Comprehensive Structured Context Profiles (CSCP): Design and Experiences

Sven Buchholz, Thomas Hamann, and Gerald Hübsch
Department of Computer Science, Dresden University of Technology
{buchholz, hamann, huebsch}@rn.inf.tu-dresden.de

Abstract

In dynamic heterogeneous environments, such as Pervasive Computing, context-aware adaptation is a key concept to meet the varying requirements of different clients. To enable such functionality, context information must be gathered and eventually presented to the application performing the adaptation. Therefore, a common representation format for the context information is required.

This paper introduces a novel representation language for context information: Comprehensive Structured Context Profiles (CSCP). CSCP is based on the Resource Description Framework (RDF) and is designed to be comprehensive and thoroughly structured to describe the entire context of mobile sessions. Besides the design of CSCP, the paper describes our experiences with CSCP in a running system, a Mobility Portal for context-aware adaptive access to email and Web information services.

1. Introduction

An inherent challenge in service provisioning in a dynamic heterogeneous environment, such as Pervasive Computing, is the adaptation of the service and its content to the current context conditions. Context-aware adaptation may be generic or application-specific and may span both network and application layers. Examples for context-aware adaptation are: adjustment of protocol parameters, compression and encryption of payload data, media conversion, transcoding, and information filtering.

Within the scope of a research project targeted at enabling access to corporate email and Intranet information services by mobile users, we have developed a *Mobility Portal* performing context-aware adaptation. It combines application-spanning media conversion and transcoding as well as application-specific information filtering. The Mobility Portal is the central access point for mobile users. By managing and evaluating context information, it allows for personalized, context-aware service provision and content adaptation. In the considered application scenario, relevant context information comprise the capabilities of the client device, transmission characteristics of the network connection, and user specific information. The latter includes viewing and filtering preferences, topics of interests, authentication information, subscriber information, etc. Even though the prototype system is restricted to device, network, and user specific context – combined to the session context profile – the presented concepts are easily extensible to reflect further aspects, such as environmental information (e.g. location, noise level).

The context information of the profile must be described in a common representation language and it must be unambiguously assigned to the client's application-level session to enable adaptive context-aware applications. A context profile representation should be applicable throughout the entire process of context management. Therefore, there are a couple of requirements concerning the representation format. A context profile representation should be: (1) *structured* to provide for natural modeling of context and efficient filtering, (2) *interchangeable* among the different components of the system, (3) *decomposable/composable* to allow for distributed maintenance of the context profile, (4) *uniform* for all flavors of context data (hardware, user, environment, etc.), (5) *extensible* to future needs, and (6) *standardized* to allow context information to be exchanged among entities that do not belong to the same administrative domain.

This paper introduces a novel representation language, called *Comprehensive Structured Context Profiles* (*CSCP*) that is designed to meet the above requirements. Existing approaches for context representation are examined in Section 2. Section 3 depicts the context management aspects of the Mobility Portal architecture. The CSCP language itself is described in Section 4, whereas the CSCP query processing in the Mobility Portal is illustrated in detail throughout Section 5. The final section presents concluding remarks.

2. Previous Work on Context Representation

The predominant approach for context representation in current work is the W3C's *Composite Capability/ Preference Profiles* (*CC/PP*) language [1]. CC/PP is a framework for describing device capabilities and user preferences based on RDF [2], an XML-based meta data description framework. CC/PP defines the basic structure of context profiles and introduces a mechanism for profile decomposability. Whereas CC/PP is interchangeable, decomposable, uniform and extensible, it lacks sufficient structuring. Its strict two-level-hierarchy is not appropriate to capture complex profile structures. Furthermore, CC/PP does not allow for context-sensitive interpretation of attributes requiring globally unambiguous attribute naming.

IETF Media Feature Sets [3] have been designed for protocol-independent content negotiation. They specify device capabilities and user preferences by unstructured attribute/value pairs, which we consider inappropriate. Complex capabilities and preferences are expressed by Boolean expressions of predicates about single attributes. Media Feature Sets suffer another major drawback: they are not decomposable and there are no formal, machine-readable means for extensions.

Besides the standardized representation formats by the W3C and IETF, there are even product and vendor specific approaches (e.g. [4]). Typically, they are tailored to certain kinds of context information, such as user profiles. They are not comprehensive, uniform representation formats for arbitrary kinds of context information.

