Personalizing Web-Based Information Systems through Context-Aware User Profiles

Manuele Kirsch Pinheiro[+,‡], Marlène Villanova-Oliver[*], Jérôme Gensel[*], Yolande Berbers[+], Hervé Martin[*]

[+]*DistriNet / Department of Computer Science, Katholieke Universiteit Leuven*

[‡]*Centre de Recherche en Informatique, Université Paris 1 – Panthéon-Sorbonne*

{ Manuele.KirschPinheiro, Yolande.Berbers }@cs.kuleuven.be

[*]*Grenoble Computer Science Laboratory*

{ Marlene.Villanova-Oliver, Jerome.Gensel, Herve.Martin }@imag.fr

Abstract

In this paper, we propose context-aware profiles and a filtering process for personalising informational content that is delivered to mobile users by Web-based information systems (WIS). The context-aware profiles allow mobile users to express their personal preferences for particular situations they encounter when using a WIS. We argue that the preferences and the needs of a mobile user may vary according to the context in which he uses the system. By defining these profiles, we propose a filtering process that takes into account both the user's current context and the user's preferences for this context. This process selects, in a first step, the context-aware profiles that match the user's current context, and then it filters the available informational content based on the selected profiles.

1. Introduction

Mobile technologies, such as WiFi networks, PDAs and smartphones, grant to mobile users a ubiquitous access to Web-based information systems (WIS). This ubiquitous use encourages the development of context-aware WIS, where context-awareness [2][7] refers to the capability of perceiving the user situation and of adapting in consequence the system behavior (*i.e.*, the services, the data and the interface). Examples of this trend are systems informing user about nearby restaurants, parking places, etc., or systems allowing users to share contextualized information (*e.g.* location-aware annotations or photos). The later illustrates another trend on WIS, mainly related to the Web2.0 concepts [1]: the development of social networks and communities, through which mobile users share information and collaborate with other users.

When considering such ubiquitous access, the user's interests and preferences may differ according to the context [8]: the same information may be interesting for a given user when he is on the office and completely useless when he is traveling. We believe that context-aware WIS should personalize supplied content based on both the *user's current context*, including his location and device as well as user's activities and collaborations, together with his *preferences for this context*.

In this paper, we propose context-aware profiles which allow mobile users to indicate context-aware preferences considering informational content supplied by WIS. Thanks to these profiles, mobile users are able express their preferences for situations they often encounter when accessing WIS (accessing from the airport, from home, using a smartphone, a laptop, when handling an object...). Through these context-aware profiles, it is possible for WIS to select available informational content according to both the user's current context and his personal preferences for this context. We propose a filtering process that performs such a selection based on the proposed profiles and on an object-oriented description of the user's context.

In the following sections, we present related works. Then we introduce the object-oriented context model we propose to represent the user's context (Section 3). After that, we propose our context-aware profile model (Section 4). In Section 5, we propose a context-aware filtering process that filters available content based on the user's context and on the user's pre-defined profiles. In Section 5, we discuss some implementation issues, before concluding in Section 6.

2. Related Work

Context-awareness [2] refers to systems able to adapt their services [15] or the supplied content [16] to the user's context. For instance, works such as [10][14] propose to adapt the content delivered to mobile users according to the user's location, whereas works such as

978-0-7695-3367-4/08 $25.00 © 2008 IEEE
DOI 10.1109/UBICOMM.2008.18

[16] propose to adapt this content to the client device capabilities, and works such as [15] propose to adapt the system composition and services according to the available resources in the client devices.

All these works rely on the notion of context. The concept of user's context refers to a very large notion [2][3], which can be seen as *any information that can be used to characterize the situation of an entity* (a person, place, or object considered as relevant to the interaction between a user and an application) [7]. Several models for representing context exist in the literature [2]. In their majority, these models cover concepts related to the user's location and mobile devices. Even if some propositions, such as [13], include the user as part of the context model, most of these models consider the user as an isolated individual. However, when considering WIS dedicated to communities, the social aspects related to the community and collaborative process that takes place on it should be considered, since the goals and the activities of a group influence the actions of each individual in the community. The interaction a user has with other users should be considered by the context models [9]. Thus, WIS supporting community of users (including wiki or photo sharing applications) need a context model that takes into account the social aspects related to the collaborative activities.

