Profile Processing and Evolution for Smart Environments

Robbie Schaefer[1], Wolfgang Mueller[1], and Jinghua Groppe[2]

[1] Paderborn University/C-LAB
Paderborn, Germany
[2] University of Innsbruck
Innsbruck, Austria

Abstract. Ubiquitous systems use context information to select and adapt multimodal user interfaces and appliances for individual users in certain situations. However, in order to enable true reactive environments, context information has to be adequately collected, filtered, and processed and combined with user, device and other profiles. In this article, we present how an XML-based transcoding system can be applied for advanced profile processing and evolution. We demonstrate how to encode domain knowledge into sets of rules, which perform adaptations of user, device and context profiles for smart environments.

1 Introduction

Context aware computing [2] is a hot topic in the application of ubiquitous computing [18]. As context awareness and natural user interaction requires smart intelligence, this is also often denoted as Ambient Intelligence (AmI) [1]. To implement context awareness, context data retrieved from various sensors and cameras have to be processed by a system and combined with user, device etc. profiles. Context data as given by context profiles are by no means static and can also have different instances. For examples, a person may walk continuously changing current location and may have varying eyesights over years or even months. Thus, in smart systems dynamic and individual aspects must be taken into account when making decisions, for example, to select an adequate output device and to encode content for best presentation to a user.

In this paper, we consider advanced *profile processing* and *evolution*, which denotes the automated modification of profiles in the course of operations and the construction of temporary profiles. In profile evolution, rules are defined based on domain knowledge under which conditions what parts of profiles are modified, deleted, or newly created. In the domain of expert systems [9], this process is known as forward chaining, i.e., defined rules evaluate existing knowledge (facts) optionally creating new facts until a final conclusion is reached. In our application not just the final result is of importance but the history of the rule selection, i.e., the decision process, to explain the decision to a potential user, which is managed by the explanation component in an expert system. We focus on profiles for ubiquitous computing systems and proof-of-concept verification in a smart home environment mainly considering user and device profiles and combine them with temporary existing context profiles. We demonstrate how an XML-based transformation system can process production rules following the principles of an expert system for profile processing and evolution. Example code in this paper is given in the RDL/TT language as its production rules are more comprehensive and can be processed faster compared to XSLT. However, though we use RDL/TT here, same principles also apply to related approaches like XSLT without loss of generality.

The remainder of this article is structured as follows. The next section discusses related work and gives a short overview of basic technologies followed by an introduction to profile based customization of ubiquitous computing systems. The main part elaborates on steps to create a forward chaining profile evolution system. Thereafter, we present examples and evaluation scenarios before concluding our work.

2 Related Work

For related work we focus on different languages and systems for profile description and processing.

For profile description, RDF (Resource Description Framework) [6] is widely applied. RDF is an XML-based format for the description of arbitrary resources through definition of their attributes. Several other formats like CC/PP (Composite Capability / Preference Profile) [7] apply RDF for their specific domain. CC/PP, for instance, is for the definition of general device properties and user preferences. UAProf (User Agent Profile) [19] is for the definition of vocabularies of device-related information and can apply CC/PP. For the exchange personal data, several systems are based on vCard [4] instead. Another general alternative is Dublin Core (DC) [3], a metadata standard with simple value pair elements for the description of networked resources. DC is not based on a specific format, so that also HTML can be used for metadata definition along the lines of the Dublin Core syntax as given in [8].

For profile processing in Web applications, recommender systems [17] implement filtering and processing of user preferences. For instance, Movielens (www.movielens.umn.edu) applies collaborative filtering for recommendations. Hereby, it captures user preferences to automatically generate a profile by rating movies and searches for similar profiles generating recommendations from them. However, in filtering based approaches no explanations can be generated [17]. Other approaches apply content-based technologies based on Bayesian text categorisation with machine learning, e.g., for book recommendations [10]. They analyse the contents of the recommended items and support the generation of explanations.

In the area of general profile processing and evolution, [14] presents a solution based on fuzzy sets, which focus on the evaluation of profiles with no modification of profile data. Sparacino [16] uses Bayesian networks to construct a profile for museum visitors customizing information of stories. [13] presents an approach to merge multiple user profiles in context-aware environment considering conflicts of users' interests. They address properties in the context of smart home environments and focus on filtering user profiles due to current user locations, available services, and technical limitations of terminals. Rentto et al. [12] describe user's attitudes and how they interact with their environment. Their studies demonstrate that seamless integration into the environment as well as correctly locating the user is essential for the introduction of the technology.

