

Towards a Generic Customisation Model for Ubiquitous Web Applications

G. KAPPEL, W. RETSCHITZEGGER

Institute of Software Technology and Interactive Systems, Vienna University of Technology, Austria

and

E. KIMMERSTORFER

Department of Information Systems (IFS), University Linz, Austria

and

B. PRÖLL

Institute for Applied Knowledge Processing (FAW), University Linz, Austria

and

W. SCHWINGER, TH. HOFER

Software Competence Center Hagenberg (SCCH), Austria

Promising application areas like e-commerce and m-commerce require ubiquitous access to web applications, thus providing time-aware, location-aware, device-aware and personalized services. From a software engineering point of view, appropriate modelling techniques are needed being able to address the ubiquitous nature of such applications. This paper proposes a generic customisation model allowing to adapt web application services towards their context as implied by ubiquity. Accordingly, the customisation model comprises a context model and a rule model providing the basic building blocks to specify certain customisations. At the web application's side, so called adaptation hooks are introduced representing the glue for integrating the customisation model with the context-independent web application model. The applicability of the approach is demonstrated by means of several examples in the area of a web-based conference management system.

Additional Key Words and Phrases: Ubiquitous web applications, customisation, context-awareness, context modelling.

1. INTRODUCTION

We are currently facing the need for new web applications enabling ubiquitous access to e-commerce and m-commerce services being characterised by the *anytime/anywhere/anymedia paradigm* (Chakraborty and Chen 2000). Ubiquitous computing was first stressed by Marc Weiser (Weiser 1993), envisioning a scenario where computational power would be available everywhere embedded in walls, chairs, clothing etc. Weiser's goal is to achieve the most effective kind of technology, which is available throughout the physical environment, while making them effectively invisible to the user. In the area of web applications, ubiquity is not seen as visionary in this highly pervasive sense, meaning that computing power is embedded everywhere. Rather, ubiquitous web applications build more on existing technology, in that web access is no longer primarily a domain of browsers based on desktop PCs but more and more done by various commercially available mobile devices. In general, ubiquity offers new opportunities and challenges for web applications in terms of time-aware (Kleinrock 1996), location-aware (Großmann et al. 2001), (Oppermann and Specht 1999), device-aware (Fox et al. 1996), (Rodriguez 2001) and personalized services (Kobsa 2001). This

implies that ubiquitous web applications have to take into account, individually for each user, time and location of access, together with the different capabilities of devices comprising display resolution, local storage size, method of input and computing speed as well as network capacity. Consequently, the fundamental objective of ubiquitous web applications is to provide services not only to people at any time, any where, with any media but specifically to communicate the right thing at the right time in the right way.

Recently, *standardisation efforts* have been undertaken to collect requirements and provide representation techniques with respect to certain ubiquity issues particularly focusing on device independency and personalisation cf. (W3C 1999) (W3C 2001a), (W3C 2001b), (W3C 2001c). Although, there are already a couple of methods especially dedicated to the modelling of web applications (Retschitzegger and Schwinger 2000), (Barry and Lang 2001), (De Bra 1999), (Frateranli 1999), (Cheri et al. 2000), considering the requirements of ubiquity introduces additional complexity which still needs to be mastered (Gomez 2001), (Kappel et al. 2000), (Kappel et al. 2001c), (McIlhagga 1998), (Rossi et al. 2001). The pre-requisite for supporting ubiquity is that the application is aware of its context (Abowd 1999), (Weiser 1993). When developing ubiquitous web applications, one must understand what context is to determine its relevancy and how it can be exploited for adapting the provided services towards this context. This paper proposes a generic customisation model allowing to adapt web application services towards their context as implied by ubiquity.

The paper is structured as follows. Section 2 introduces the overall architecture of our approach to customisation modelling. Section 3 presents a context model, clearly separating between physical context and logical context. Generic models for specifying customisation rules are proposed in Section 4, together with a concept for their integration into the context-independent web application model. Finally, Section 5 summarises our approach and points to future research.

2. OUR APPROACH TO CUSTOMISATION MODELLING

As already mentioned, existing approaches often focus on particular aspects of ubiquity, such as personalisation or device-independence (W3C 1999), (W3C 2001a), (W3C 2001b), (W3C 2001c). Our approach is based on a broad view of customisation uniformly considering *personalisation* aspects, together with issues resulting from the *anytime/anywhere/anymedia paradigm* (Kappel et al. 2001b), (Kappel et al. 2001d), (Finkelstein 2002). To accomplish this we base on a reflective architecture as depicted in Figure 1. The *context* provides detailed information about the environment of a web application and the web application itself. Thereby the context influences not only the requirements as gathered by requirements elicitation but also triggers the actual customisation as soon as the context changes. The context is divided into a *physical context* representing the level of environment sensors and the *logical context* representing abstracted information. A rule-based mechanism in terms of *customisation rules* is employed in order to specify the actual customisations. Customisation rules are, in turn, determined by requirements. To ensure separation of concerns between the primary service requirements and the requirements resulting from ubiquity, the application is divided into a *stable part*, comprising its default, i.e., *context-independent* structure and behaviour and a *variable, context-dependent part*, thus being subject to most of the adaptations. For a detailed description of the architecture the reader is referred to (Finkelstein 2002).

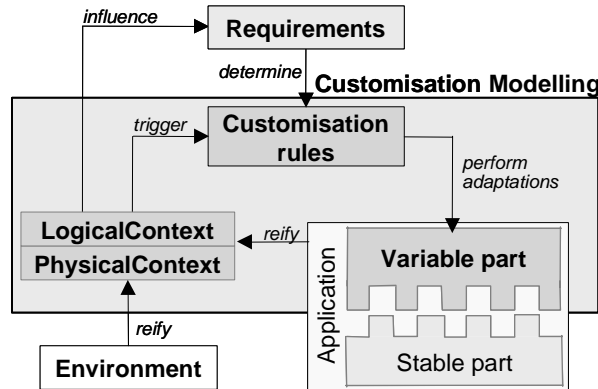


Fig. 1. Overall Architecture of Customisation Modelling

Generic Customisation Model. To support the architecture introduced in Figure 1, we propose a *generic customisation model* in the sense of an object-oriented framework, which provides the customisation designer with appropriate model elements. Generic means that the model is application independent and provides some pre-defined classes and language constructs in order to model application dependent customisation. In addition, the pre-defined classes can be extended by the customisation designer through sub-classing, in order to cope with application specific details. The generic models comprising a context model together with a customisation rule model and several sub-models are discussed in more detail in the forthcoming sections.

Customisation Modelling vs. Traditional Application Modelling. It has to be noted that customisation could also be made part of traditional application modelling without explicitly considering customisation rules, context and the variable part of an application separately from its stable part. By means of factoring out these aspects from the application already during the early stages of the software lifecycle, however, the dynamic nature of ubiquitous web applications can be much better dealt with, not least with respect to reusability and locality of changes (Abowd 1999). The rationale behind an explicit customisation model is similar to the motivation given for factoring out business policies from object-oriented applications (Kappel et al. 2001a).

3. MODELLING CONTEXT

Based on a survey of existing approaches (Kappel et al. 2000), we define context as the *reification of certain properties*, describing *the environment of the application* and *some aspects of the application itself* which are necessary to determine the need for customisation. Please note that, this paper focuses on the environment of the application only thus the reification of the application is not considered in the following. As already mentioned, regarding the user as being part of the context of the application and thus monitoring physical context properties relative to the application like done in, (Alatalo and Siponen 2001), (Mohan et al. 1999), and (Nagao and Hasida 1998) is different to some other approaches which monitor them relative to the user itself, cf., e.g., (Abwod 1999), (Aragao et al. 2001), and (Cherniack et al. 2001). We adhere to the application centric perspective thus avoiding to bias any of the considered context properties. According to the level of abstraction where these context properties are situated, our context model is separated into a *physical context model* and a *logical context model*, which both are described in more detail in the following.

3.1 Physical Context Model

Physical context properties are at a very low level of abstraction and are continuously, dynamically updated to take into account the fact that the environment and the application state itself continuously changes. It has to be noted that, sensing of context information and updating the properties of our physical context model are outside the scope of customisation modelling. It is assumed that, each time the context properties are sensed, the new context property values are made available automatically to our physical context model. For us, physical context is *not modifiable* and so is outside of the control of the ubiquitous web application. It just provides a manageable description of the environment and the application so that it can be addressed within customisation modelling. Although the physical context is application independent, i.e., generic, the customisation designer is able to specify which properties are of interest for a particular application.