Moreover, several works such as [15] [14] reserve a small place for user's preferences, or do not relate these preferences to the user's context when considering the adaptation process. However, user's actions and expectations considering context-aware systems directly depend on context in which the user is interacting with the system [8]. Traditional approaches for content adaptation on WIS are usually are guided by profiles expressing the user's preferences [4] [6]. However, often these works do not consider user's context or relate these profiles to particular situations. Mobile users' interests and preferences may vary according to the context in which these users access a WIS. A system that did not take into account the user's preferences or that considers only fixed preferences may not correspond to the user's expectations.

3. An Object-Oriented Context Model

We propose in this paper to personalize content supplied to mobile users by using context-aware profiles and a filtering process. These profiles represent the user's preferences considering a particular situation, which is described using an object-oriented context model we have previously defined [15], which focuses on a mobile and collaborative use of WIS. Moreover, we choose an UML approach for its easiness of use and for its appropriateness to the MDA software development approach.

Based mainly on a set of UML diagrams, the proposed model represents both the user's *physical context* (including concepts referring to location, device and application) and the user's *collaborative context* (including concepts as group, role, member, activity and shared object). We claim that collaborative context should be considered since we are considering mobile users participating of one or more communities. Figure 1 shows all the concepts forming the model.

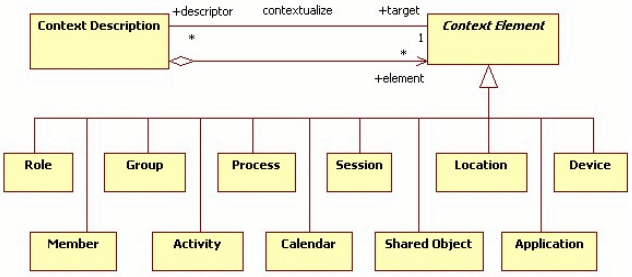

Figure 1. A context description is seen as a composition of context elements.

In this model (see Figure 1), the concept of context is represented by a class Context Description, which is a composition of both physical (Location, Device, Space and Application) and collaborative elements (Group, Role, Activity, Shared Object, etc.). These elements are represented by classes that are specializations of a common superclass, called Context Element. Furthermore, these context elements are related to each other, defining associations between the corresponding concepts. Each element of context belong thus to a complex representation of the user's situation. For instance, we consider that a user is the member of a group through the roles he plays in this group, and that a user handles a shared object through some application that allows it, and so on. A complete description of these associations can be found in [15].

The context of a user is represented in this model by an object of the class Context Description, which is linked by composition to objects of the class Context Element and its subclasses (see Figure 1). Figure 2 illustrates an application of this context model. In this figure, we consider a user (*Alice*), who is the coordinator (*coordinator* role) of a team (*administration* group), and who uses a wiki application. Let us suppose that Alice is accessing this system through her PDA in order to consult the latest changes on the document that her team and she are working on. When Alice requests these changes, her current context can be represented by the context description object represented in Figure 2. This object is composed by the context elements representing Alice's location (*office D322* object of the Location class) and device (*PocketPC* object of the Device class), her team (*administration* object), her role

in this team (*coordinator* object), and so on. These objects are related through a set of associations: Alice belongs to the administration group through the role coordinator; this role allows Alice to perform the *report edition* activity, which handles the shared object *report2007* through the *wiki* application, etc. All these associations, together with the context elements they connect compose the current Alice's context.

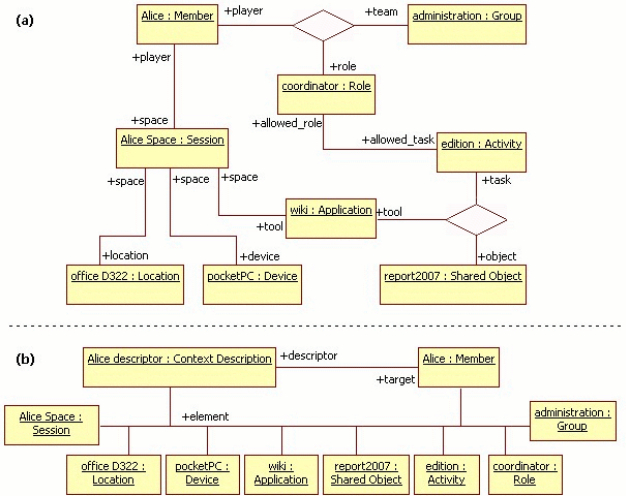

Figure 2. Example of a context description for a given user (Alice).