3. Mobility Portal Prototype

The Mobility Portal (Fig. 1) provides adaptive access to portal services through a Web proxy using a single source publishing approach [5]. Furthermore, adaptive access to email accounts (i.e. adaptation or pruning of attachments) through native email protocols (viz. POP3) is provided by an adaptive email proxy. The actual content adaptation is performed by the adaptation engine.

The adaptive Web proxy and email proxy, the adaptation engine as well as adaptive portal services retrieve information about the context of the client's session, such as user preferences and device capabilities from the context management component (CMC) of the Mobility Portal. The CMC is the central entity for context management. After initial setup of a context profile by the context monitor on the mobile device the context profile is transferred to the CMC using an HTTP-based protocol called the Context Information Exchange Protocol (CIEP). The CIEP is a session protocol defining primitives for session setup including user authentication and transfer of the initial context profile, context profile updates, and session shutdown. Due to the session semantics of the CIEP context profiles do not need to be re-sent during the lifetime of a session. As opposed to that, the CMC stores the context profile in a local profile repository and assigns a Mobile Session Identifier (MSID) with the associated CIEP session. The MSID is negotiated between the CMC and the context monitor on the mobile device during session setup.

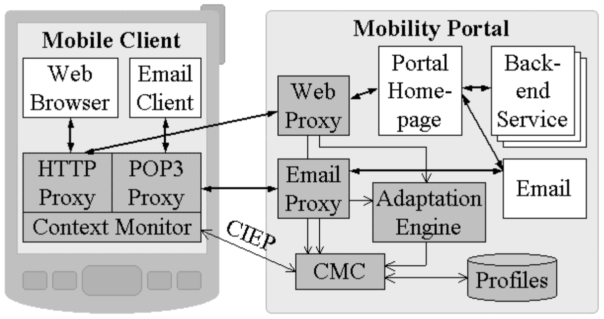

Figure 1. Mobility Portal – System Scenario

CIEP sessions are long lived and may span multiple application sessions, i.e. multiple HTTP requests, WAP or POP3 sessions. In order to identify the CIEP session that an application session is assigned to, the MSID has to be conveyed within the application protocols. Therefore, the mobile client has to be extended by protocol proxies that maintain the session context by inserting the MSID into client requests sent to the portal (cf. Fig. 1). The protocol proxies are configured with the MSID of the current CIEP session by the context monitor.

In the prototype, we have implemented protocol proxies for HTTP and POP3. In HTTP, the MSID is conveyed in the `user-agent` header of each HTTP requests. As POP3 is stateful, the MSID is sent once per POP3 session during POP3 session establishment alongside the mandatory user authentication information. Both protocol extensions are easy to implement and require minimal computational effort on the mobile client, thus yielding almost no performance decrease on state-of-the-art Pocket PCs.

However the proxy solution cannot be implemented on all mobile platforms. This is due to limited multitasking support on platforms running Palm OS 4 or lower as well as on low-end mobile phones. In these cases, the prototype uses the network address of the mobile client as MSID. Although this approach basically works for all protocols supported by the client, there is an inherent problem with WAP 1.x. The WAP 1.x content delivery architecture specifies a WAP gateway between the client and the content server, which is the Mobility Portal in our case [6]. Hence, the Mobility Portal only receives requests sent by the gateway. Thus, it is unaware of the network address of the mobile client. Clients using the same gateway cannot be distinguished by the portal. This problem is solved by an extension of the WAP gateway. It is modified to insert the serialized network address of the mobile client into the HTTP `user-agent` header of the request that is forwarded to the portal. While this approach is transparent to the Mobility Portal it requires a dedicated WAP gateway.

Furthermore, the limited multitasking capabilities of some client platforms renders automatic context monitor-