We have introduced a fuzzy-based profile processing [14] with rule-based description in RDL/TT [15] for transcoding of XML files. In this article, we introduce a dedicated architecture for context-aware profile processing and outline how our system compares to basic principles of expert systems with respect to rule description and processing [9], where human knowledge is represented as data and rules and rules follow *IF-THEN* patterns and operate on so-called facts. In our applications, facts are given by different profiles and rules refer to more or less complex implications like $rain \rightarrow close\_shutter$. We have developed an architecture based on a specific set of profiles dedicated to context-aware computing in ubiquitous environments. In contrast to experts systems, our approach directly processes the intermediate DOM-tree representation of XML files and are thus most effective and efficient with respect to rule representation and runtime [11]. The execution of our rules compares to principles of *forward chaining* enhanced by fuzzy rules starting with initial facts concurrently executing possibly recursive rules until a conclusion is reached.[3] Additionally, our approach supports explanations of decisions in rule derivations. This is of utmost importance when the approach is applied in smart home environments, i.e., to explain non obvious decisions to a user on demand.

3 Profiles for Context-Aware Computing

We have developed a framework for automated customization of smart home environments. We presume that a smart home integrates multiple distributed embedded devices and sensors, which continuously send data and are to be monitored and controlled. Figure 1 sketches the architecture of our customization system, which is further outlined in the next paragraphs.

A set of pre-defined profiles provides information with user preferences, capabilities, and device properties. The monitoring system captures the environment of the smart home, detects users and devices, and continuously generates a *soft-context* profile from the obtained information. The profile processing system retrieves necessary information from input profiles and optionally applies conflict resolution strategies to handle conflicting information finally generating a customization profile. The evolution system uses the customization profile to operate the devices and to generate recommendations as additional customiza-

[3] Fuzzy rules were introduced because user preferences and capabilities are often given as fuzzy sets in such applications, like $\{high, medium, low\}$.

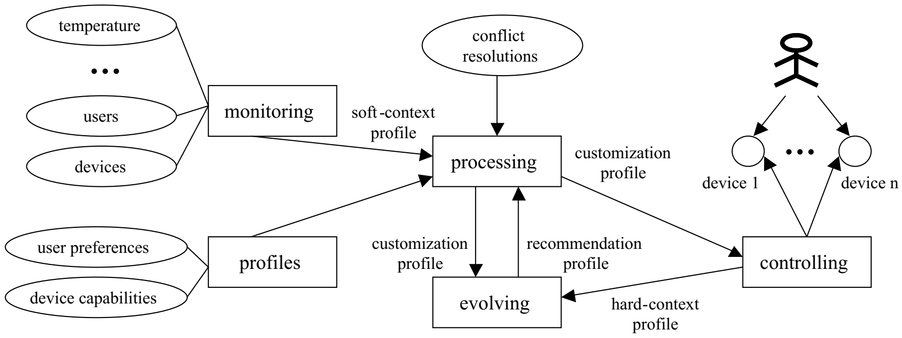

Fig. 1. Customization System Architecture for Smart Homes

tions. Additionally, it monitors the status of the devices by loading *hard-context* profiles.

Users can individually adapt the customization by optionally operating devices manually. The adaptation is captured by the controlling system and is reflected in the hard-context profile. The evolution can capture the manual operation to the environment by comparing the customization profile with the hard-context profile. When new preferences are available, the evolution system filters the preferences and automatically evolves the user profiles.

The customization system manages two categories of profiles:[4] a set of persistent profiles, which contain basic information independent of the concrete context; a set of temporary profiles, which are constructed on-the-fly and mainly keep information related to the current context like location, temperature etc.

Persistent Profiles

– *User profiles* capture users' general data, preferences and abilities. Beyond general user information like name, preferences specify, e.g., the favourite TV channels and the preferred temperature. Abilities describe the information about a user's physical abilities, like eyesight.
– *Device profiles* describe information related to device properties including the software and hardware as well as general information. A device can support different I/O interfaces.

Temporary Profiles

– *Soft-context profiles* have current information from sensors received via the monitoring system from the so-called soft environment, e.g., temperature, humidity, brightness, and locations of people.