4. Context-Aware Profile

User's preferences are represented through the notion of profiles. We propose here a context-aware profile model. The defined profiles are associated to *potential contexts* that may characterize a user's situation (called an *application context*). They also define implicitly *filtering rules* that should apply when the user's current context matches the application context (expressed in the context model above). In a few words, a profile describes which information the user wants to be informed of when he is in this situation. In the following sections we detail the definition of these profiles.

4.1. Informational content: representing group awareness information

Context-aware profiles aim at indicating which content is relevant for a user when he is in a given situation. To reach this goal, we modeled in a content model the informational content that is filtered and delivered to the user by WIS. This model is an abstraction for the available information, represented through the class content (Figure 2). This class represents any piece of useful information for the users and for their communities: a list of restaurants, a set of shared notes, on-line users, etc. Since we are particularly interested on communities of users, we consider here, for illustration purposes, to supply mobile users with information particularly related to the community (colleagues' actions in the community, users' presence in a given location, etc). This information refers to the notion of *group awareness*, which allows user to better coordinate their own activities in a collaborative process [3]. We refer to this information as an *event* content subclass.

Besides the content it represents, we consider that an *event* may refer to one or more elements of the context model (see concerns association in Figure 3), since it may carry some information about a topic referring to these elements (for instance, the event referring to the Alice's presence is linked to the object of the Member class referring Alice). We also consider that each event object is associated with a context description object (see occurs association in Figure 3), representing the context in which the event is produced (for instance, the context description object describing Alice's current context in Figure 2).

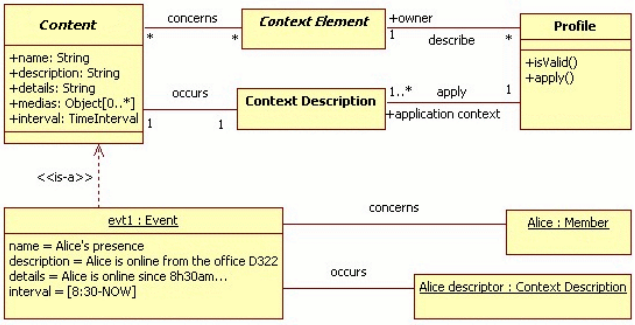

Figure 3. UML class diagram describing the content class and an instance of the event subclass.

4.2. Profile composition and application context

The basic idea behind the context-aware profiles is to allow users to define the profiles and the potential context in which these profiles should be used. A potential context is called here an *application context*, and it is described using the context model. A user may define profiles for the most common situations he uses the system, describing for these the corresponding application contexts. For instance, considering a WIS with wiki services, its designer may pre-define some content subclasses for notification purposes such as a *document changed* class, whose objects describe the changes performed in a document, or a *new comment* class, whose objects include the comments made by the group members about a document. In such case, the user *Alice* may define a profile signing up only the *comment* content and associate it with the situations in

which she uses her PDA (*i.e.* a profile that is valid only when she is using this device). The same user may define another profile for the situations in which she is playing the coordinator role of the community, selecting *document changed* content instances.