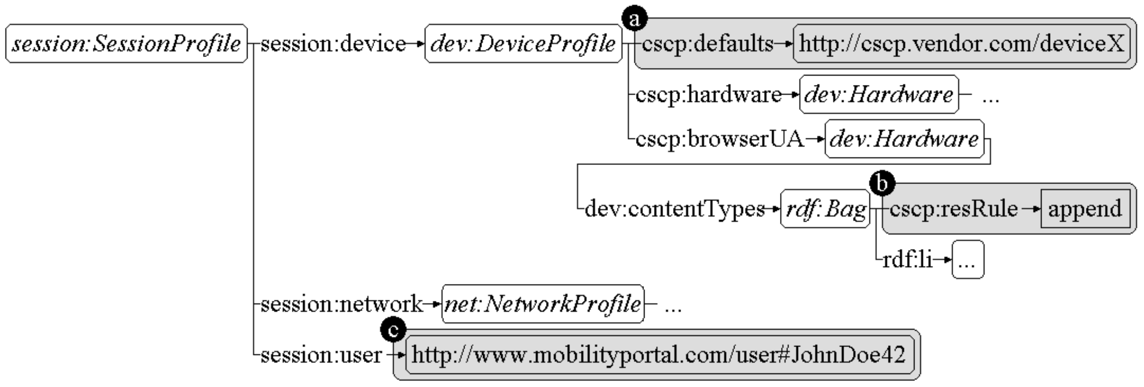

Figure 2. Sample session profile (RDF graph notation [2])

ing impossible. Hence, context updates require user interaction. As opposed to that, session setup and shutdown is always connected with user interaction since the user is required to provide authentication information.

4. Comprehensive Structured Context Profiles

As illustrated in Section 2, existing solutions particularly lack sufficient structuring for complex context profiles. Hence, we propose a representation format that is thoroughly structured and comprehensive to allow for all flavors of context information: *Comprehensive Structured Context Profiles* (*CSCP*). Just as CC/PP, CSCP is based on RDF and thus inherits its interchangeability and extensibility. Yet CSCP overcomes the deficits of CC/PP regarding structuring. It does not impose any fixed hierarchy. It rather supports the full flexibility of RDF to express natural structures of context information. Attribute names are interpreted context-sensitively according to their position in the profile structure. Thus, unambiguous attribute naming throughout the profile is not required.

In the Mobility Portal scenario, context information is stored in CSCP session profiles (Fig. 2). A profile describes all context information relevant to a client's mobile session. It is initially assembled on the client and transferred to the portal during session establishment using XML serialization. The initial assembly of a profile on the client does not mean that all context information must be gathered on the client and transferred via the wireless link. A profile may rather contain references to external resources, such as device defaults (Fig. 2 (a)), which can be retrieved from the device vendor's web site, or the user profile (Fig. 2 (c)) stored by the portal. The CSCP defaults mechanism is similar to the one of CC/PP. Unlike CC/PP, which provides for overriding of default attribute values only, the CSCP mechanism additionally allows to merge profile subtrees with their corresponding default subtrees and thus allows for complex profile structures. The respective defaults semantics of a subtree is determined by the meta property `cscp:resRule` (Fig. 2 (b), Tab. 1+2).

Furthermore, the defaults mechanism is utilized to propagate updates of the session profile. Updates are expressed by a differential profile that refers to the previous profile as its default profile and that overrides attribute values that have changed. By this means, a client does not need to re-send a complete session profile during the lifetime of a session.

In order to extend the capabilities to express preferences, CSCP provides mechanisms to attach conditions

Table 1. Defaults semantics for profile subtrees

`cscp:resRule`	**Description**
`merge` (default)	merging of the new subtree with its corresponding default subtree
`override`	new subtree substitutes the attribute value in default profile

Table 2. Defaults semantics for containers

`cscp:resRule`	**Description**
`override` (default)	new container substitutes the attribute value in default profile
`append`	union with the default container whereas new elements are appended to the default container
`prepend`	union with the default container whereas new elements added at the head of the default container
`intersection`	intersection with the default container
`difference`	difference of default container and new container

