two or one of them.

4.6. Communication Middleware

The context information is based on sensors, databases or files which can be distributed over the whole system. It is not feasible to have one central physical database for storing, accessing and retrieving context information. The reason is that such an approach would lead to a huge communication effort which is particularly not feasible for addressing mobile devices because of the high transmission cost. Therefore a scalable decentral solution is needed which can address local context information and provide this to incoming requests.

This can be achieved through a distributed peer-to-peer based architecture that does not include central servers and clients that send requests to these servers. As already mentioned the entities service and mobile device could initiate communication as well as be subject of a request. A Policy Decision Point, e.g., can request context and policies that are locally as well as remotely (sensor, other mobile device or other service) available. Therefore all entities of our architecture can be seen as peers in a peer-to-peer – based middleware with a distributed system topology. Furthermore a powerful, distributed and scalable directory service must be available trough which every peer could address all other peers and their interfaces.

One next step when talking about peer-to-peer systems is the usage of the agent paradigm on the top of such a peer-to-peer architecture. An agent is characterized by three attributes *autonomous*, *proactive* and *social* [7]. Services and mobile devices that have a policy decision point are *autonomous* because they have control of their own actions. They make independent decisions based on context information and policies that are executed by policy decision points. Policy decision points are also *proactive* because they initiate the acquisition of context and policies as well as trigger actions in policy enforcement points. Furthermore the entities of our architecture must be *social* because they offer information to other entities and accept defined commands from other entities. Therefore the different entities service, mobile device and sensors are agents whereas sensors are limited agents.

The development of such a peer-to-peer middleware is a very time consuming part. Therefore we use in our architecture the Java Agent Development Framework (Jade) [8] which fulfills all the mentioned aspects. It also supports wireless and wired connections and it runs on mobile devices. The interoperability is assured through the usage of the FIPA (Foundation for Intelligent Physical Agents) – standard and the communication between the agents is defined by the FIPA Agent Communication Language.

5. Defining Context and Policies

After the design of the architecture one further important part is the definition of context information and the policies needed for the policy-based adaptation process. For this complicated and time-consuming part there is so far no easy and practical methodology or visualization available. In this section we describe a process which supports the developer during this process.

Like other software development processes, this methodology is iterative because not all requirements, the desired result and the intermediate steps can not be recognized at the beginning. We define five different steps which are needed for the definition of context and policies that are explained afterwards. Their processing sequence is visualized in Figure 4.

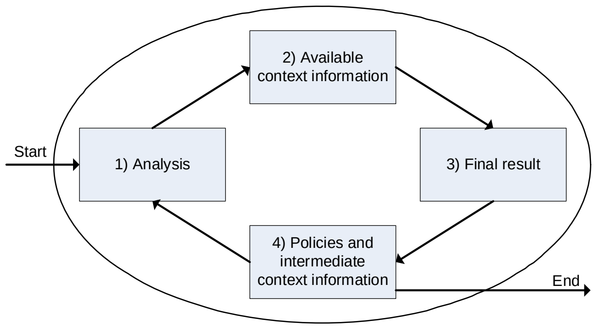

2) Available context information
Start
1) Analysis
3) Final result
4) Policies and intermediate context information
End

Figure 4. Different steps for the definition of context and polices

Different steps for the definition of context and policies:

1. Analysis of the requirements of the specific adaptation like in any other software development project.
2. Definition of the available context information (knowledge engineering)

The developer should collect all available context information that might be useful for the desired decision process und he or she should also define the corresponding data structure. At the end of this step it is already possible to define the data structure of the available context information and concrete example data as RDF Schema and RDF documents.

3. Definition of the desired final result

In this step the developer should specify the desired result which is also a piece of new context information generated by policies. As in step 2 it is possible to define the context information in RDF and RDF Schema.

4. Definition of the policies and intermediate context information

   a. Gradual development of policies based on existing context information until the definition of the desired final result has been reached.

   b. Iterative development of new preliminary context and rules on the basis of available context information and preliminary results

   c. Separating of context information and policies into modules which leads to a module pipeline

   In this step Jess policies as well as RDF and RDF Schema for the description of the intermediate context information can already be defined.