Based on existing literature and considering what is currently made available by existing technology, our physical context model, which is shown in Figure 2 by means of a UML class diagram, considers a certain set of properties (cf. class `PhysicalContextProperty`). Since the list of physical context properties is virtually unlimited and certain applications would potentially require additional ones, the model can be extended with additional context properties by means of sub-classing. Inspired by (Schmidt 1999), the properties are structured into *natural context* (`NaturalContext`), *technical context* (`TechnicalContext`) and *social context* (`SocialContext`).

Natural Context. The natural context comprises the context properties *location* and *time*.

- *Location:* Location (`Location`) copes with the need for mobile computing and location-aware services and captures information about the location from which an application is accessed. Actually, this information is not directly provided by mobile devices themselves but is obtained via a so-called location server. Currently location context will be mainly available as `cellId` (`cellID`).
- *Time:* The context property time (`Time`) allows to customise the application with respect to certain timing constraints such as opening hours of shops or timetables of public transportation. Currently the time context property is assumed to be represented at the server only.

Technical Context. The technical context includes information about the *user agent*, the *network* and the *application* itself.

- *User Agent:* The user agent (`UserAgent`) property refers to the demand of ubiquitous web applications for anymedia in terms of multi-delivery. It provides basic information about the device (`Device`) and the browser (`Browser`) by means of a `deviceID` and a `browserID`.
- *Network:* To allow customisation on basis of the network it is necessary to provide network context information (`Network`) in terms of, e.g., bandwidth or package losses.
- *Application:* Additionally to the previously stated context properties, all of them representing information about the environment of the application, it is also required for customisation purposes to get information about the *state of the application* itself. This is captured by the class `ApplicationState` which aggregates a couple of classes depicting the state of the application at each design level.

Social Context. Finally, the social context holds information about the *user* accessing the application.

- *User:* The knowledge about the user (`User`) takes into account the necessity of personalisation. The mechanism which can be used for identifying a certain user mainly depends on the device used for accessing the application. In case of a mobile phone the telephone number can be used, whereas in case of a PC, either the IP-address would be feasible or the user has to provide an explicit identification. Thinking of more advanced applications, user identification could be also done by other mechanisms such as finger prints or other biometric data. Within our physical context model, an attribute `userID` is provided, independent of the actual identification mechanism used.

Considering the UML diagram of our physical context model given in Figure 2, the physical context, represented by the class `PhysicalContext`, is an aggregation of the properties described above. For this, we do not aggregate the generic abstract class `PhysicalContextProperty` but rather the specific, concrete subclasses. We chose this representation because we prefer to have a context vector that explicitly represents the specific subclasses, herewith emphasising the fact, that at a certain point in time, only one instance of each specific context subclass may be present.

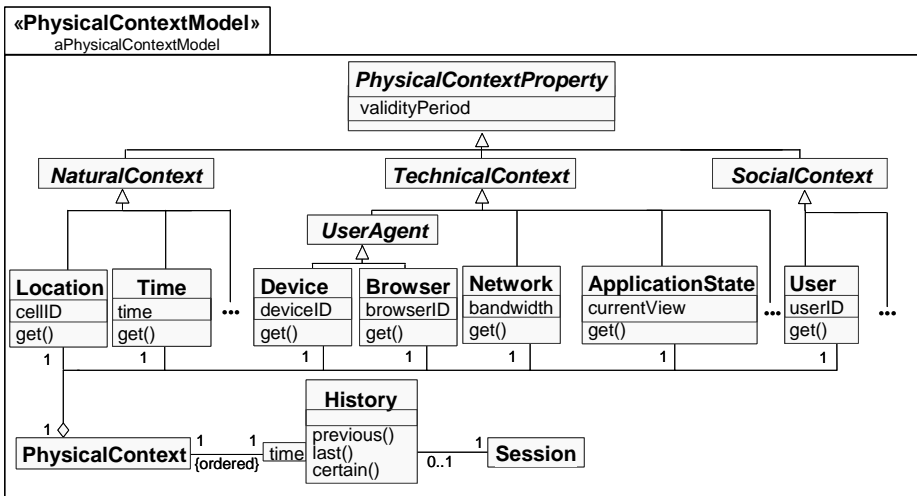


Fig. 2. Physical Context Model

Some general considerations apply for the physical context.

Session. Since web applications enforce the notion of sessions, possibly consisting of a sequence of transactions, the physical context properties need to be considered within the boundaries of sessions, i.e., each session has its own context. Regarding the UML diagram in Figure 2, the class `Session` acts as the root of our physical context model and at the same time holds the collection of all session instances within the system.

Current Physical Context. Furthermore, since the context within a session is subject to continuous changes, it is necessary to identify the most recent context, which is further on called *current physical context*, using the latest timestamp. The current physical context comprises the current values of the physical context properties for a given interaction (e.g., the most recent) within the session of a ubiquitous web application.

Historical Physical Context. Practice has shown that it is useful to broaden the view on context by considering not only the current context at a given point in time but also

historical information. This is necessary to be able to identify changes in the values of the physical context properties over time. Thus the context model also needs to include a *history dimension*, in that a relevant context C can be formulated as a vector of physical context property values over time. Considering our UML diagram in Figure 2, for each session the class `History` holds a sequence of contexts indexed by timestamps. For example since the bandwidth might be constantly changing it might be more important to be able to trace the average bandwidth instead of just having information about the latest one. Also to retrieve information about the travelled distance requires to have all previous positions. Furthermore, to identify user navigation patterns or areas of common interest, it is also necessary to have historical information. In contrast, the information about which device is used allowing to customise the presentation to fit to a restricted display size, requires information of the current device only.

Future Physical Context. Besides historical context information it might be relevant to anticipate possible future states of a context too. For example, concerning video streaming, it is not only relevant how the bandwidth changed in the past, but also how the bandwidth will develop or how stable it can be considered in the future, to be able to tune the resolution of the video accordingly, thus guaranteeing a constant video stream. Another example would be the optimisation of the navigation structure. For this not only knowledge about a user's previous navigation is relevant but additionally the projection how the user will interact with the system in the future. Context projection provides the prerequisite to move from *reactive adaptations* to *proactive adaptations* and is supported within the physical context model using the class `History`.

Availability. Not all physical context property values might be available all the time. For example, the availability of information about the location obviously depends on the capability of the device used. This rises the question what happens, if the value of a physical context property needed for customising a certain service is not available. In this case, there could either be a *default value* provided, or the user is requested to *provide the value manually* or the service simply cannot be requested. The goal should be, of course, to design the application to be robust enough, in order to deal with missing context property values, in the most extreme case, to run without getting any physical context information (Badrinath et al. 2000).

Dynamicity. As already mentioned in the previous section considering the frequency of changes, the physical context may be either *static*, i.e., determined once at application start up, without considering any further changes or *dynamic*, i.e., determined at runtime, every time the context changes within a session. Regarding static context, the device used to access a ubiquitous web application can be identified prior to the usage of the application, but changing the device at runtime is not considered. An example for dynamic context would be to consider the change in bandwidth at runtime to adapt the resolution of an image on the fly.

Validity. Context property values may not be valid during the complete period until they are sensed again. In a mobile scenario an example would be that if the location information doesn't change within a certain period, the device might not be online any longer, thus the location information might no longer be valid. Thus it may be required to specify the validity period during which a context property value is valid (cf. attribute `validityPeriod` within the class `PhysicalContextProperty`).

3.2 Logical Context Model

In contrast to the physical context model, our *logical context model* represents more abstract information about the context. Basically, logical context information is needed to

enrich the semantics of physical context information (e.g., a cellID), thus making it meaningful for customisation purposes (e.g., a street name). Thus, for each property which is part of our physical context model, the logical context model provides appropriate logical information in terms of so called *profiles*¹. These profiles are, because of their complexity, organised as a set of UML packages. Similar to our physical context model, the logical context model can be easily extended by the customisation designer to represent additional kinds of profiles necessary for a particular ubiquitous web application.

Before describing these profiles in more detail in the forthcoming subsections, we stick to some general considerations especially helpful when using profile information for customisation purposes or even having to extend the existing profiles.

Abstraction. Logical profile information can be built by applying different kinds of abstraction mechanisms such as *aggregation* and *fusion* (Abowd 1999), (Schmidt et al. 1999). Aggregation is the transformation of physical context property values to aggregated ones, e.g., certain cellIDs relate to the same location in terms of a city, individual users can be classified by means of a user category. Fusion is the mechanism to derive a logical context from a series of values of *different* context properties, e.g., "Vienna at night" is a fusion of location and time context.

