A CONTEXT AWARE USER PROFILE FOR PERSONALIZATION

D. Halvatzaras, M.H. Williams
School of Math. & Computer Sciences, Heriot-Watt University, Riccarton, Edinburgh EH14 4AS, UK

ABSTRACT
With the emergence of the first 3G mobile technologies in everyday life, mobile phones and PDAs are able to function as more than just simple means of communication. Research on these devices, in combination with 3G networks includes the development and use of a context aware personal assistant to support users. The Youngster project was one of the pioneers, which sought solutions for a more efficient interactive experience over mobile devices based on context aware and personalization technologies. This paper reports on some aspects of the way in which personalization and context awareness are handled in the Youngster platform to meet the overall goals. As a case study, two applications that make use of these functionalities are presented.

KEYWORDS
Context Aware, User Profile, Personalization, User Model

1. INTRODUCTION
Access to the Internet is widening from static PCs to mobile devices due to 3G mobile technology. As a consequence, the concept of Internet services is also affected and the idea of “allowing mobile users to access any information at any time from any location” (Billsus et al., 2002) has become an important goal. However, a “mobile” Internet user has different needs in terms of interaction and information from those of a static user and this has serious consequences for the design of services and interfaces. Services for mobile users have a greater need for personalized and context aware content delivery. Some research has been done on different scenarios of context aware interaction and even more on the delivery of personalized content. The “Youngster” project was a European research project funded under the Fifth Framework involving partners from Germany, France, Norway and the UK. It aimed to exploit new web and mobile technologies by combining them to create an open active mobile multimedia environment. Young people were chosen as a focus because of their interest in mobile technology and their urge to embrace new technologies (Williams et al., 2002). The outcome of this research was the Youngster Mobile Service Platform (MSP) which supports (and hosts) the implementation of context aware personalized applications and services with minimum effort. At the same time the implementation of the platform assures user privacy and profiling transparency, which are crucial factors for every context aware personalized service.

2. CONTEXT AWARE PERSONALIZATION
So far the majority of projects that deal with personalization and context awareness divide the user model into two parts: a) the user profile (which mainly supports personalization aspects of the service) and b) the context user model (which is used to support context aware aspects of the user interface).
In such projects it is common for the implemented architecture to offer personalization and context aware functionality in different layers (see figure 1). For example, in the GUIDE project (Cheverst et al., 2000) content is selected first according to the location of the user and then is filtered according to the interests of the user (it follows topology B as presented in figure 1). An example of topology A is the Conference Assistant (Dey et al., 1999), which selects content depending on the context of the user and his/her personalized interests.

Using different models and different layers gives the benefit of focusing the development of the two different areas independently. This facilitates the development of more powerful algorithms able to adapt the user interface to the user context and personal needs.

The biggest drawback, though, of using powerful adaptation techniques, is that they are based mainly on black box architectures, such as the neural networks architecture KBANN (Haddawy et al., 2002), or a hard-coded interface structure (e.g. Map Annotation Assistant (Patern, 2000)) leaving the user without any control over the process.

Intelligent context sensitive systems tend to adapt randomly to the changes of user's context. Even when a system attempts to optimize the behaviour of the user interface at the user end this is often perceived as something inconsistent. Users usually expect to find the same interface regardless of their context. A few research and case studies have already reached this conclusion. An evaluation study (Cheverst and Nigel, 2002) of the Guide project revealed that the trial user group was not satisfied with the way that the user context was influencing the delivery of content. Also Reiter et al. (2003) conclude that users must be given control over their user model if possible.

One might argue that user controlled personalized interfaces are more desirable since they are more consistent although Barkhuus and Dey (2003) amongst others argue that users would also prefer to avoid the boredom of configuring a fully personalized interface.

It is hard to draw a line separating the personalized preferences of a user from his/her context. In the physical world human beings are associated with their context so tightly that even their personality seems to be influenced by that. For instance, if somebody with a sporty disposition lives on a mountain that is covered in snow in the winter, (s)he is probably more inclined to favour winter sports. However, if in the future (s)he
moves to the sea the attention of the individual will be drawn to water sports. Also the context has a different feeling for each user depending on the skills and the experience of the individual. For example, somebody who reads a lot on a train may not need a reading aid (e.g. larger fonts offered as an aid by Adaptive GSM phone and PDA (Schmidt et al., 1999)) although an inexperienced traveller might do so. In conclusion, separating the context from the user model results in “losing the true sight of the user” (Jameson, 2001).