Thus, each context-aware profile P is seen as a set composed by:

- an owner (Ow), for whom/what the profile is defined. The owner is represented by a context element object (association describe in Figure 3), allowing the definition of a profile either for users or for any element of the context model;
- at least one application context (Cp) to be considered, which represent the situations in which the profile is valid (association apply in Figure 3) and should be selected by the filtering mechanism (cf. Section 5);
- a list of signed up content subclasses (Ev) whose instances can be selected (association sign up in Figure 4), representing the informational content (the events in our case) considered as relevant for the owner;
- a set of contextual conditions (Cc) that filters events instances of the signed up classes (see Figure 4);

P = { Ow, Cp, Ev, Cc }

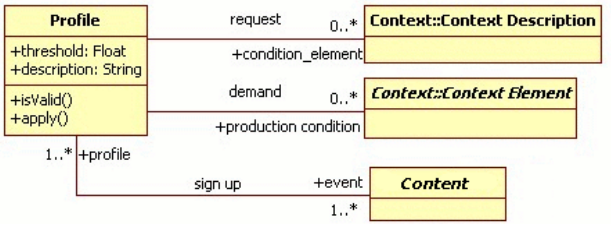

Figure 4. The signed up content and the contextual conditions that define the filtering rules of a profile.

The contextual conditions (cf. Section 4.3) and the list of signed up content (event) subclasses compose the filtering rules of the profile. These implicit rules guide the filtering process, allowing the selection of information that belongs only to the classes explicitly indicated as relevant (signed up classes) and that respects the contextual conditions imposed by the profile definition.

4.3. Contextual conditions

In order to present the contextual conditions, let us consider first an illustration of these conditions. Let us consider that Alice has defined a profile in which she is interested only in events from *new comment* class that have been produced in a given location (the meeting room) and that concern a given document. The first argument (events from new comment class) corresponds to the signed up content, the second one (events in a given location) is a request condition, whereas the third one (events concerning a document) correspond to a demand condition.

The contextual conditions defined in this profile model are represented in Figure 4 by the request and the demand associations. These conditions analyze the context in which content is produced, as well as the context elements concerned by it. The definition of a request condition in a profile indicates that only content directly related to a given context should be selected, whereas the definition of a demand condition means that only content that concerns specific context elements should be selected. These contextual conditions exploit occurs and concerns associations proposed in the content model (see Figure 3) in order to select content based on this information. Thus, when applying a profile with such conditions, these conditions are combined to the signed up indication, recommending that only signed up content whose occurs and concerns related objects match respectively the request and the demand conditions should be selected. This matching is performed by the filtering process, presented in the Section 5.

5. Context-Aware Filtering

Based on the context-aware profile presented above, we propose a filtering process in *two steps*. The first step selects the profiles WIS should apply in order to filter the available content according to the user's current context. The second step consists in applying the filtering rules defined by the selected profiles. These rules are based on the *signed up* content and the contextual conditions associated with the profile. We assume that, for each user, a set of pre-defined profiles is available.

5.1. First step: selecting profiles

The first step of the proposed filtering process consists in selecting among the available profiles those that are valid with regard to the user's current context. This selection is performed by comparing the application context related to the available profiles with the user's current context. For each profile, we test if one of its application contexts is a subset of the user's current context description. In other words, we test whether the situation described by the application profile occurs in the user's current context. If it is the case, then the profile is selected to be applied.

In order to identify this subset relationship, we consider that each context description object and the context element objects associated with it define a graph, where the nodes represent the objects and the edges between them represent the associations involving

these objects. Thus, a context C is a sub-context of a context C' whenever the graph corresponding to C is a subgraph of the graph corresponding to C'. The subgraph relationship is established using a pattern matching algorithm, which is based on two operations (equals and contains), defined as follow:

- *Equals*: (i) a node N is considered as equal to a node N' if the object O represented by N belongs to the same class (or a subclass) and defines the same values for the same variables that the object O' represented by N'. (ii) an edge E is equal to an edge E' if the associations they represent belong to the same type and connect equal objects.
- *Contains*: a graph C contains a graph C' if: (i) for each node N' that belongs to C' (called $N'_{C'}$), there is a node N belonging to C (N_C) for which $N'_{C'}$ *equals* N_C; (ii) for each edge E' in C' (called $E'_{C'}$), there is an edge E in C (E_C) for which $E'_{C'}$ *equals* E_C.