Genericity. In contrast to physical context, most logical context information can be separated into an *application specific* part which is needed for a particular ubiquitous web application only and an *application independent* part, being *generic* and thus can be reused. Application independent information may even be provided by external sources, e.g., third-party providers. The structure of such third-party profile information might not be public, but rather accessible through defined interfaces only. An example for such an external source providing logical location context would be a GIS (Geographical Information System). There are already first attempts to develop universal components, providing various physical and logical context information, cf., e.g., the ContextToolkit (Abowd 1999) or the NEXUS project (Roßmann et al. 2001).

It has to be emphasised that, as soon as a profile is enriched with application dependent information, the border between customisation modelling and application modelling gets blurred. It is not always clear, if certain information should be modelled as part of the profile or as part of information design. Consider, e.g., a user profile - should the different actors of an application be modelled as part of the profile or as part of information design? In this respect, it is suggested that, if certain information is also used for other purposes than customisation, it should be modelled primarily as part of information design, and reused within the profiles using the UML import mechanism.

Automation. Whereas physical context information is considered to be provided automatically, logical context information can be provided *manually*, *automatically* or *semi-automatically*. Most often, profile information is given explicitly by, e.g., a user, a designer or the vendor of a device. In case of automatic construction, the profile is computed on the basis of information available in the physical context, another logical context or the application itself. Projection of bandwidth available in the future is an example of information constructed automatically on bases of the history of bandwidth which is part of the physical context. Building user categories on the bases of interaction patterns would be another example (Mobasher 1999). The construction is semi-automatic if automatic construction is assisted by information entered manually.

Availability. This criteria refers to the same issue as physical context availability, i.e., what happens if certain profile information is not available. For example users accessing

¹ Note that the terms *logical context* and *profiles* are used interchangeably in this document.

the web application the first time may not have any user profile representing their preferred navigation style, although a navigation style would be needed for the application. In this case, it holds the same as already mentioned for logical context availability, i.e., providing default values or requesting the user to manually provide the value.

Dynamicity. First of all, profiles represent more stable information than physical context information. Depending on the kind of profile, information may be available already at *design time*, e.g. device characteristics, or, if the profile is application dependent, not before *runtime*, e.g., usage statistics. In the latter case, the customisation designer needs to take care to explicitly provide the information throughout the application runtime. Location profile information about the geography of a country is already available at design time and can be considered as stable. In contrary, the user's preferences may change over time and thus need to be considered during runtime.

Validity. Profile information once provided might lose its validity over time, e.g., opening hours. Thus it is required to specify a time constraint, determining the validity of certain profile information. In contrast to the physical context model, however, an attribute validity period is not predefined for all logical context information since there is no common super-class.

Standardisation. The proper representation of profile information is already subject to standardisation efforts. The World Wide Web Consortium (W3C) is working on a framework for the management of device and user profile information called "*Composite Capabilities / Preference Profiles*" (CC/PP) which is based on the Resource Description Framework (RDF) (W3C 2001a). It specifies how client devices express their capabilities and users express their preferences to the web application server. In addition, it defines recommendations about the content of such profiles. One major goal followed by CC/PP is extensibility, so that new properties can be defined and included in the description and users can overwrite vendor-defined default preferences. Another interesting initiative in this respect is the "*Platform for Privacy Preferences*" (P3P) (W3C 2001c) effort of the W3C and the "*RDF Site Summary (RSS)*" (RSS 2002) specification. Taking such standardisation efforts into consideration is also of great importance, when profile information from third party providers should be incorporated. We build on these standardisation efforts when modelling profile information.

In the following the different profile models are presented in more detail.

3.2.1 Location Profile

The location profile provides a variety of generic information about the political and physical cartography, along with the possibility to specify logical locations which may or may not be application specific. The application independent information also comprises third party profile information through a location service and a geographical information system, represented by the packages `LocationService` and `GIS` respectively. A real world example of such a third party system is the "Nexus" project, a platform providing location information for both, indoor and outdoor applications (Großmann 2001). Regarding our location profile, `LocationService` offers an interface `LocationMapper`, containing the methods `getGeoPos()` and `getCellID()` to convert from a physical `cellID` to an XY-position in space and vice versa. The `GIS` offers two different interfaces. One dealing with the political cartography (`PoliticalCartography`) and another one dealing with the physical cartography (`PhysicalCartography`). The interface for political cartography offers, for example, the method `getCountry()` returning the country and the method

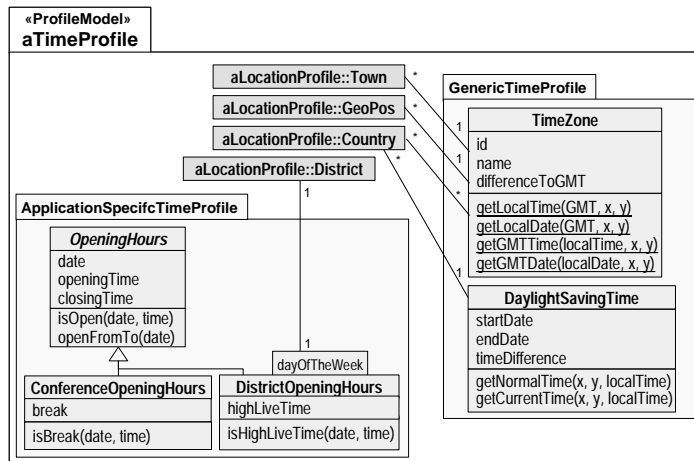


Fig. 4. Time Profile

The application specific part of the time profile comprises information about the opening hours (*OpeningHours*) of both, the conference and the shops and restaurants in certain districts, captured through the classes *ConferenceOpeningHours* and *DistrictOpeningHours*, respectively. For the latter, the location profile class *District* is imported to show the appropriate relationship to location information.

3.2.3 User Agent Profile

In addition to the identification of the hardware and software used when accessing a ubiquitous web application as provided by the physical context, it is indispensable to know about both, their capabilities and their restrictions. Thus, the user agent profile captures device properties (*DeviceProperty*) in terms of hardware (*Hardware*) and software (*Software*) in a fully application independent way. Hardware comprises information about the display capabilities like display size, colour and graphic support, memory, processor speed and connection capability. Software comprises information about the operating system (*OperatingSystem*) and the browser software (*Browser*), specifying its capability to interpret mark-up languages (*MarkupLanguage*), certain media in terms of MIME types (*MIMETypes*) and browser side application logic like JavaScript.

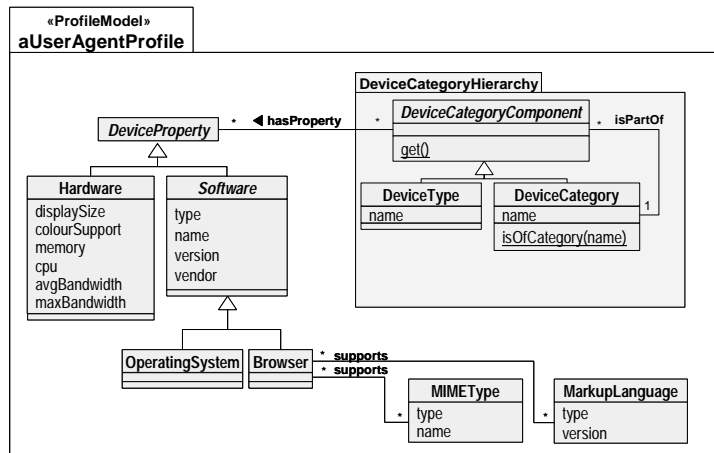


Fig. 5. UserAgent Profile

Furthermore, the user agent profile contains the hierarchy of device categories shown in the sub-package DeviceCategoryHierarchy. Each device category (DeviceCategory), e.g., WAP-enabled mobile phone may subsume a number of other device categories as components (DeviceCategoryComponent) which in turn can be a device type (DeviceType), e.g., Nokia 6120 or again a device category. Consequently, device categories are available at different levels. To check whether a given device is of a certain category the class method isOfCategory of the class DeviceCategory can be invoked taking as parameter the name of a device category.

3.2.4 Network Profile

The network profile provides additional generic information about network connections offered by the different mobile telephone providers such as GMS, HSCSD and GPRS. These different connection types are captured by means of the class Connection. Information about the mobile telephone providers offering these connection types are made available through the class MobileTelephoneProvider.

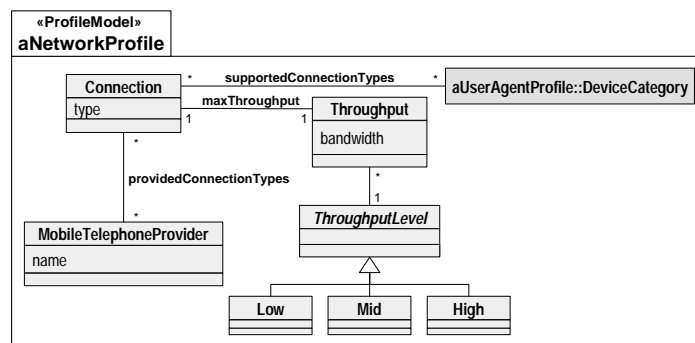


Fig. 6. Network Profile

Each of these connection types has a maximum throughput (Throughput) associated defining bandwidth limitations. The throughput level (ThroughputLevel) can be categorised into different classes, comprising Low, Mid, and High. The network profile could comprise further information about the maximal and expectable connection

speed available for a certain connection type, e.g., GMS is limited to 9600 Byte/s and so forth.