[4] We consider a simplified system here. For broader application, we can additionally combine them with network, service, and application profiles.

– *Hard-context profiles* describe the current status of devices, e.g., whether a shutter is close or open, which are controlled and captured by the controlling system.
– *Recommendation profiles* define a set of new settings, i.e., mainly conclusions for user preferences.
– *Customization profiles* are computed from persistent profiles and soft-context profiles and include modified settings.

4 Forward Chaining Evolution

The implementation of a forward chaining profile evolution takes the four steps, which are outlined in the next subsections.

4.1 Merging Profiles to a Working Memory

The evolution system first creates a master profile denoted as working memory. All relevant profiles are merged into that working memory (wm) on which all further transformation and evaluation steps are applied. In the domain of smart homes, for example, one frequently needs to process the combination of user, device and context profiles. In the working memory, each instance of a profile has a unique ID, which allows the processing of replications of individual profiles. The following example, which uses a language neutral type-values XML representation of profiles, shows how they merge into one working memory.

```
<wm>
  <profile id="xyz123" type="user">
    <meta> ... </meta>
    <property type =firstName> <value>Peter</value> </property>
    <property type="lastName"> <value>Pederson</value> </property>
    <property type="visCap"> <value>high</value> </property>
    <property type=location> <value>Room 35</value> </property>
    ...
  </profile>
  <profile id="dvx432" type="device">
    <meta> ... </meta>
    <property type="screenX"> <value>320</value> </property>
    ...
  </profile>
  ...
</wm>
```

4.2 Specifying Production Rules

Our system is based on RDL/TT, which restructures an incoming XML document to a new XML document by means of transformation rules. For each XML element we can define a set of rules in a Java-like syntax. The following example sketches the rough structure of an RDL/TT definition for the previously introduced wm-element.

```
global:
   <initialization code>
wm:
  <rule 1>;
  ...
  <rule n>;
```

RDL/TT rules can be just simple statements, e.g., for renaming elements or more complex programming constructs such as loops and branching. To build a forward chaining system, we branch when a set of conditions evaluates to true as it is given in the next example.

```
wm:
  if (valueOf(xy123.visCap)=="Low")
  ||(valueOf(dvx432.screenX)<= "640")
  {
   // increase font size
   replace(dvx432.property.fontSize.
           content, "20pt")
  }
  else if ....
```

Note here, that the above excerpt is simplified. In order to retrieve the required profiles, we first have to navigate the tree of XML elements to the right identifier and then to return to the parent profile. This can be, e.g., achieved in one statement through a complete path: "valueOf(this.profile.attribute.id.cont ent.xy123.parent.parent.parent.parent.property.visCap)". Since such long paths are quite cumbersome to describe, we are using a pre-processor that expands the id to the required pre- and suffix and reduces the statement to valueOf(xy123.vis Cap). Once the condition is true, changes in the working memory are executed. The above example increases the font size to 20 to increase readability for people with poor visual capabilities.

RDL/TT allows the modification of the structure of the working memory, which corresponds to the creation of new facts for expert systems. This is a quite powerful feature but can also be harmful in practice, since the resulting profiles then might no longer be compatible with the applications. Therefore, complex restructuring has to be applied with care. In order to be on the safe side it might be helpful to restrict on modifying only contents of existing properties. Similar to expert systems, which typically have an explanation component, our system stores explanations in a log file for all successful operations on the working memory.

4.3 Executing Rules

Having defined a rule set in combination with a master profile (working memory) we now can start the transcoder with the rule set and the working memory as parameters. The resulting document is a modified version of the working memory, in case of cascaded if-statements multiple transcoder runs can be necessary. The process terminates when there are no more differences between the input- and output documents. Since we can not guarantee that there are no recurrences in the production rules, an additional upper limit of allowed transcoding runs has to be defined. This number depends on the actual number of rules and the individual application.

4.4 Reconstructing Profiles

After the completed transformations we break up the working memory into individual profiles. For each profile we check for differences with respect to the original profile and, in case of changes, those are marked in a history file because

1. some anomalies might have occurred during processing or an unlikely result has been generated. In such a case, the user or system has to restore the original profile and additionally ask the explanation component why the changes were made. As a consequence, the production rules can be a subject of improvement.
2. the history of changes has to be processed by another component to learn from past behaviour and predict future settings in order to adapt automatically to the user's preferences.