Thus, we consider that a context description C' is a subset of a context description C if the graph defined by C contains the graph defined by C'. For instance, considering the user Alice that is consulting the notes about a document, as in Figure 2, let us suppose that Alice has defined two profiles: (i) the first one has an application context referring to her PDA (*i.e.* a profile for the situations in which she is using this device); (ii) the second is applicable only when she is working with a given document on her desktop. This means that the application context related to this profile includes the context elements corresponding to the shared object *report2007* and to the device *desktop*. When Alice finds herself in the situation described by Figure 2, only the first profile is selected. The second one is rejected since the context description object representing the application context of this profile does not match the user's context description (the later does not include a node referring to the desktop device that is present in the former). Figure 5 shows the graphs defined by context description objects related to Alice's current context (Figure 5a) and her profiles (Figure 5b and Figure 5c respectively). In this figure, we can observe that the graph defined by the application context of the second profile is not a subgraph of the graph defined by the Alice's context, while the graph defined by the application context of the first profile is a subgraph of the one defined by Alice's context.

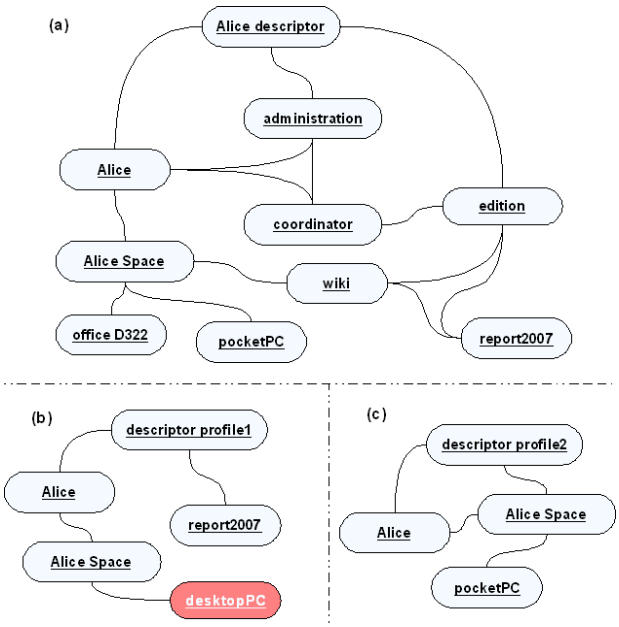

Figure 5. The graphs defined by a user's current context (a), and the application context of two profiles (b and c).

5.2. Second step: applying filtering rules

Once the profiles have been selected, the second step of the filtering process applies the filtering rules defined in the selected profiles. This implies to filter the available content according to the contextual conditions (see Section 4.3) and the content signed up by the profile (see Section 4.2).

Since many profiles can be selected in the first step, the second step of the filtering process orders the profiles by priority. The priority of a profile is calculated by a similarity measure between the application context of the profile and the user's current context. This measure evaluates the matching degree between the application context and the user's context. It estimates the proportion of elements of the graph defined by the later that have equal elements in the graph defined by the former. Therefore, more specific profiles (*i.e.* profiles whose application context is composed by several context elements) will have a higher priority than more general profiles, which have fewer context elements in their application context. This similarity measure is defined as:

- *Sim* $(C_u, C_p) = x$, $x \in [0,1]$, where: x=1 if each element of C_u has an equal element in Cp; otherwise $x = |X| / |C_u|$ with $X=\{x \mid x \textit{ equals } y, x \in C_u, y \in C_p\}$

Once the profiles have been ordered by the *Sim* measure, the filtering process applies, following the order, the filtering rules defined by each profile. For each one, it selects the objects belonging to the signed up content subclasses, organizing them in a list of

selected content objects. As an illustration, let us consider a WIS with a wiki service. The designer of such a system may have defined three content classes: *new comment*, *document changed* and *user presence* (indicating on-line users). Let us consider now that Alice has two selected profiles, whose application context objects are represented in Figure 6. The first profile signs up the first content class, whereas the second profile signs up the third content class. Considering that application context associated with the second profile refers to Alice using her *pocketPC* to access the *wiki* (see Figure 6b), when comparing these two profiles with Alice's current context (Figure 2), the second one has the priority over the first one, since $Sim(C_{Alice}, C_{P3})=0,42$ for the second one and $Sim(C_{Alice}, C_{P1})=0,38$ for the first one. This means that the second profile is applied before the first one. As a consequence, objects of the *user presence* class are selected and will be available for Alice before objects of *new comment* class, and objects of other content classes are not selected.