3.2.5 User Profile

A user profile can comprise information that is *voluntarily entered* by the user describing the user's preferences with respect to customisation as well as information that is transparently *acquired* by the system including, e.g., usage statistics or a combination of both (Kobsa 2001). Note that most parts of a user profile are application dependent as opposed to many other profiles. Also the user profile illustrated in Figure 7 consists of a small generic part and a larger application specific part.

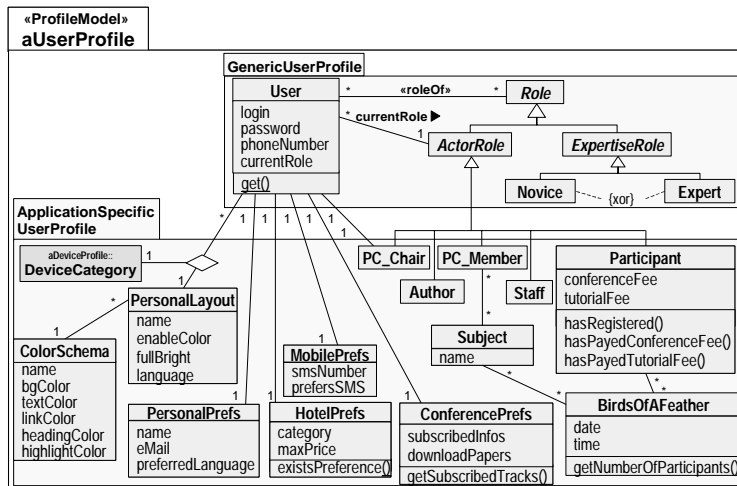


Fig. 7. User Profile

Central to the generic part is the class `User` allowing to associate the individual users accessing the ubiquitous web application as depicted by the physical context with their personal user profile. For this the class `User` offers a class method `get()` which returns an instance of the class `User` according to the physical context `userID` given as input parameter. Each user may play different application independent roles expressing, e.g., the user's level of experience (`ExpertRole`) in terms of `Novice` and `Expert`. Application specific roles can be distinguished with respect to the responsibilities, a user subsumes concerning the application. Considering our conference management system example, such roles (`ActorRole`) are, e.g., `PC_Chair`, `Author`, `PC_Member`, `Staff`, and `Participant`.

Other application specific profile information is related directly to the user and contains special user-specific information like hotel preferences (`HotelPrefs`), conference preferences (`ConferencePrefs`), mobile preferences (`MobilePrefs`) and personal preferences (`PersonalPrefs`). Finally, to be able to personalise the layout of a ubiquitous web application, each user may specify a particular preferred layout for the various device categories. This is represented by the n-ary relationship between the class `User`, the class `PersonalLayout` and the class representing the different device categories, imported from the user agent profile.

4. MODELLING CUSTOMISATION RULES

This section presents the generic customisation rule model and how context information can be exploited in terms of customisation rules.

4.1 Customisation Rule Model

For specifying a certain customisation, we propose the usage of rules (further on called *customisation rules*) in terms of the *event/condition/action (ECA) mechanism*. ECA rules are well known in the area of active database systems (Kappel and Retschitzegger 1998) and represent among others a commonly accepted mechanism for a central specification of business policies as opposed to spreading it over several applications (Kappel et al. 2001a). Although it would be possible to use CA-rules only, we stick to an event-driven specification to better reflect the dynamic nature of ubiquitous web applications. In our approach, the event together with the condition describes, on the basis of the context model, the *situation* when customisation is necessary.

Event. The event part of a customisation rule determines the events which are able to trigger the rule. With respect to our overall architecture given in Figure 1, it detects changes in the context and therefore indicates the need for a potential customisation. An example for an event would be a change of bandwidth. For more details it is referred to Section 4.1.1.

Condition. The condition part is evaluated as soon as the rule is triggered by an appropriate event. It checks whether an adaptation is required and if so, which one is actually desired. An example would be to test whether the bandwidth falls below a certain minimum. Conditions are in fact predicates on the context which can be combined by means of logical operators using OCL-syntax (Rumbough et al. 1998). For enhancing both, readability and reusability, recurring predicates are formulated in terms of *macros*. The specification of the macros can be found in (Schwinger 2001).

Action. The action part of a customisation rule is responsible for activating a certain adaptation of the application, e.g., switch to text mode. An action mainly deals with so called *customisation hooks* as provided by the variable part of the ubiquitous web application (cf. Section 4.2.3.) but can also activate other operations of the stable part of the application which are not explicitly foreseen for customisation purposes.

Rule Properties. Properties of a rule, in terms of *priority* and *activation state* (Kappel and Retschitzegger 1998), may be used to further specify the actual customisation process at runtime. The priority defines the execution order for multiple, possibly conflicting rules. Two rules are conflicting if they are triggered at the same time and their actions try to adapt one and the same model element in incompatible ways. For example, with respect to the user's preferences, one rule may ensure that images are transferred in high resolution mode, whereas another rule should decrease the resolution if bandwidth falls below a certain minimum. In such a case, the corresponding requirements should prescribe a certain precedence, which can be enforced by means of the rules' priorities. Activation state denotes whether a rule can be triggered by an event or not, thus providing, among others, a convenient way for experimenting with different rules, i.e., analysing the impact of different customisations on the system behaviour by temporarily activating or deactivating them at runtime.

In the following, the generic models for a rule's event, condition, and action are described in more detail.

4.1.1 Event Model

Applying the ECA mechanism in the domain of ubiquitous web applications requires a dedicated event model. The events of this event model need to monitor changes within both, physical context information (including changes in the application state, resulting from explicit user requests) and logical context information. Figure 8 shows a UML class diagram of our event model. The event model gives not only an insight into the syntax for specifying certain events but provides also a set of pre-defined event types which can be used by the customisation designer. Since we consider events as first-class objects, the customisation designer is able to extend the pre-defined event types by means of subclassing, thus adhering again to the idea of object-oriented frameworks. This is particularly important to be able to sufficiently monitor application specific context information. Events can be either primitive or composite.

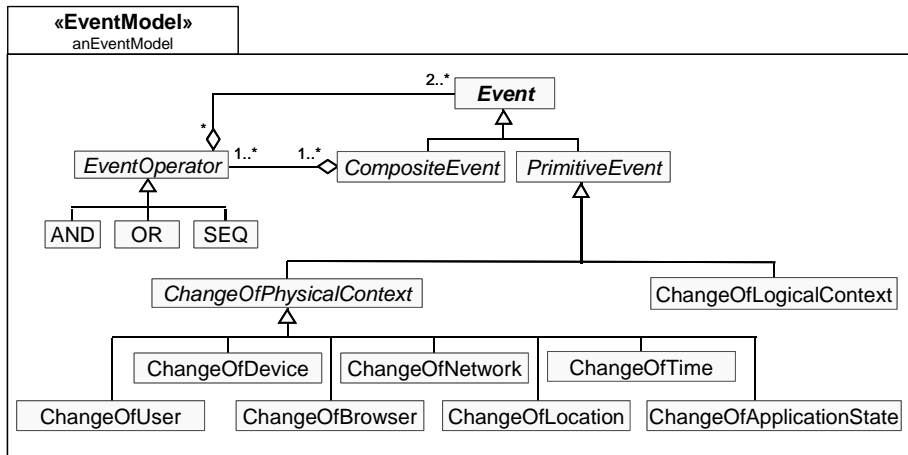


Fig. 8. Event Model

Primitive events indicate changes in the physical or logical context model. Thus, the event model provides the following two basic types of primitive events:

- *Change of physical context:* All events supported for the physical context model monitor changes within the leaf classes of the class hierarchy of this model. It has to be noted that an event of type `ChangeOfApplicationState` may be, in contrast to the other events, signalled by each class in the physical context model which is aggregated by the class `ApplicationState`. Which particular change of application state occurred needs to be identified by explicitly referring to the physical context model within the rule.
- *Change of logical context:* Since the properties of certain profiles are not necessarily limited and can be extended by the customisation designer, we propose a single pre-defined event (`ChangeOfLogicalContext`) monitoring changes in the logical context. To consider the application specific logical context, the customisation designer may extend also this part of the event model by application specific events.