5 Examples and Evaluation

Our main target applications are in the home automation domain. In this context the software is currently evaluated in our Ambient Computing Laboratory (AC-Lab), which is equipped with several sensors as well as means to remotely control different appliances such as lights, shutters, displays etc. In this context we consider a realistic scenario, which can already be partly controlled by our current software.

5.1 Scenario

Consider that it is 5pm and due to previously collected data, the ambient intelligence system concludes that a person - let us call him David – will be at home in half an hour. At or just before the expected arrival time, the smart home system starts with David's preference settings. One action is, for instance, to match the current indoor temperature with the preferred value. Since the temperature is about 18 degrees Celsius and the preferences define 20 degrees Celsius, the heating is turned on. After having authenticated himself at the door terminal, David enters the TV room. The system automatically switches the TV on and checks for David's preferred TV programs in the order of his priorities. Since the sun is standing low and still shining through the windows and since David defined that he likes having daylight at home as long as possible, the system increases the contrast and brightness of the display. However, since

David considers the maximum contrast as uncomfortable, he manually selects a lower value. Later, the sun sets and as his preferences define that he prefers more privacy at night, shutters close and some indoor lights are switched on at low levels. As a consequence, TV contrast and brightness are re-adjusted.

Later, David's wife Sarah arrives, so that the smart home system also has to concurrently process her profiles. Since her temperature preferences are two degrees above David's preferred temperature, the conflicting parameters are easily resolved by applying the average value. However, her TV preferences are in principle on a different channel but since David is enjoying a specific show he has always watches in the past history, the system rates David's preference higher than Sarah's. This is a typical situation, which sometimes cannot automatically be resolved by a system and some manual user interaction is often required to step back and reselect the channel. Nevertheless, processing the history of manual decisions can help to provide a list of good recommendations here.

5.2 Rules

This simple scenario shows two important aspects for profile evolution that need the application of rule based systems: (i) combination of different properties and (ii) conflict resolution. Since even simple scenarios require the definition of a considerable set of rules, an advanced generation of new rules with learning facilities are of high importance for the acceptance of such a system. To outline basic principle, we give a short example with rules and facts in pseudo code.

```
R1: IF(current temperature < preferred temperature)
    THEN switch on heating
R2: IF(someone is scheduled to be present)
    THEN retrieve the preference profile
R3: IF(there is a preference profile)
    THEN retrieve the preferred temperature from the profile

F1: current temperature is 16 degree Celsius
F2: David is arriving in 30 minutes
```

Initially, the system checks for rules with true conditions, given the two facts (F1 and F2). The only matching rule is rule R2, which is selected and fires. The action "*retrieve the preference profile*" is an instruction to load the user profile from the profile database "*there is a preference profile*" and add it to the working memory, so that a new fact F3 is added.

```
F3: there is a preference profile
```

In the next phase, the condition of the third rule evaluates to true, since there is a preference profile available. The fired action informs the system to retrieve the preferred temperature data from the preference profile in the working memory, e.g., the profile says that the preferred temperature is 20, so yet another fact is added to the working memory as F4.

```
F4: preferred temperature is 22 degree Celsius
```

The next cycle begins and Rule R1 fires switching the heating on. After the system has automatically customized the environment according the rules and facts, a user might be not satisfied with the result. The reason for that could be that the user's preference has changed over time or a user has temporarily different preferences as she/he prefers a higher room temperature than usual. In order to modify the environment, the user either informs the system about changes or controls the devices directly. Finally, the system recognizes the manual change, updates profiles, and automatically modifies settings.

5.3 Implementation

In order to implement the scenarios, our proof-of-concept lab is equipped with several sensors for determining locations, temperatures, brightness, humidity etc., which help to generate temporal context profiles. The lab is additionally equipped with home automation facilities like electrical shutters and doors. Finally, we have different interaction devices and modalities installed, e.g., graphical user interfaces, speech control, and gesture control. All appliances as well as monitors and entertainment devices can be switched and controlled by our software. The diversity of sensors and devices allow many different configurations. Different users can work with different preferences and capability profiles, so that scenarios can be evaluated in reasonably complex settings.