Since the distinction between personalization and context awareness is not always clear (resulting sometimes in undesired side effects at the user end), the Youngster project explored the merger of these two concepts to produce “context aware personalization”. Based on the study of several personalization and context aware projects a definition can be suggested:

Context aware personalization aims to achieve automatic adjustment of the system behaviour as the user context changes, to fit the preferences and interests that the user has specified for different contextual conditions.

A topology representing a context aware personalized system is illustrated in figure 2. This topology, in contrast to the traditional ones, interprets context according to the requirements of the individual. This topology suggests the need for a context aware user profile. The extended characteristics of this type of profile are based on the ability to sustain different profile values for different user context states.

Based on the conclusions presented in the study of Barkhuus and Dey (2003), this hybrid approach results in a more human centric interaction, reducing frustration since the user is able to control the behaviour of the system under different contexts. A personalized context aware system makes the user more confident since there are no unexpected behavioural effects. At the same time the user feels more secure because he/she can control the way that contextual information is interpreted by the system.

3. ARCHITECTURE OF YOUNGSTER PERSONALIZATION MODULE

The architecture of the Personalization Module used by the Youngster MSP is presented in figure 3. The architecture is deployed in 3 layers:

A. The support layer (right light grey area in figure 3). The main component of this layer is the Profile repository, which is responsible for the administration of the user profile. The Ontology Constructor is the component that is able to read and implement a profile that follows a particular ontology. The description of the user profile ontology can be defined by means of an XML document (following the

Figure 3 Architecture of the Personalization Module
A CONTEXT AWARE USER PROFILE FOR PERSONALIZATION

Youngster Personalization DTD). These components are not accessible from external modules for security reasons. Another component that belongs in this area is the Interface Instructor. This component offers necessary functionality to applications that need to implement a profiling user interface.

B. The service layer (left dark grey area in figure 3). These components are accessible from other services or applications in the Platform. They offer a safe gateway to the user profile and also offer personalization as a group of functionalities to other applications or services. The Personalization Manager can parse and personalize any XML document (provided it contains some dedicated annotations). The Personal Assistant (P.A.) is responsible for providing at the user end any necessary interfaces for the administration of his/her profile. The user interface is constructed based on the interpretation that the Interface Instructor provides from the XML document that specifies the user profile.

C. The external (or virtual) layer (bottom white area in figure 3). Special services (or applications) might need to infer higher context through dedicated algorithms. The personalization module can provide “trusted” applications with access to the core components, allowing them to perform dedicated personalization techniques.

4. CONTEXT SENSITIVE PROFILE ATTRIBUTES

The initial user model used in the Youngster MSP was static and could only give answers to the question:

- What is the value of personalization-attribute X for user U?

However, this approach did not prove to be very efficient. As stated in section 2, users will always be conservative in the amount of effort that they are willing to put into configuring applications. This also applies to the configuration process of the user profile. The findings of the field trials (Halvatzaras and Williams, 2004) confirmed this type of user behaviour among youngsters. What must be noted, however, is that this behaviour is mainly due to the fact that an average user prefers to avoid tedious interactions and not because (s)he disregards the benefits of personalisation. This study showed that 70% of the users did try to personalise at least some aspects of the platform although most users configured personalisation without getting deeply involved.

Figure 4.(Left top) ORD model of a context sensitive link.
Figure 5.(Right)3D representation of context sensitive attributes

The initial user model used in the Youngster MSP was static and could only give answers to the question:

- What is the value of personalization-attribute X for user U?

However, this approach did not prove to be very efficient. As stated in section 2, users will always be conservative in the amount of effort that they are willing to put into configuring applications. This also applies to the configuration process of the user profile. The findings of the field trials (Halvatzaras and Williams, 2004) confirmed this type of user behaviour among youngsters. What must be noted, however, is that this behaviour is mainly due to the fact that an average user prefers to avoid tedious interactions and not because (s)he disregards the benefits of personalisation. This study showed that 70% of the users did try to personalise at least some aspects of the platform although most users configured personalisation without getting deeply involved.
The basic problem with personalisation services is that they are affected by the context of the user. When the user context changes, his/her preferences might change, and as a consequence the user is forced to go through trivial interactions. A good example is the cinema scenario which demands the user to set the mobile device in silent mode when (s)he enters a cinema. It is quite unlikely that the user will browse through a sequence of menus in order to set and reset this type of preference every time he/she goes in and out of a cinema or theatre. It is more likely that (s)he will simply switch off the device.