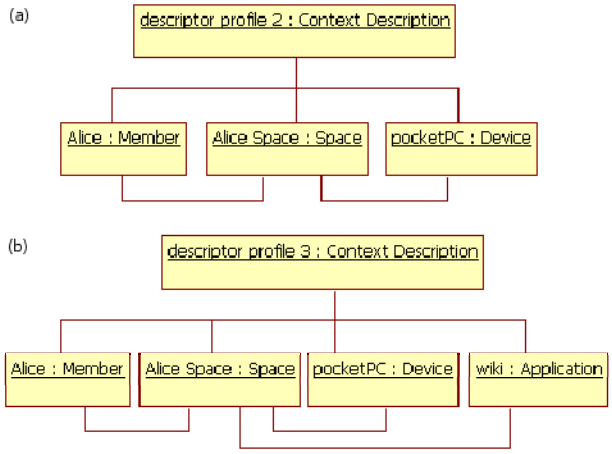

Figure 6. The application context objects related to two selected profiles.

After this first selection, the filtering process evaluates the contextual conditions associated with the profile (cf. Section 4.3), removing from the list content objects that do not match these conditions. In order to evaluate the *request* condition, the filtering process uses the *contains* operation: if the context description associated with the content object does not contain the context description give by the request condition, then the object is removed from the selection. For evaluating the *demand* condition, the filtering process uses the *equal* operation, comparing the context elements associated with each selected content object and those indicated in the demand condition related to the profile. If for one object in this condition, there is no matching object related to the content object, this last one is removed from the list of selected content.

The contextual conditions reduce the list of selected content objects, allowing only those whose related context matches the conditions. It is worth noting that if a selected content object does not possess any context description or context element associated to it, the contextual conditions do not apply, since there is not enough knowledge about context to compare it with these conditions. In this case, object remains selected.

Once the processing of a profile finished, the filtering process starts processing the next profile, according to the priority order defined above. For the next profile, it applies the filtering conditions and includes the selected content objects in the same list of selected content. It is worth noting that if a content object that is already in the list is selected by another profile, it is not added twice in the list. This occurs when two selected profiles sign up the same content. In this case, the priority among the profiles is used, and only the first instance is kept in the list.

At the end of this step, a list of selected content objects, ordered by the selected profiles, is available and can be delivered to the user by the WIS.

6. Implementation Issues

The proposed models and filtering process can be implemented by context and adaptation managers inside WIS. Indeed, authors such as [15] have been advocating for separating application logic from adaptation mechanism and context management. We adopt the same position by proposing, for testing purposes, a framework called BW-M. This framework is implemented as a Web Service, using the Java and an object-based knowledge representation system called AROM [11]. We choose the AROM system since it proposes classes/objects and associations/tuples as main representation entities, allowing a quick translation of the proposed models into a knowledge base. Next subsections introduce the BW-$\mathcal{M}$ framework and discuss some testing results.

8.1. The Structure of the BW-M Framework

The main structure of the BW-M framework is shown in Figure 7. The underlying assumption we made is that BW-$\mathcal{M}$ does not handle directly user's interface neither the context detection. We suppose WIS that have dedicate components for acquiring context information from physical or logical sensors, such as in [7][10][4]. These components identify context elements such as user's location and current activities, and inform BW-$\mathcal{M}$ framework about this information. It is worth noting that when these

components cannot determine all context elements (for instance, a GPS unable to determine mobile user's location in a building), BW-$\mathcal{M}$ framework (and the AROM knowledge base) handles these situations by omitting unknown elements and attributes (that remain unvalued in this case) from the user's context description. This omission means that the system does not have enough knowledge to represent an element (or parts of it).