Our current implementation is based on RDL/TT. Single runs on complexer rule sets with up to 85 rules execute with up to 590ms, where approximately 80% of the processing time is due to file I/O operations and copying and merging of profiles, which is a clear potential for improvements in our current software. This gives promising results so that we expect to keep system reaction below one second, which is completely sufficient for our applications. In our first evaluations, we just have tested single user scenarios. This was mainly due to the fact that we currently have no precise user localisation system available, which can manage more than one user.

6 Summary and Conclusion

In this article, we introduced advanced concepts for profile processing and evolution and investigated how XML transformation systems can be applied as a forward chaining production system. We identified several different types of temporary profiles and demonstrated how to merge them into a working memory, which was processed by RDL/TT production rules finally splitting them to individual profiles again. Though we have demonstrated principle by means of RDL/TT, concepts also apply to other XML transformation systems like XSLT.

In applications we are mainly interested in context aware computing for home automation, where decisions have to be dynamically taken based on different sets of profiles. Though we have identified a runtime bottleneck in our current implementation when processing large data and rule sets, this is not a critical issue as we expect that the software will be sufficiently reactive after little optimizations of file I/Os and operations on internal tree structures. Thus, after that, our approach seems to be a good trade-off between performance and rapid prototyping of context aware applications in Ambient Intelligence scenarios.

References

1. Aarts, E., Marzano, S.(eds.): The New Everyday – Views on Ambient Intelligence. 010 Publishers, Rotterdam, The Netherlands (2003)
2. Dey, A. K.: Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1), (2001) 4–7
3. Dublin Core Home Page. purl.oclc.org/metadata/dublin_core
4. Howes, T., Smith, M., Dawson, F.: *RFC 2425: A MIME Content-Type for Directory Information (vCard Specification 3.0)*. IETF (1998)
5. Kay, M.: *XSL Transformations (XSLT) Version 2.0. W3C Working Draft.* World Wide Web Consortium (2005)
6. Klyne, G., Carroll, J.J.: *Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation.* World Wide Web Consortium (2004)
7. Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M. H., Tran L.: *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation.* World Wide Web Consortium (2004)
8. Kunze, J.: Encoding Dublin Core Metadata in HTML. IETF-RFC2731. www.ietf.org/rfc/rfc2731.txt
9. Levine, R., Drang, D., Edelson, B.: *Artificial Intelligence and Expert Systems.* McGraw-Hill Book Company, USA (1996)
10. Mooney, R. J., Roy, L.: *Content-Based Book Recommending Using Learning for Text Categorization.* Proceedings of the Fifth ACM Conference on Digital Libraries, San Antonio, TX, USA (2000)
11. Plomp, J., Schaefer, R., Mueller, W.: Comparing Transcoding Tools for Use with a Generic User Interface Format. In *Proceedings of Extreme Markup Languages 2002, Montreal, Quebec, Canada.* (2002)
12. Rentto, K., Korhonen, I., Vtnen, A., Pekkarinen, L., Tuomisto, T., Cluitmans, L., Lappalainen, R.: Users' Preferences for Ubiquitous Computing Applications at Home. In *Proceedings of EUSAI 2003*, *LNCS*, Springer (2003)
13. Salem, B., Rauterberg, M.: Multiple User Proifle Merging (MUPE): Key Challenges for Environment Awareness. In *Proceedings of EUSAI 2004*, *LNCS*, Springer (2004)
14. Schaefer, R.: Fuzzy Evaluation of User Profiles. In *Workshop on User Profiling at CHI 2004, Vienna, Austria*, (2002)
15. Schaefer, R., Mueller, W., Dangberg A.: RDL/TT – A Description Language for the Profile-Dependent Transcoding of XML Documents. In *International ITEA Workshop on Virtual Home Environments, Paderborn, Germany* (2002)
16. Sparacino, F.: Sto(ry)chastics: A Bayesian Network Architecture for User Modeling and Computational Storytelling for Interactive Spaces. In *Proceedings of Ubicomp 2003, Seattle, WA, USA*, Springer, (2003)
17. Ujjin, S., Bentley, P.J.: Building a Lifestyle Recommender System. In *Proceedings of the 10th International WWW Conference, Hong Kong* (2001)
18. Weiser, M.: The Computer for the 21st Century. *Scientific American*, 256(3) (1991)
19. Wireless Application Protocol Forum: *Wireless Application Group User Agent Profile Specification.* (1999)