Composite Events. To be able to model complex real-world situations which should be monitored, we suggest the notion of so-called *composite events*. Composite events are constructed by means of the logical event operators AND, OR, and SEQ, using the above mentioned primitive events or again composite events as operands. For example, the requirement that a page should be both personalised and tailored to a certain device demands for a composite event like (`ChangeOfUser OR ChangeOfDevice`).

4.1.2 Condition Model

In general, conditions can be formulated either directly within ECA-rules or (parts of them) can be factored out and specified by a *macro* which can be reused within an ECA-rule (cf. Section 4.2.). We propose the usage of OCL (Object Constraint Language) (Rumbough et al. 1998) for specifying conditions, not least because OCL is part of UML. OCL, originally developed as "business modelling language" by IBM, is a formal language used to express constraints in terms invariant conditions that must hold for the system being modelled, pre- and post-conditions, guards, etc.

It has to be emphasised that when formulating a condition, only a sub-set of OCL, namely *logical expressions*, are used. Such logical expressions allow, among others, the access of attributes and the execution of methods. Both are heavily utilized to combine the physical context with information from the logical context. Although, virtually any method can be invoked to query and retrieve relevant context information, the execution of such methods, triggered through the evaluation of the enclosing expression, needs to be side effect free, i.e., may not alter the state of the application.

4.1.3 Action Model

The actual adaptations which are required by customisation are specified within the action part of an ECA rule. The generic action model giving an overview about the syntax used for specifying actions as well as the potential adaptation operations is depicted in Figure 9. Note that, for readability reasons, this diagram does not show the complete syntax used for action specification (terminal symbols are, for example, omitted). Basically, the action part may contain a list of actions to be performed (cf. class `ActionList`), comprising at least one action (`Action`). Such an action can be an atomic action (`AtomicAction`) or a block action (`BlockAction`). Atomic actions can be either an assignment of a value (`Assignment`) or the invocation of an adaptation operation (`Operation`). Concerning the latter, this can be either the invocation of an application specific adaptation operation (`ApplicationSpecificAO`) or the invocation of a generic adaptation operation (`GenericAO`). Application specific adaptation operations need to be specified explicitly by the customisation designer whereas generic adaptation operations can be pre-defined on bases of the underlying web modelling method, i.e., for each of the provided modelling concepts (e.g. links can be enabled, hidden or disabled).

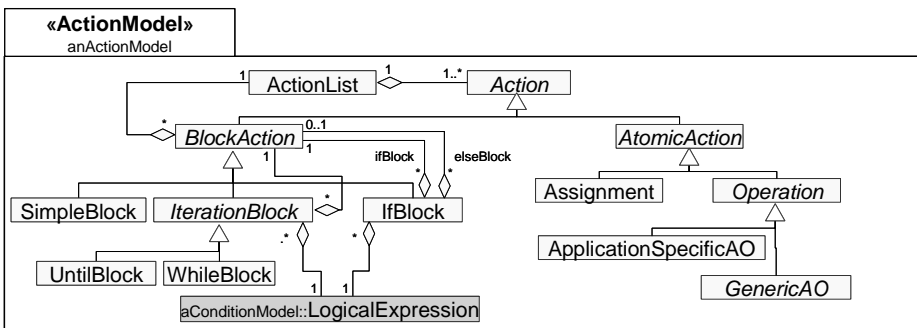


Fig. 9. Action Model

Regarding block actions they can be a simple block (`SimpleBlock`), an if-block (`IfBlock`) allowing to perform conditional execution of actions or an iteration block (`IterationBlock`). Each type of block action, except the simple one, has a condition

associated determining which `BlockAction` of an if-block to execute or determining the iteration of an `UntilBlock` and `WhileBlock`. The condition is specified according to the `ConditionModel` in terms of a logical expression. For this, the class `LogicalExpression` is imported from the `ConditionModel`.

4.2 Adding Customisation Rules to the Web Application Model

Customisation rules are specified within an UML annotation, using the stereotype «`CustomisationRule`» (cf. Figure 10). The specification of a customisation rule comprises, according to the rule model, a unique name, a reference to one or more requirements for being able to trace customisation rules back to their requirements and an ECA-triplet. Each of these components is given a separate section within the annotation.

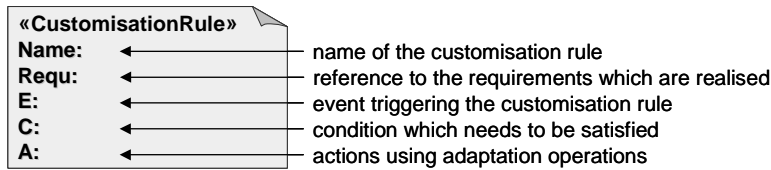


Fig. 10. Stereotyped Annotation for Specifying Customisation Rules

The annotation is attached to those model elements being subject to customisation, i.e., providing the *hook* used by the action (cf. Section 4.2.3.). Thus the customisation rules can be attached to any web application modelling method utilizing UML as the basic formalism.

In the following, it is shown how our approach to customisation modelling can enhance web application modelling with W2000, a web application modelling method developed at the Politecnico di Milano (Baresi et al. 2002)². Customisation rules extend web application design with W2000 in that customisation rules may be attached to model elements within all web application models provided by W2000, covering Information Design, Navigation Design, Presentation Design, and Operation Design, alike. The examples given in the following origin from a conference management system.

4.2.1 Device-Dependent Customisation Example

The example depicted in Figure 11 shows a device-dependent customisation rule, which is attached to the corresponding W2000 model elements within Navigation Design. In particular, the example comprises a small fraction of the Navigation Design, modelling the Navigation Cluster for a PC-member of the conference management system example. The Navigation Cluster contains the available nodes, holding information about PC-members, i.e., a short description (cf. `ShortDescription`), a full description (cf. `FullDescription`), a reviewer profile (cf. `ReviewerProfile`) and the links in-between.

² Note that W2000 is currently undergoing some redefinition. Thus, some of the graphical notation may no longer reflect the most recent version of W2000's development.

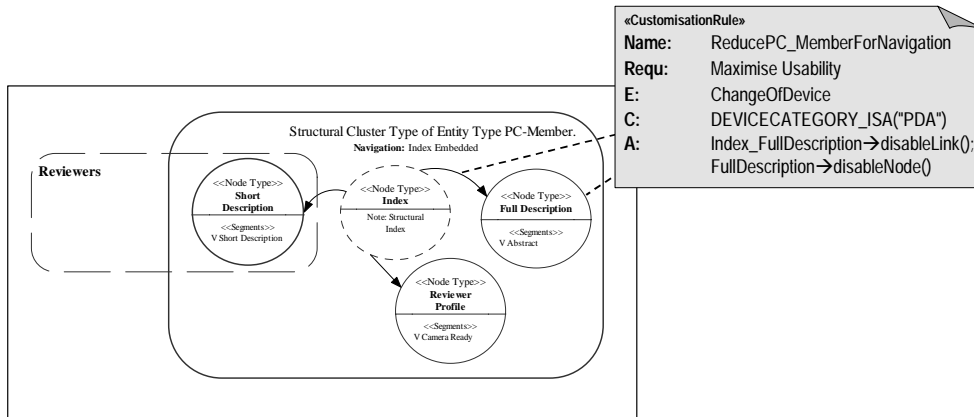


Fig. 11. Device-Dependent Customisation within Navigation Design

The customisation rule `ReducePC_MemberForNavigation` is responsible for reducing the navigation possibilities according to the device used in order to fulfil the requirement of maximising usability. For this, the event part of the rule incorporates the pre-defined event type `ChangeOfDevice`. The condition checks whether the device used was a "PDA", by using a macro called `DEVICECATEGORY_ISA`. This macro returns true, if the currently used device is of kind "PDA" as given in the parameter. The action uses the adaptation operations `disableNode()` and `disableLink()` to reduce navigation by disabling the node `FullDescription` and disabling the link `Index_FullDescription`, respectively. It has to be noted that, additional rules would be needed to cover other device types.

4.2.2 Location-Dependent Customisation Example

The following location-dependent customisation rules realise the requirement that only the equipment of the lecture room, a user is currently in, is presented. For readability reasons, just the customisation rules are illustrated in the following, omitting the underlying W2000 models. According to the point in time when customisation is initiated, this customisation as depicted in Figure 12 can be accomplished through filtering the list of all equipment according to the lecture room before it is actually requested by a certain user.