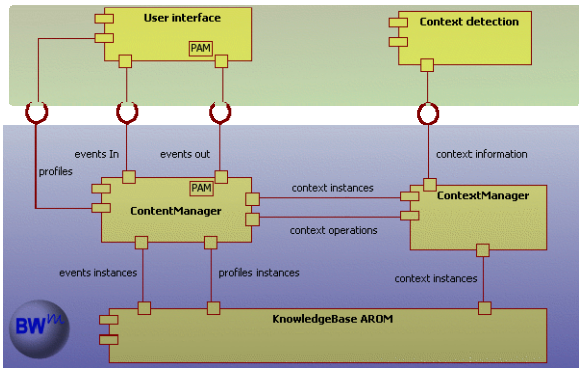

Figure 7. BW-M Framework architecture.

The BW-$\mathcal{M}$ framework is composed by three main components (see Figure 7): the *content manager*, the *context manager* and the *AROM knowledge base*, which keeps the instances of the profile, content and context models. The *content manager* component handles and filters the available content objects. In our experiences, we have used a set of pre-defined events for this purpose. The content manager also receives the filtering requests from the WIS. These requests trigger the filtering mechanism described in Section 5.

The *context manager* handles the context information it receives from the context detection components and introduces this information on the AROM knowledge base. It implements also the *equals*, *contains* and *Sim* operations, which are used by the content manager during the filtering process (cf. Section 5).

8.1. Discussions

Using the BW-$\mathcal{M}$ framework, we have performed a set of tests, simulating situations against a set of pre-defined profiles and content events. These tests raised interesting points considering *equals* and *contains* operations, which are key aspects of the proposed filtering process. Tests showed that the operations as defined in this paper are too restrictive. For instance, *equals* operation, as defined here, compares all attributes that have a known value in two objects. Consequently, all attributes are equally important when comparing two objects, and if one attribute has different values in the objects, these will be considered as distinct. When applied inside the *contains* operation, this may lead to the non selection of an object based only on one attribute.

Based on our tests results, we have implemented new versions of the *equals* and *contains* operations that are more flexible than the original ones. These new versions include a weighted equals, in which we associate to each attribute ai a weight wi ($\Sigma wi = 1$), considering than some attributes as are more pertinent than others. We have also implemented a new version of the *contains* operation in which we allow indicating a list of objects, classes or associations that should be ignored when comparing two graphs. Through this ignore list, we reduce the size of the graphs and we prevent comparing instances that are associated with the context objects, but that do not belong to the context description (*e.g.* profile and event objects related to context description objects).

With these new versions, our tests pointed out that the execution of the filtering process does not represent a significant bottleneck for WIS. The Table 1 summarizes one of the test cases we have performed. In this test case, we compared *n* objects in the knowledge base using a given operation and evaluate the memory and time consumption for executing *n x (n-1)* times the operation. We observe that the pattern matching algorithm implemented by *contains* operation is not time consuming, even if it establishes subgraph relationships. However, it is memory consuming, since AROM system keeps the knowledge base active in the memory. Thus, we believe that this implementation fits the server implementation we choose by using the Web service technology for implementing the BW-$\mathcal{M}$ framework. This architecture corresponds to a client-server solution (typical for WIS) in which the filtering process is performed by the server. However, the use of this implementation in the client side may represent performance risk in constrained mobile devices, such as a smartphone. However, the recent evolution of mobile devices let us imagine that memory consuming will not represent a problem in the near future

Operation:	*contains*		
Number of runs:	88	**Avg Memory**	59,761 Kb
Number of compared objects:	28	**Avg Time**	43,454 ms
Total n° of operations: n x (n – 1) x runs	66528		

Table 1. Results of a BW-M execution test case.

7. Conclusions

In this paper, we have proposed a context-aware profile model, which allows users to indicate their preferences (concerning the information content) when in a given situation. These profiles address the fact that the user's preferences may vary according to the context in which they are using a Web-based Information System (WIS). We have proposed a filtering process that takes into account both, the user's current context and the user's preferences for this context (expressed through the proposed profiles). This process allows WIS to adapt the information delivered to mobile users by filtering it based into the user's current context. We implemented this filtering process in a framework called BW-$\mathcal{M}$, which uses an object-based knowledge base in order to keep the context information, as well as the profiles.