A better approach to improve the benefits of personalisation, even for the less interactive users, is to acquire as much user profiling information as possible the first time the user updates his/her profile. This, however, must be achieved without requiring the user to complete lengthy interaction forms. An alternative approach to support this, followed later on in the Youngster project. This was the introduction of a context aware user model. Thus the user model must also be able to give an answer to the question:

What is the value of personalization-attribute X for user U if user's context state is CS?

where CS is a set of boolean expressions that formulate the set of context conditions which determine a context state, for example:

CS(location_type)=(location_type="theatre") OR (location_type="cinema")
CS(time)=(time="night")

When this set of expressions is true it means that the context state of the user is “night at the cinema or theatre”.

So far several projects (such as MyYahoo.com (Manber et al., 2000), news service of Los Angeles Times (Billlsu et al., 2002), PAL (Silber, 1990), Shopping Assistant (Asthana et al., 1994), ComMotion (Marmasse and Schmandt, 2000), CybreMinder [(Dey and Abowd, 2000) and others) have investigated personalization and context awareness techniques. Until now, though, there are no user modelling standardizations and no project (to our knowledge) attempting to combine context awareness and profiling information under a common model.

In order to bridge context awareness and personalization under a single model, context sensitive user profile attributes were introduced. To simplify the study of context sensitive objects, the modelling component Context Gate is suggested as a means to model context sensitive behaviour. The Object Relational Model (ORM) (Blaha & Premerlani, 1998) diagram that conceptualizes a context sensitive relation between two objects is presented in figure 4.

A context gate has a set of possible context states, which can be defined by the user. Every context state has one or more context conditions. A context condition is a Boolean equation that has as an input parameter Context Attribute (as defined by Dey et al (2001)). The Context Gate is a type of context sensitive node that relates one object to a set of different versions of another. The different versions apply to different context states (as an example see figure 5). Thus, in figure 4 Object 1 is not linked constantly to the same version of Object 2. Every time that it is instantiated, it will be linked to the version of Object 2 that is valid for the current user context state. A context gate, at a programming level, must implement an algorithm that will validate the possible context states in order to determine which is active.

In the case of the Youngster User Profile a context gate relates different values to the same set of attributes, depending on the current context state of the user (see figure 5). For example, the “news” group attribute (which supports profiling information relevant to the news application) owns the attribute “notification_status”. The user might desire this attribute to be set to “silence_mode” when (s)he is in a “cinema at night” and to “sound_mode” at all other times. This dynamic behaviour is achieved by assigning two different context states to the same attribute set (which is dedicated to the news profile).

Each context state is characterized by a priority parameter. The latter parameter is used in the case that one or more context stages overlap. For instance, a user might set the notification_status of the news application to “silence_mode” every morning but (s)he has also set the notification_status to “sound_mode” every Sunday all day. These two context states overlap. By assigning the first context state a lower priority than the second one, a virtual subtraction is performed which results in the user achieving the silencing of the notifications every morning apart from Sunday mornings.

A context sensitive user profile by nature has a dynamic behaviour (since one attribute does not have a static value). By introducing the context gate, as a component, it allows the design of such dynamic models with the same ease as designing a static model.
Traditionally, user interfaces are modelled in isolation from data models although an interface dedicated to profiling administration must use the data model as a guideline. In the Youngster MSP the user model has been designed to support both (data and interface) at the same time. This was achieved by keeping the structure of the user model to a level that is similar to the structure of a user interface model. This principle made it possible to map every component in the user model to a component in the user interface model. For instance, the component ‘group of attributes’, in the user model, can be mapped to an interaction screen (in the user interface model), which will administer the values of the ‘group of attributes’.

In the Youngster MSP the user can administer his/her profile through the Personal Assistant. The latter is able to interpret the description of the user model (provided in an XML format) into appropriate user interaction screens (in xHTML format). Figure 6 illustrates a user interface as implemented by the P.A. One of the applications offered by the Youngster Platform was the News application with personalization and context aware features. Figure 6 gives an example of interaction screens that the P.A. has created (on behalf of the News application) for the administration of the news user profile. In this case the user can choose a context state from a list (combobox in Figure 6) and perform the following actions:

a) To personalize the news application by setting the desired startup news screen, the notification status, etc.

b) To assign a selected context state to the current personalization settings.

c) To modify the personalization settings which are assigned to another context state.

d) To edit the context conditions which define the selected context state.

The following XML structure (which is based on the youngster-personalization-XML schema) is an example of the user profile ontology which is relevant to the news application:

```xml
<group name="news-app" button="News" title="What is your preferred...">
  <local name="prefPage" pa-question="First page?"
    value-set=";0{Last 2 days};1{Last 7 days};2{MyNews}" default-value="2"/>
  <local name="notification"  pa-question="Do you want to be notified?"
    value-set=";0{NO};1{YES}"  default-value="1"/>
  <associate concept-group="interests"/>
</group>
```

Figure 6. P.A. dialogue for setting context sensitive profiling information regarding the News application
The previous XML structure is used in parallel a) by the Profile Repository to build the structure of the user profile and b) by the P.A. to create the appropriate interaction screens (presented in Figure 6). This approach does not give application or content developers any options to hide any profiling attributes from the user. Thus full profiling transparency is achieved which is a crucial factor for every personalization system (Sinha R and Swearingen K., 2002).