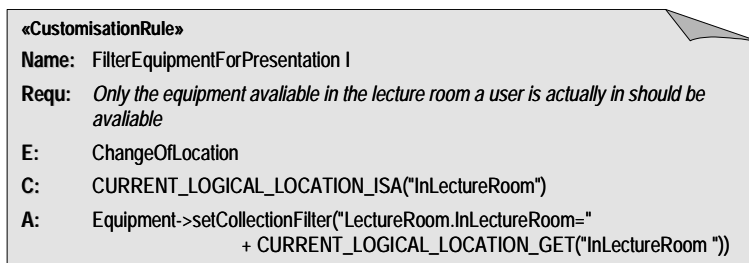


Fig. 12. Location-Dependent Customisation

This customisation rule is triggered every time the location of the user changes (cf. event `ChangeOfLocation`), thus indicating a potential movement from one lecture room to another. The condition checks whether the current location is within a lecture room by using the macro `CURRENT_LOGICAL_LOCATION_ISA()`. Note that, to

simplify things, this customisation rule relinquish the check whether a change of location actually brought the user into the same lecture room again. Thus it could be, that the adaptation is done unnecessarily in case the user steps in and out or moves within the same lecture room. The collection of all available equipments (cf. `Equipment`) is filtered according to the current lecture room by applying the generic adaptation operation `setCollectionFilter()`, taking as parameter a logical expression, checking for the current lecture room. The current logical location is determined through the macro `CURRENT_LOGICAL_LOCATION_GET`, returning an instance of the class `LogicalLocation` as mentioned above. It has to be emphasised that this customisation rule does not indicate when the reduced information will be consumed. This is dependent on the realisation of the stable part of the web application.

4.2.3 Specifying Adaptation Hooks

As a major prerequisite for realising customisation we propose that the application should provide appropriate *adaptation hooks*, herewith again resembling a basic idea of object-oriented frameworks. These adaptation hooks provide the basis for customisation rules to adapt the application towards a particular context by using adaptation operations. As already mentioned, the application is separated into a *stable part* that provides the basic functionality, and a *variable part* that provides adaptation hooks. Depending on the granularity of adaptation, different kinds of mechanisms are used for defining adaptation hooks.

Micro Adaptation. For modelling micro adaptation, we propose that the UML model element `class`, representing a basic model element in W2000 should be extended with an additional compartment for representing adaptation operations in terms of adaptation hooks (cf. Figure 13). Generic adaptation operations, are defined at the meta-model level of W2000, application specific ones can be incorporated directly into concrete W2000 models.

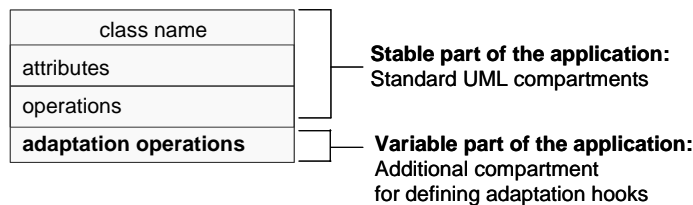


Fig. 13. Additional Class Compartment for Specifying Adaptation Hooks

Macro Adaptation. Concerning macro adaptation, we propose to use the *UML package mechanism* in order to group those model elements which are subject to the same macro adaptation within so called *adaptable packages* (using the stereotype `«AdaptablePackage»`, cf. Figure 14). Since packages allow to group an arbitrary number of arbitrary UML model elements together, the granularity of adaptation is not limited. Similar to classes providing adaptation hooks for micro adaptation within additional compartments, adaptable packages provide adaptation hooks within a pre-defined so called *adapter interface* (using the stereotype `«AdapterInterface»`, cf. Figure 14). Since for plain UML packages the specification of interfaces is not allowed, we use in fact a special kind of package, namely *UML subsystems* which are denoted by the fork symbol (cf. Figure 14).

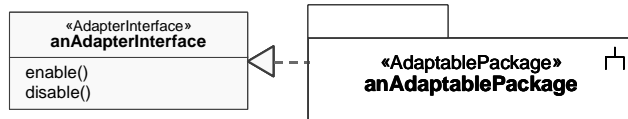


Fig. 14. Adapter Interface and Adaptable Package

This adapter interface centrally provides adaptation operations which again are delegated to the appropriate classes of the adaptable package. In that way, the adapter interface builds on adaptation hooks defined during micro adaptation. Besides generic adaptation operations such as `enable()` and `disable()`, the customisation designer is able to specify application specific adaptation operations, e.g., `changeLanguageTo()`.

Figure 15 gives an example of a customisation rule applying macro adaptation. The customisation rule ensures that a user perceives an adapted design of the application according to her/his current role.

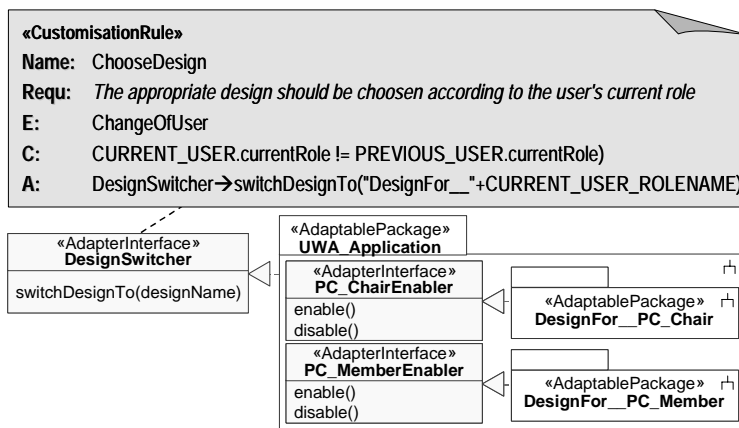


Fig. 15. Macro Adaptation Choosing Between Different Statically Adapted Designs

The customisation rule in Figure 15 is triggered each time the user changes. In case that it is detected that the user's role has changed, it is switched to the adapted model corresponding to the user's current role (cf. `CURRENT_USER_ROLENAME`). To be able to switch between models each of the adapted variants is placed within an adaptable packages (cf. `DesignFor_PC_Chair` and `DesignFor_PC_Member`). Furthermore, each of the adaptable packages again placed within another adaptable package, namely `DesignSwitcher`. The `DesignSwitcher`'s adapter interface offers the adaptation operation `switchDesignTo()` allowing to select one of the adapted model by triggering the corresponding `enable()` and `disable()` adaptation operations of the embraced sub-packages corresponding to the current user's current role (cf. logical user context).

5. SUMMARY AND OUTLOOK

In this paper we have introduced customisation as uniform mechanism for dealing with ubiquitous issues in web application development. The approach presented allows to focus on the task of customisation modelling, thus ensuring separation of concerns with respect to modelling the core functionality of a web application. For this a reification of

the environment in terms of a physical context model and a logical context model is proposed. Customisation rules allow the designer to specify which adaptation to perform if certain context is detected. Customisation modelling is integrated into an existing web application modelling method, namely W2000, by means of attached UML-annotations and adaptation hooks. A series of examples demonstrate the applicability of our approach.

Though the presented approach gives the designer a means to consider ubiquity for web applications, appropriate tool support in applying the approach is needed. To ease the task of the designer we are currently implementing an integrated modelling environment in terms of a *customisation toolkit*. In particular, this customisation toolkit should provide a set of *graphical editors* for defining and maintaining customisation rules, a *customisation rule browser* for facilitating the reuse of already existing customisation rules and an appropriate *repository* for managing the *rule base* together with a *macro library*.

ACKNOWLEDGMENTS

This work was partially funded by *UWA (Ubiquitous Web Applications)*, an EU-funded Fifth Framework Programme project (IST-2000-25131) and *CustWeb (Customizeable Web Applications)* a strategic research project of the *Software Competence Center Hagenberg*.

REFERENCES

- ABOWD G. D. 1999. Software Engineering Issues for Ubiquitous Computing, International Conference on Software Engineering (ICSE), Los Angeles.
- ALATALO T., SIPONEN M. T. 2001. Towards the OWLA methodology for development of Open, Web/Wireless and Adaptive hypermedia information systems, A poster proposal submitted to the ACM HyperText.
- ARAGAO V., FERNANDES A., GOBLE C. 2001. Towards an Architecture for Personalization and Adaptivity in the Semantic Web. In *Proceedings of the Third International Conference on Information Integration and Web-based Applications & Services (iiWAS2001)*, Linz, Austria.
- BARESI L., GARZOTTO F., PAOLINI P. 2002. From Web Sites to Web Applications: New Issues for Conceptual Modeling. In *Lecture Notes in Computer Science*, Springer LNCS 1921.
- BARRY C., LANG M. 2001. A Survey of Multimedia and Web Development Techniques and Methodology Usage. In *IEEE Multimedia*, Special Issue on Web Engineering, April.
- BADRINATH B., FOX A., KLEINROCK L., POPEK G., REIHER P., SATYANARAYANAN M. 2000. A conceptual framework for network and client adaptation. In *Proceedings of the IEEE Mobile Networks and Applications (MONET)*, Vol. 5 No. 4, pp 221-231.
- DE BRA P. 1999. Design Issues in Adaptive Web-Site Development. In *Proceedings of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW*, Toronto, Canada.
- CERI S., FRATERNALI P., AND BONGIO A. 2000. Web Modeling Language (WebML): a modeling language for designing Web sites. In *Proceedings of the 9th World Wide Web Conference (WWW9)*, Amsterdam, The Netherlands.
- CHAKRABORTY D., CHEN H. 2000. Service Discovery in the future for Mobile Commerce. In *ACM Crossroads*.