5. If all context information, the final result as well as the intermediate context information have been defined, this process is either finished or the next iteration is started.

5.1. A Diagram for the Visualization of Context and Policies

To support the proposed process of the definition of context information and policies we have developed a specialized diagram which helps the developer in the different steps and is also very suitable for documentation proposes. The diagram is based on the UML class diagram and integrates static (structure of the context information) as well as dynamic information (execution of policies). It shows the available and intermediate context information, the final result, the modules, the processing order of the modules (module pipeline) and indirectly the execution of policies.

Figure 5 shows the core elements of our diagram and the depicted example is afterwards used for the explanation of this visualization. A module is represented by the space between vertical dashed lines and its name is depicted like a state in an UML state diagram inside a rounded rectangle on the top. The context information is visualized like classes in a UML class diagram. This should look familiar for people who are already used to this type of diagram. Figure 5 shows the different resources (*Resource* A - D) and their properties (*property*).

The arrows between two modules go from left to right and show the processing order of the modules. In Figure 5 first Module I, then Module II and at the end Module III is processed. Module I shows the information that is available at the beginning of the decision process and is therefore the result of step 2 of the definition process introduced before. Module II is a result of step 4 and represents the preliminary information resulting from the combination of the information in Module I. Module III shows the final result which was defined in step 3 and which is used for the concrete adaptation done by the policy enforcement point.

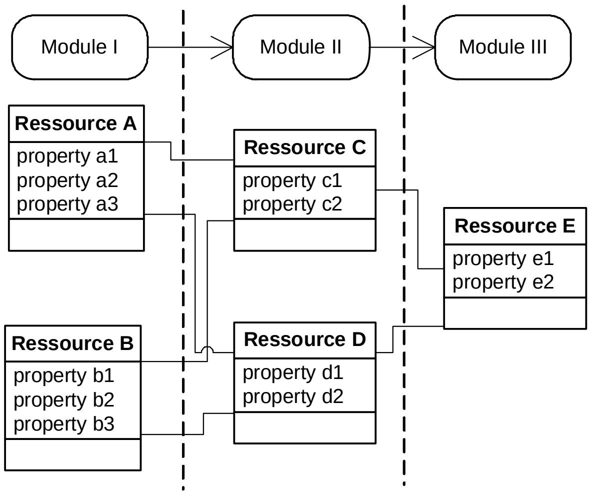


Figure 5. Core elements of the diagram

Policies calculate new context information based on existing information. This is indirectly visualized by the lines which connect resources of different modules. Hereby all lines that connect resources in a left module to a resource in a right module visualize policies that use the resources in the left module to calculate the information that is represented in the right module.

5.2. Module Pipeline

Common problems of policy based adaptive systems are computability, conflict detection and resolution, complexity and performance. We address some of these issues through the usage of our module concept. Basically it is a possibility for the developer to structure a potentially big set of policies into modules. Through this *divide and conquer* strategy it is possible to concentrate on only few policies during a specific time.

One other very important problem is cycles during the execution of policies. If for instance the policies *A* and *B* react on a change of the resource *c* and both policies change this resource then *A* and *B* run infinitely. This effect is usually undesired and needs a lot of processing power. This problem can be solved using modules that define an execution order. If for instance the two modules *I* and *II* are defined then first all policies in module *I* must be fired before a policy in module *II* can be fired. The disadvantage of this is that the parallelism of the execution of policies is restricted. But it is still possible that all policies in one module can be executed in parallel.

Often a policy should only be fired if a specific set of information is available. If, for instance, a policy has to select the minimal duration for a data transmission then all possibilities for the data transmission and their duration have to be calculated first.

6. Prototype

To prove our concept we developed a prototype based on the presented architecture, the process for defining context and policies and the scenario which was explained in Section 3. The following Figure 6 depicts the architecture of our prototype taking the elements introduced in Section 4 into account.