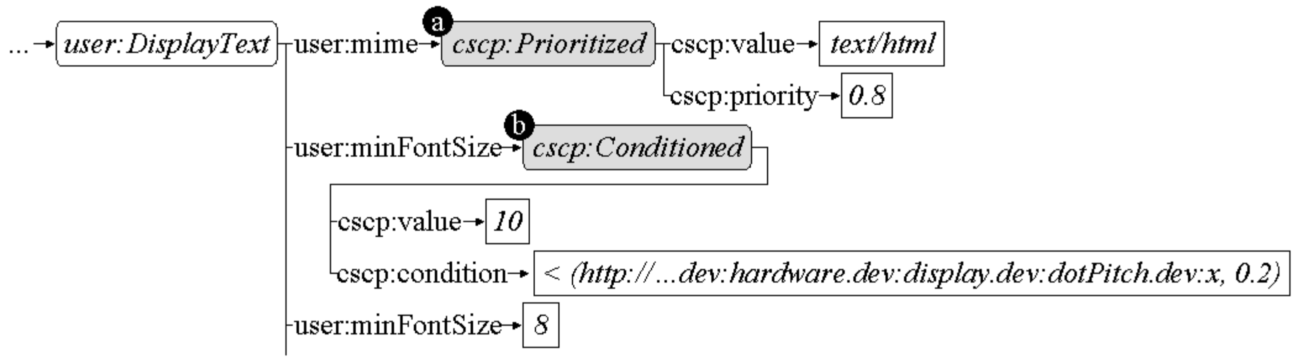

Figure 3. CSCP conditions and priorities

and priorities to attributes. Priorities are applied to resolve potential conflicts between preferences and service capabilities. They are expressed by a decimal number in the interval (0, 1) (cf. Fig. 3 (a)). A maximum priority of 1 means the attribute is indispensable. If no priority is specified, a default priority of 0.5 is assumed.

By CSCP conditions, we allow for preferences to vary depending on other context conditions. CSCP conditions are formulated by expressions over attributes of other entities in the profile that are referenced by CSCP path expressions (cf. next paragraph). CSCP supports numeric, string, and subset comparisons of attribute values within Boolean terms to formulate conditions. Figure 3 (b) illustrates an example of a conditional attribute value. It expresses a minimum font size of 10pt if the dot pitch in the x-dimension is less than 0.2 mm. This conditional value of the minimum font size overrides the alternative attribute value of 8pt that is assumed if the condition is evaluated to false. By means of multiple conditional RDF statements about a single attribute, if-then-elsif-else expressions may be formulated. The conditions of the different statements about an attribute are successively evaluated and the attribute value is set to the one in the first statement that evaluates to true.

CSCP profiles are queried by referencing attribute values of entities in the profile. Attribute values are referenced using CSCP path expressions. A CSCP path expression always starts at a named resource (i.e. root node) identified by its URI. Attributes of this resource are addressed by simply appending a query component to the URI containing the attribute name. As the value of the selected attribute can be a resource itself, attribute names can be chained in a dot-separated list, thus forming a path traversing a sub-tree of the root node of the CSCP profile. Unlike plain RDF, CSCP semantics allows for only a single valid attribute value at a time. Hence, CSCP paths are unambiguous even if there are multiple statements about an attribute (having different conditions).

Besides the CSCP language, we have also defined CSCP vocabularies to express session profiles comprising device, network, and user specific context information. The vocabularies are tailored to the Mobility Portal scenario. Nevertheless, the CSCP vocabulary is easily extensible for future applications using RDF Schema [7].

5. CSCP Processing in the Mobility Portal

Figure 4 shows the CSCP query engine of the Mobility Portal, which is used by adaptive applications to access context information stored in the session profile. The implementation of the profile repository is based on the Jena Semantic Web Toolkit [8] that provides functions to map RDF models to relational databases, a MySQL database in our implementation. Furthermore Jena supports the RDF Query Language for RDF models (RDQL) [9].

The adaptation engine as well as the context-aware portal services request context information from the query engine of the CMC by passing CSCP path expressions to the CSCPNavigator (Fig. 4 (0)+(9)). The CSCPNavigator maps CSCP path expressions to RDQL queries and executes them on the CSCP profiles (Fig. 4 (1)+(2)). Afterwards, the query results (CSCPResult) are passed to the CSCPResultProcessor (Fig. 4 (3)+(8)). It processes CSCP semantics in the RDQL query result, such as conditions, priorities and defaults. If conditional attribute values encounter, they are passed to the CSCPConditionEvaluator (Fig. 4 (4)). This component parses and resolves conditions to Boolean values (Fig. 4 (7)). CSCP path expressions referenced within CSCP conditions are resolved by recursive calls to the CSCPNavigator (Fig. 4 (5)+(6)).

The client-side context monitoring components and the CMC implementation, which is based on Java Servlet Technology, communicate via HTTP.