CHERNIACK M., FRANKLIN M. J., ZDONIK S. 2001. Expressing User Profiles for Data Recharging, In *IEEE Personal Communications*, pp 6-13.

FOX A., BREWER E., GRIBBLE S., AND AMIR E. 1996. Adapting to Network and Client Variability via On-Demand Dynamic Transcoding. In *Proceedings of the ACM Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Cambridge, Massachusetts, US.

FRATERNALI P. 1999. Tools and approaches for data-intensive Web applications: A survey, In *ACM Computing Surveys*, Vol. 31, No. 3.

GOMEZ J., CACHERO C., PASTOR O. 2001. Conceptual Modeling of Device-Independent Web Applications. In *IEEE Multimedia*, Special Issue on Web Engineering.

GROßMANN M., LEONHARDI A., MITSCHANG B., ROTHERMEL K. 2001. A World Model for Location-Aware Systems. In: *Informatik*. Vol. 8(5).

KAPPEL G., RETSCHITZEGGER W., AND SCHRÖDER B. 1998. Enabling Technologies for Electronic Commerce, In *Proceedings of the XV. IFIP World Computer Congress*, Vienna/Austria and Budapest/Hungary.

KAPPEL G., RETSCHITZEGGER W. 1998. The TriGS Active Object-Oriented Database System - An Overview, In *ACM SIGMOD Record*, Vol. 27, No. 3.

KAPPEL G., RETSCHITZEGGER W., SCHWINGER W. 2000. Modeling Customizable Web Applications - A Requirement's Perspective. In *Proceedings of the International Conference on Digital Libraries: Research and Practice (ICDL)*, Koyoto, Japan.

KAPPEL G., RAUSCH-SCHOTT S., RETSCHITZEGGER W., SAKKINEN M. 2001a. Bottom-up design of active object-oriented databases. In *Communications of the ACM (CACM)*, 44(4).

KAPPEL G., RETSCHITZEGGER W., SCHWINGER W. 2001b. A holistic view on web application development - the WUML approach, Tutorial notes at *First International Workshop on Web-oriented Software Technology (IWOST'01)*, Valencia, Spain.

KAPPEL G., PRÖLL B., RETSCHITZEGGER W., SCHWINGER W. HOFER T. 2001c. Modeling Ubiquitous Web Applications – A Comparison of Approaches, In *Proceedings of the International Conference on Information Integration and Web-based Applications and Services (iiWAS2001)*, Austria.

KAPPEL G., RETSCHITZEGGER W., SCHWINGER W. 2001d. Modeling Ubiquitous Web Applications - The WUML Approach. In *Proceeding of International Workshop on Data Semantics in Web Information Systems (DASWIS 2001)*, Kyoto, Japan.

FINKELSTEIN A., C. W. SAVIGNI A., KAPPEL G., RETSCHITZEGGER W., KIMMERSTORFER E., SCHWINGER W., HOFER TH., PRÖLL B., FEICHTNER CH. 2002. Customising Web Applications Towards Ubiquity - The Notion and the Issues. In *Proceedings of the W3C Workshop on Delivery Context*, Sophia-Antipolis, France.

FINKELSTEIN A. C. W., SAVIGNI A., KAPPEL G., RETSCHITZEGGER W., KIMMERSTORFER E., SCHWINGER W., HOFER TH., PRÖLL B., FEICHTNER CH. 2002. Ubiquitous Web Application Development - A Framework for Understanding. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, Florida, US.

KLEINROCK L. 1996. Nomadicity: Anytime, Anywhere. In *A Disconnected World, Mobile Networks and Applications*, 1(4).

KOBSA A. 2001, Generic User Modeling Systems. In *User Modeling and User-Adapted Interaction*, Vol. 11.

KUNZ TH., AND BLACK J. P. 1999. An architecture for adaptive mobile applications. In *Proceedings of Wireless 99, the 11th International Conference on Wireless Communications*, Calgary, Alberta, Canada.

MCILHAGGA M., LIGHT A., AND WAKEMAN I. 1998. Towards a Design Methodology for Adaptive Applications. In *Proceedings of The fourth annual ACM/IEEE*

International Conference on Mobile Computing and Networking (MOBICOM), Dallas, TX, USA.

MOBASHER B., COOLEY R., AND SRIVASTAVA J. 1999. Creating Adaptive Web Sites Through Usage-Based Clustering of URLs. In *Proceedings of the 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX)*, Chicago, Illinois, US.

MOHAN R., SMITH J. R., LI: C.-S. 1999. Adapting Multimedia Internet Content for Universal Access. In *IEEE Transactions on Multimedia*, Vol. 1, No. 1.

NAGAO K. AND HASIDA K. 1998. Automatic Text Summarization Based on the Global Document Annotation. In *Proceedings of the Seventeenth International Conference on Computational Linguistics (COLING-ACL'98)*, Montréal, Québec, Canada.

OPPERMANN R., AND SPECHT M. 1999. A Nomadic Information System for Adaptive Exhibition Guidance. In *Proceedings of the International Conference on Hypermedia and Interactivity in Museums (ICHIM)*, Washington, US.

RETSCHITZEGGER W., SCHWINGER W. 2000. Towards Modelling of DataWeb Applications - A Requirements' Perspective. In *Proceedings of the Americas Conference on Information Systems (AMCIS)*, Vol. I., Long Beach California, US.

RODRIGUEZ J. R. ET AL. 2001. Extending e-business to Pervasive Computing Devices - Using WebSphere Everplace Suite Version 1.1.2, IBM Redbooks, International Technical Support Organisation, SG24-5996-00.

ROSSI G., SCHWABE D., GUIMARÃES R. M. 2001. Designing Personalized Web Applications. In *Proceedings of the International World Wide Web Conference (WWW)*, Amsterdam, The Netherlands.

RUMBAUGH J., JACOBSON I., BOOCH G. 1998. The UML Ref. Manual. Addison-Wesley.

SCHMIDT A., AIDOO K. A., TAKALUOMA A., TUOMELA U., VAN LAERHOVEN, K. VAN DE VELDE W. 1999. Advanced Interaction in Context. In *Proceedings of the first International Symposium on Handheld and Ubiquitous Computing (HUC99)*, Karlsruhe, Germany.

SCHWINGER W. 2001. Modelling Ubiquitous Web Applications - Requirements and Concepts, PhD thesis, Johannes Kepler University Linz.

RSS - RDF Site Summary 2002, <http://www.oasis-open.org/cover/rss.html>.

WEISER M. 1993. Some computer science issues in ubiquitous computing. In *Communications of the ACM*, 36 (7),

WORLD WIDE WEB CONSORTIUM (W3C) 1999. PIDL - Personalized Information Description Language. W3C Note, <http://avocado.w3.mag.keio.ac.jp/TR/NOTE-PIDL>.

WORLD WIDE WEB CONSORTIUM (W3C) 2001a. Composite Capabilities/Preference Profiles, <http://www.w3.org/Mobile>.

WORLD WIDE WEB CONSORTIUM (W3C) 2001b. Device Independence Principles, W3C Working Draft, <http://www.w3.org/TR/di-princ/>.

WORLD WIDE WEB CONSORTIUM (W3C) 2001c. Platform for Privacy Preferences (P3P) Project, <http://www.w3.org/P3P>.

APPENDIX

This appendix provides examples of customisation rules situated in the area of the conference management system.

Customise PC-Member Information. The following two customisation rules (`ReducePC_MemberInformation` and `IncreasePC_Member Information`) given in Figure 16 work together to provide two different versions of the PC-members' information, dependent on the device category used (i.e., "PDA" or "Desktop").