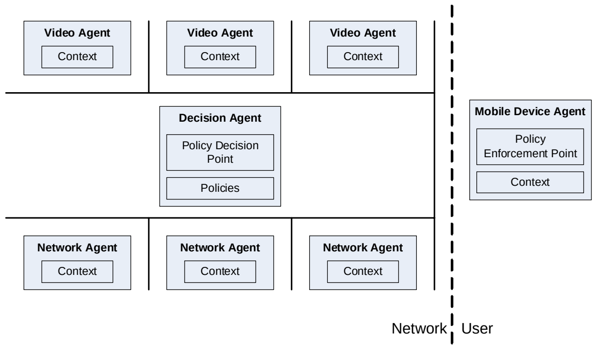


Figure 6. Architecture of the prototype

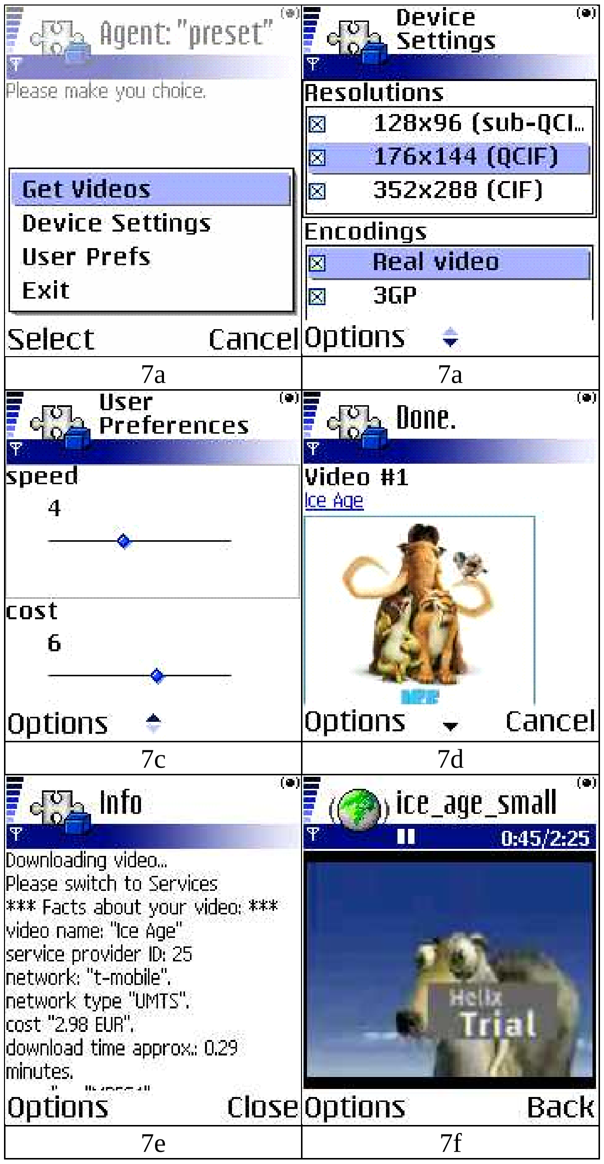
The architecture is divided into the network side and the user side. The network side includes the video provider agents, the network provider agents and a decision agent. The user side includes the mobile phone agent which represents the device as well as the preferences of the user. As already mentioned not every entity of the architecture must include all possible core elements. So the video and network provider agents provide only context information. The decision agent consists of a policy decision point and provides policies. The mobile device provides context information and acts as a policy enforcement point.

All the context information are described in RDF and are translated in the policy decision point by XSL transformations into the Jess syntax for describing knowledge because the Jess library is not capable of processing RDF content directly. All entities of our prototype are realized as Jade – Leap agents. The Lightweight Extensible Agent Platform (Leap) [8] is an extension of the Jade platform which allows developing applications for mobile devices that support the Java 2 Micro Edition (J2ME). As the mobile device we used a Siemens S65 as well as a Nokia 6600 that support CLDC 1.1 and 1.0, respectively, MIDP 2.0 and the Mobile Media API (MMAPI). In our current configuration all agents in the network run on a single PC. Because of the Jade middleware it is no problem to distribute the different agents to different servers or different mobile devices.