6. CONTEXT AWARE NOTIFICATION SYSTEM

One of the aspects that the Youngster MSP offers to the user is the notification of events that may occur in the platform - for example, the delivery of a new message or a news article of interest. Any application (embedded in the Youngster MSP) that has the need to send a notification to the user, can achieve this through the notification service. In brief the life circle of a notification is as follows:

1. An application creates a notification through the Notification Service. 2. The notification service checks the user profile to verify if the user allows the delivery of a notification with certain specifications in the current context. 3. If the user allows it then the notification is sent. In the cases where this is not allowed, the notification is cached and step 2 is repeated at regular intervals.

In the basic version of the Youngster platform the notification service was a personalized service where users could personalize:

a) The acceptance of a notification depending on the sender related to that notification. For instance, the user could specify that a notification signalling the arrival of a new message from a sender from the black-list group must be rejected.

b) The actions of the notification client in case of the arrival of a notification. The user could set the visual and sound effects of this action.

The user could explicitly activate or edit, at any time, one of the preconfigured profiles.

In the field trials the notification profiles were not used as frequently as expected (Halvatzaras and Williams, 2004). In order to improve their efficiency, a new user model was specified including context aware capabilities. Based on the new specifications, the new notification user profile was designed as presented in figure 7.

The new user profile releases the user from the boredom of creating different notification profiles which (s)he will have, later on, to remember to activate manually. Under this new model, (s)he can create a single profile with many different context states. For example, a user can set to accept notifications from the messaging application when (s)he is in a restaurant but not from the news application (since notifications coming from the latter application may not be that important).

The user interface, that allows control of the context aware notifications, is fully implemented by the Personal Assistant. Figure 8 presents the interaction screens (created by the P.A.) that support the administration of a context aware notification profile. More precisely, figure 8 shows a sequence of interaction screen-shots that result in the following user-profiling configuration:

\[
\text{if I am in the school}
\]
\[
\text{if a notification related to Bob arrives then play the '007_music' melody.}
\]
Or

if a notification related to News arrives then play the 'koukou' melody.

No changes in the implementation of the Notification Service were required to implement a context aware behaviour at the Notifier component. As presented in the previous paragraph (step 3) the notification service caches any notification that does not comply with the user preferences for later verification. Every time a cached notification is re-evaluated the user profile is accessed again (step 2). Since the profile repository is context sensitive, every time an application requests an attribute, it delivers back the values assigned for the current user context. As a result the notification service inherits a context aware behaviour.

7. CONCLUSIONS

This paper presented an alternative framework that perceives context awareness as an aspect of the user profile (instead of dealing with them as two isolated fields).

The Context Gate was introduced as a means of modelling context sensitive components in a user model. This component does not aim to cover complex dynamic user models. It was introduced mainly as an attempt to create a static equivalent of a context sensitive attribute. This is important for two reasons:

A. To reduce the effort of dealing with context sensitive user models. Context sensitive applications can be modelled with the same ease as non context sensitive ones.

B. To improve the transparency of the user profile. As mentioned before dynamic user models cannot guarantee transparency since they follow black box architectures. As long as a context sensitive profile can be mapped to a structure similar to a static user model transparency becomes feasible.

The personalization module of the Youngster MSP succeeded in providing different context aware and personalization functionalities to the hosted applications under a single architecture without jeopardizing profiling transparency and privacy. Under the Youngster project three context aware personalised applications were developed, context messaging, context aware news (presented in section 4), and a context aware music player, which used the functionalities offered by the personalization module. Also two context aware personalised services were developed, the notification service (presented in section 5) and the context aware content server. The Youngster user model proved able to satisfy the needs of these applications and services. It was even able to support applications requiring higher context such as an Intelligent Music Player, which created play-lists based on the user context.
There is scope for further research on context sensitive profiles. Many issues were not investigated in this project. One of the next steps must be the study of synchronization issues between an application and a context sensitive user profile. An optimal way has not yet been found for an application to pull a context sensitive value. Performance issues, as well, are crucial in such projects and bottlenecks can easily arise in a profile repository that is constantly accessed by many applications.

ACKNOWLEDGMENT

This work was funded by the Fifth Framework Programme of the European Union through the Youngster project. The authors acknowledge this support.

REFERENCES