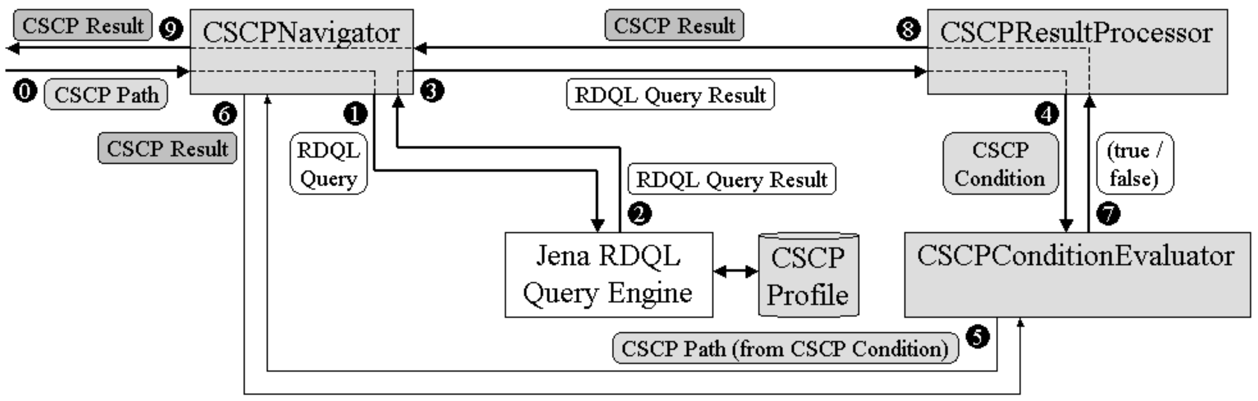

Figure 4. CSCP Query Engine

6. Conclusion

In this paper, we have introduced a novel representation language for context information: Comprehensive Structured Context Profiles (CSCP). CSCP is based on RDF and it is designed to be comprehensive and thoroughly structured to describe the entire context of mobile sessions. CSCP overcomes the deficits of existing approaches regarding structuring. In addition, it provides extended mechanisms to express preferences, viz. conditions and priorities.

Furthermore, we have presented our experiences with CSCP in a running prototype. We have implemented a Mobility Portal providing personalized, adaptive email and Intranet information services for mobile users with adaptation to a substantial amount of context parameters. The described system allows efficient management of context information associated to a client's mobile session including context profile setup, transmission, storage, and utilization. It employs an extensible architecture to allow future support of further application protocols in addition to HTTP, WAP, and POP3, supported by our prototype. Support for client-side context monitoring and management has been successfully implemented for the Pocket PC platform using Sun's PersonalJava [10] and for Palm OS devices and mobile phones using the Java 2 Micro Edition MIDP 1.0 [11], providing for cross-platform compatibility.

Even though the evaluation of complex CSCP paths, conditions, and priorities may be time-consuming, the major performance issues is the actual adaptation procedures not the CSCP processing. Nevertheless, future activities will include a review of profile repository access mechanisms to speed-up CSCP query processing.

CSCP was designed to fit the Mobility Portal scenario. However, it is flexible and open to be generally applicable for context-aware applications.

7. References

[1] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies", W3C Working Draft, 2001.
[2] O. Lassila, R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification", W3C Recommendation, 1999.
[3] G. Klyne, "A Syntax for Describing Media Feature Sets", RFC 2533, 1999.
[4] "4.4.2: Keeping user profiles", in "IBM WebSphere Application Server Version 3.5 Advanced Edition", IBM Corporation. (http://www.ibm.com/software/webservers/appserv/doc/v35/ae/infocenter/was/040402.html)
[5] S. Göbel, S. Buchholz, T. Ziegert, A. Schill, "Software Architecture for the Adaptation of Dialogs and Contents to Different Devices". *Int'l Conf. on Information Networking (ICOIN-16),* Cheju Island, Korea 2002.
[6] "Wireless Application Protocol Specification", Open Mobile Alliance, 2001. (http://www.openmobilealliance.org/wapdocs/wap-210-waparch-20010712-a.pdf)
[7] D. Brickley, R. Guha, "Resource Description Framework (RDF) Schema Specification 1.0", W3C Candidate Recommendation, 2000.
[8] "Jena Semantic Web Toolkit", HP Labs. (http://www.hpl.hp.com/semweb/download.htm)
[9] "RDQL - RDF Data Query Language", Hewlett-Packard Company. (http://www.hpl.hp.com/semweb/rdql.html)
[10] "PersonalJava Application Environment", Sun Microsystems Inc. (http://java.sun.com/products/personaljava)
[11] "Mobile Information Device Profile (MIDP)", Sun Microsystems Inc. (http://java.sun.com/products/midp/)