As depicted by the screenshots of our prototype in the following Figure 7, the user can do four different tasks (three of them are shown in Figure 7a). First it is possible to change the settings of the device to simulate different mobile devices (Figure 7b). Here it is possible to define the screen resolution as well as the supported video encodings and network interfaces. Furthermore the user can define his or her preferences (Figure 7c) regarding quality, speed and cost on a scale between 1 (unimportant) to 10 (important).

After the user selects the option *Get Videos* (Figure 7a) he or she sees a list of available videos each represented by a title, image and description. This information was requested by the mobile device agent from the video agents that represent the different videos. After the user selects a specific video by clicking on it, the decision agent calculates, based on the policies and different context information, the best combination of video (resolution, encoding, size), network (e.g. UMTS, WLAN or Bluetooth) and network provider based on the user preferences and device capabilities. This information is shown to the user of the prototype (Figure 7e). Afterwards the mobile device agent requests the video and the user can watch the trailer of the movie with the built-in browser of the mobile phone (Figure 7f).

This sequence can be repeated several times whereby the user can change the device capabilities and preferences which lead to other decisions by the decision agent.



```
Downloading video...
Please switch to Services
*** Facts about your video: ***
video name: "Ice Age"
service provider ID: 25
network: "t-mobile",
network type "UMTS",
cost "2.98 EUR",
download time approx.: 0.29
minutes,
```

Figure 7. Screenshots of the interface of the mobile device agent

Because of the complexity of our prototype it is not possible to discuss all aspects regarding the definition of context information and policies. Therefore we focus in Figure 8 only on the available context information and the desired result.

The context information that is already available at the beginning can be grouped in *offered services*, *available networks*, *mobile device*, *user* and *general properties*. The module pipeline consists of four different steps.

First step: This module calculates which videos with which encodings (*offered services*) could be played by the device that supports a specific set of video encodings (*mobile device*). So if the device supports only MPEG-4 and Real Video only these videos should be considered for further calculations. Similar policies estimate which videos with which resolutions can be played by the device and which networks could be used by the device. Furthermore the time is calculated that is needed for the transmission of the video to the mobile devices. Another policy calculates the costs for the user when a specific video is transmitted by a network of a specific network provider.

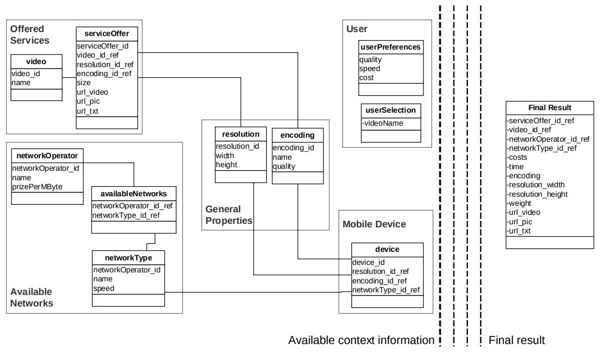


Figure 8. Available context in formation and final result

Second step: All possible combinations of videos, networks, network providers and their cost and transmission time are calculated. Other policies calculate the minimal transmission time and cost of a video.

Third step: The different possible combinations are assessed regarding their correlation to the user preferences time, quality and cost.

Fourth step: The final or best result is estimated which is the combination which conforms best to the user preferences. This final result is also shown to the user (Figure 7e) and defines which video is played (Figure 7f) and which network provider is selected.

7. Conclusion and Future Work

In this paper we presented an architecture as well as a methodology for the development of policy based adapted services for mobile commerce. We developed our architecture and their core elements based on three basic requirements. Furthermore we discussed the usage of RDF to represent context information as well as the usage of Jess as the policy language. Afterwards we depicted the physical architecture and the Jade middleware we use. We propose the combination and adaptation of these matured standards, APIs and middleware from the fields of artificial intelligence, expert systems, semantic web and agent based mobile middleware for further developments of systems for context aware mobile services. We also addressed the issue of the definition of context and policies by introducing a corresponding methodology and diagram as well as the module pipeline concept. We show the feasibility of the proposed architecture and methodology based on a prototype that implements a typical scenario illustrating the advantages and problems of these services.