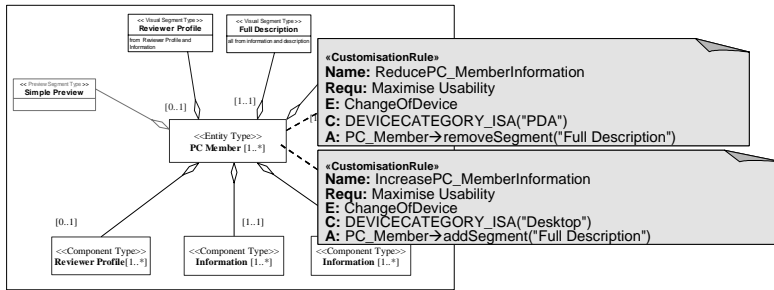


Fig. 16. Customisation Rules Reduce/IncreasePC_MemberInformation

This means that the "Full Description" should only be available for devices of type "Desktop" and not for "PDA". In both rules, the check for the device categories is performed each time the device context changes which is detected by the event ChangeOfDevice. The conditions use the macro DEVICECATEGORY_ISA () which tests, whether the current device is of the category given as parameter. The macro returns true if the current device is of the specified device category. The action in the customisation rule ReducePC_MemberInformation is responsible for removing the segment "Full Description" from the entity "PC_Member" by invoking the entity's generic adaptation operation removeSegment () given as parameter "Full Description". In contrary, the customisation rule IncreasePC_MemberInformation adds the segment "Full Description" by invoking the entity's generic adaptation operation addSegment () .

Customise PC-Member Navigation. The customisation rules in Figure 17 show the customisation which is necessary in Navigation Design, due to the customisation done by the previous example in Information Design.

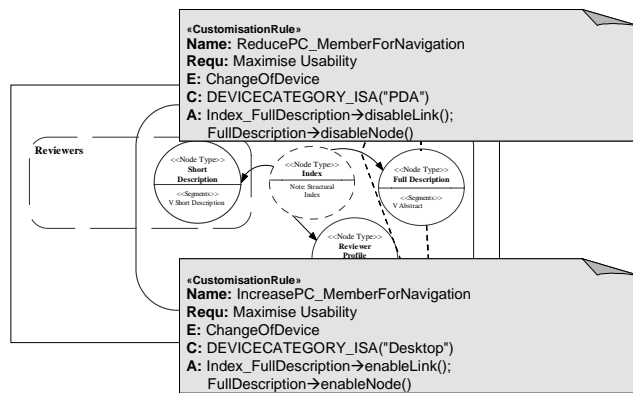


Fig. 17. Customisation Rules Reduce/IncreasePC_MemberForNavigation

In both rules, events and conditions are identical to the ones used in the previous example. Only the action differs in that generic adaptation operations are activated which are defined for model elements of Navigation Design. More concrete, the node "Full Description" and the link from the "Index" to the node "Full Description" are disabled if the used device is a "PDA" and enabled in case the device is "Desktop".

Filter Preferred Hotels. The customisation rule FilterPreferredHotels assumes that for a user only those hotels are relevant which confer to the current user's preferences (c.f. Figure 18).

```

«CustomisationRule»
Name: FilterPreferredHotels
Requ: Participant gets a list of hotels according to his/her preferences
E: ChangeOfApplicationState
C: CURRENT_COLLECTION_ISA("ConferenceHotels")
A: ConferenceHotels->setCollectionFilter(
    "Hotel.price <= " + CURRENT_USER.HotelPrefs.maxPrice +
    " AND Hotel.hotelCategory == " + CURRENT_USER.HotelPrefs.category)

```

Fig. 18. Customisation Rule FilterPreferredHotels

The customisation rule subscribes to the event `ChangeOfApplicationState`. The condition in turn checks whether the collection `ConferenceHotels` is invoked. The user's hotel preferences (`HotelPrefs`) in terms of the maximum price the user is willing to pay and the desired category of hotel are used to formulate a filter condition. This filter condition is set for the collection `ConferenceHotels` by invoking the corresponding generic adaptation operation `setCollectionFilter()`.

Filter Papers. The following customisation rule (cf. Figure 19) `FilterPapers` refers to the requirement that only the papers of the current track should be made available.

```

«CustomisationRule»
Name: FilterPapers
Requ: Only the papers currently presented in the current room's track should be available
E: ChangeOfLocation
C: CURRENT_LOGICAL_LOCATION_ISA("InLectureRoom")
A: Papers->setCollectionFilter(
    "Paper.Track.Location=" + CURRENT_LOGICAL_LOCATION_GET("InLectureRoom") +
    " AND Paper.Track.Time == " + CURRENT_TIME)

```

Fig. 19. Customisation Rule FilterPapers

The current track is identified by the current physical location. For this, the customisation rule incorporates the event `ChangeOfLocation`. The condition checks whether the current position is within a lecture room by referring to the logical location `LectureRoom` and invoking the macro `CURRENT_LOCATION_ISA` which returns true if this is the case. If so, the collection of all papers is filtered so that only those papers are available which are related to the current room and are currently scheduled for presentation.

Filter Staff. The customisation rule `FilterStaffForPresentation` reduces the list of all staff to the one currently assigned to the current lecture room (cf. Figure 20).

```

«CustomisationRule»
Name: FilterStaff
Requ: Only the staff currently assigned for the current lecture room should be available
E: ChangeOfLocation OR ChangeOfTime
C: CURRENT_LOGICAL_LOCATION_ISA("InLectureRoom")
A: Staff->setCollectionFilter(
    "Time=" + CURRENT_TIME +
    " AND Room="+CURRENT_LOGICAL_LOCATION_GET("InLectureRoom"))

```

Fig. 20. Customisation Rule FilterStaff

For this, the customisation rule is triggered by either a change of the current location or a change of time (cf. event `ChangeOfLocation OR ChangeOfTime`). The

condition of this rule checks whether the current location is within a lecture room. If this is the case, the filter condition of the collection `Staff` is set to contain any staff which is assigned to the current lecture room at the present time.

Upload Paper. The following customisation rule shown in Figure 21 realises, that the requirement that depending on the device's capabilities and the user's preferences, as soon as a user enters a lecture room, the papers of the current track are pushed onto the user's device.

```

«CustomisationRule»
Name: UploadPapers
Requ: On entering a room the papers of the current session are uploaded if the user decided to
do so AND the device is capable of doing this
E: ChangeOfLocation
C: CURRENT_LOGICAL_LOCATION_ISA("InLectureRoom") AND
CURRENT_DEVICE_HW_PROPERTIES.memory > "16 MB" AND
CURRENT_DEVICE_PROPERTIES.cpu > "8 MHz" AND
CURRENT_NETWORK_THROUGHPUT_LEVEL = "High" AND
CURRENT_USER.ConferenceInfo.downloadPapers = "TRUE"
A: Papers->setCollectionFilter(
    "Room=" + CURRENT_LOGICAL_LOCATION_GET("InLectureRoom") +
    " AND Time=" + CURRENT_TIME);
FileServer->downloadFiles("Papers", CURRENT_USER.phoneNumber)

```

Fig. 21. Customisation Rule UploadPapers

For this the customisation rule is triggered by the event `ChangeOfLocation`. The condition comprises three parts. First, it is checked whether the current location is actually within a lecture room. Second, it is checked whether the device currently used is capable of receiving the list of papers by testing the device properties memory, CPU, and the current network throughput-level. Finally, the condition comprises the checks whether the user prefers the automatic download of papers. The action contains two steps. First, the list of papers is reduced to the set of papers currently presented in the current lecture room (`setCollectionFilter()`). The second step invokes the component `FileServer` with the method `downloadFiles()` and the parameters to download the papers (now already the reduced set through the previously issued `setCollectionFilter()` and the current user's phone number.

Change Language of Entity. The customisation rule `ChooseLanguage` (cf. Figure 22) changes the language used for an object so that it is represented in the user's desired language each time the entity is accessed.

```

«CustomisationRule»
Name: ChangeLanguageOfEntity
Requ: When an entity is accessed change the language used for that object to the preferred one
E: ChangeOfApplicationState
C: CURRENT_ENTITY <> PREVIOUS_ENTITY AND
CURRENT_ENTITY.MetalInformation.enabledToChangeLanguage = "TRUE" AND
CURRENT_ENTITY.MetalInformation.currentLanguage <>
CURRENT_USER.PersonalPrefs.preferredLanguage
A: CURRENT_ENTITY->changeLanguage(CURRENT_USER.PersonalPrefs.preferredLanguage)

```

Fig. 22. Customisation Rule ChangeLanguageOfEntity

The customisation rule is activated through the event `ChangeOfApplicationState`. In the condition is checked whether the accessed entity is different from the previously accessed entity. Furthermore, the condition checks whether the entity is capable of changing its language to the one preferred by the user. If

so, the entity gets the adaptation operation `changeLanguage()` specifying as input parameter the current user's preferred language which is assumed to be present for any object able to change the language.