As future directions, we expect to extend the proposed filtering process by refining the pattern matching algorithm used to compare instances of the context description class. We are interested in calculating an acceptable semantic distance between the objects, in order to check if they are sufficiently similar (and not necessarily equal) to establish a subgraph relationship. We are also interested in using the proposed profile model and the adaptation process to service adaptation. In this case, instead of selecting available content, we intend to use the context-aware profiles to select among a set of available services those that better fit the user's current context.

8. References

[1] Ankolekar, A., Krötzsch, M., Tran, T., Vrandecic, D.: The two cultures: Mashing up Web 2.0 and the Semantic Web. Int. World Wide Web Conference (WWW 2007) – Track Semantic Web, ACM Press (2007) 825-834.

[2] Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4 (2007) 263–277.

[3] Borges, M.R.S., Brézillon, P., Pino, J.A., Pomerol, J.-Ch.: Groupware system design and the context concept. In: Shen, W. et al. (eds.): CSCWD 2004, LNCS 3168, Springer-Verlag (2005) 45-54

[4] Conan, D. Rouvoy, R., Seinturier,L.: Scalable processing of context information with COSMOS, Int. Conference on Distributed Applications and Interoperable Systems (DAIS'07), LNCS 4531 (2007) 210-224.

[5] Conlan, O., O'Keeffe, I., Tallon, S.: Combining adaptive hypermedia techniques and ontology reasoning to produce dynamic personalized news services. In: Wade, V. et al. (eds.), 4th Int. Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2006), Dublin, Ireland, Springer (2006) 81-90.

[6] Daoud, M., Tamine, L., Boughanem, M., Chabaro, B.: Learning Implicit User Interests Using Ontology and Search History for Personalization. In: Weske, M., Hacid, M.-S., Godart, C. (Eds.), Personalized Access to Web Information (PAWI 2007), Workshop of the 8th Int. Web Information Systems Engineering (WISE 2007), LNCS 4832, Springer-Verlag (2007) 325-336.

[7] Dey, A.: Understanding and using context. Personal and Ubiquitous Computing, vol. 5 n. 1 (2001) 4-7.

[8] Greenberg, S.: Context as a dynamic construct. Human Computer Interaction, vol. 16 n. 2-4 (2001) 257-268.

[9] Grudin, J.: Desituating action: digital representation of context, Human-Computing Interaction, vol. 16, n° 2-4 (2001) 269-286.

[10] Hightower, J., LaMarca, A., Smith, I. E.: Pratical lessons from Place Lab, IEEE Pervasive Computing, vol. 5 n. 3 (2006) 32-39.

[11] Kirsch-Pinheiro, M., Gensel, J., Martin, H.: Representing Context for an Adaptative Awareness Mechanism. In: 10th Int. Workshop on Groupware (CRIWG 2004), LNCS 3198, Springer-Verlag (2004) 339-348.

[12] Page, M., Gensel, J., Capponi, C., Bruley, C., Genoud, P., Ziébelin, D., Bardou, D. and Dupierris, V.: A New Approach in Object-Based Knowledge Representation: the AROM System. 14th Int. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, LNAI 2070, Springer-Verlag (2001) 113-118.

[13] Preuveneers, D.; Vandewoude, Y.; Rigole, P.; Ayed, D., Berbers, Y.: Context-aware adaptation for component-based pervasive computing systems, Advances in Pervasive Computing 2006, Adjunct Proceedings of the 4th Int. Conf. on Pervasive Computing (2006) 125-128.

[14] Rocha, R. da; Endler, M.: Context Management in Heterogeneous, *Evolving Ubiquitous Environments*, IEEE Distributed Systems Online, Vol. 7, No.4, April 2006.

[15] Rouvoy, R., Eliassen, F., Floch, J., Hallsteinsen, S., Stav, E.: Composing Components and Services using a Planning-based Adaptation Middleware. In: Pautasso, C., Tanter, E. (Eds.), 7th Int. Symposium on Software Composition (SC'08), LNCS 4954, Springer (2008) 52–67.

[16] Yang, S.J.H., Shao, N.W.Y.: Enhancing pervasive Web accessibility with rule-based adaptation strategy. Expert Systems with Applications, vol. 32, n. 4, (2007) 1154-1167.