In the future we will consider the use of Web Ontology Language (OWL) instead of RDF for the representation of context information. We will also work on a user study based on the proposed prototype to find out what people think about it and in particular if people understand the meaning and usefulness of user preferences. Furthermore we plan to use the proposed methodology in several prototypes which will be developed in the Simplicity project [19].

8. Acknowledgement

This work was partially supported by the project IST-2004-507558 Simplicity funded by the EU. The authors wish to express their gratitude to the other members of the Simplicity Consortium [19], to Siemens and Nokia for lending us mobile phones and Paul Holleis, Albrecht Schmidt and Rainer Fink for their kind support.

References

[1] Efstratiou, C.; Friday, A.; Davis, N.; Cheverst, K. Utilising the Event Calculus for Policy Driven Adaptation on Mobile Systems. In 3rd International Workshop on Policies for Dis-tributed Systems and Networks (POLICY'02). Monterey, California, USA. 2002.

[2] Friedman-Hill, E., Jess in Action: Java Rule-based Systems. Manning Publications. ISBN 1930110898, 2003.

[3] Chen, G.; Kotz, D. A Survey of Context-Aware Mobile Computing Research. Technical Report: TR2000-381 Dartmouth College, 2000.

[4] DAML Rules, http://daml.semanticweb.org/rules/

[5] Jena 2 Inference support, http://jena.sourceforge.net/inference/index.html

[6] Abowd, G.; Dey, A.; Brown, P.; Davies, N.; Smith, M.; Steggles, P. Towards a Better Un-derstanding of Context and Context-Awareness. In Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing, pp. 304-307, 1999.

[7] Jennings, R.; Wooldridge, M. Applying Agent Technology. In Journal of Applied Artificial Intelligence Special Issue on Intelligent Agents and Multi-Agent Systems, 1995.

[8] Bellifemine, F.; Caire, G.; Poggi, A.; Rimassa, G. JADE - A White Paper. In Journal TILAB "EXP in search of innovation", September 2003.

[9] Core Information Model, a DMTF (Distributed Management Task Force) Standard, 2004.

[10] Wireless Application Group. User Agent Profile Specification. Open Mobile Alliance WAP Forum, 2001.

[11] Verma, D. Policy-Based Networking: Architecture and Algorithms. New Riders Publishing, ISBN 1-57870-226-7, 2001.

[12] Russell, S.; Norvig, P. Artificial Intelligence: A Modern Approach., Prentice Hall, ISBN: 0-13-790395-2, 2003.

[13] Keeney, J.; Cahill, V. Chisel: A Policy-Driven, Context-Aware, Dynamic Adaptation Framework. In Proceedings of the Fourth IEEE International Workshop on Policies for Dis-tributed Systems and Networks (POLICY 2003), pp. 3-14, 2003.

[14] Lago, P. A Policy-based Approach to Personalization of Communication over Converged Networks. 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02), 2002.

[15] Suryanarayana, L.; Hjelm, J. Profiles for the situated web. In Proceedings of the Eleventh International Conference on World Wide Web, ISBN 1-58113-449-5, pp. 200-209, Hono-lulu, Hawaii, USA, 2002.

[16] Henricksen, K.; Indulska, J.; Rakotonirainy, A. Modeling Context Information in Pervasive Computing Systems. First International Conference on Pervasive Computing. Zurich, 2002.

[17] W3C Recommendation Resource Description Framework (RDF), http://www.w3.org/RDF/

[18] Java Rule Engine API (JSR 94), http://www.jcp.org/en/jsr/detail?id=94

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

[20] Schmidt, A.; Takaluoma, A.; Mäntyjärvi, J. Context-Aware Telephony Over WAP. In journal Personal Ubiquitous Computing. 4(4), pp. 225-229, 2000.