A Personalized and Context-Aware Adaptation Process for Web-Based Groupware Systems

Manuele Kirsch-Pinheiro, Marlène Villanova-Oliver, Jérôme Gensel, Hervé Martin

Laboratoire LSR – IMAG
BP 72 – 38402 Saint Martin d'Hères CEDEX - France
{ kirsch, villanov, gensel, martin }@imag.fr

Abstract. The evolution of mobile technologies, like web-enable cellphones, PDAs and wireless networks, makes it now possible to use these technologies for collaborative work through web-based groupware systems. However, due to the limitations of these technologies, some adaptation is essential in these systems. In order to adapt their behaviour to the user's context, groupware systems must be built as context-aware systems. Besides the characteristics of the user's context, we believe that the user's preferences should also be taken into account by the adaptation process. In this paper, we present a two-fold approach for adapting the informational content delivered to a mobile user by web-based groupware systems: we propose a filtering mechanism which considers both the current context and preferences for this context. The notion of context is represented through an object-based model we have proposed, which takes into account the user's physical context as well as the user's collaborative context, including elements dedicated to the collaborative process in which the user is involved (notions of group, role, activity, etc.). The user's preferences are represented by a set of pre-defined profiles, which are exploited by the adaptation process in order to organize the delivered information into several levels of detail, based on a progressive access model. The proposed filtering mechanism is performed in two steps: first, it analyses the user's current context and selects, among the user's pre-defined profiles, those which have been defined for a situation found in this context. Second, it uses the progressive access model in order to filter and organize the available information, according to the selected profiles.
Keywords. context-aware computing, user adaptation, progressive access, web-based groupware systems, CSCW.

1 Introduction

Mobile technologies, such as PDAs, cellphones and wireless networks, are more and more adopted by mobile workers in order to collaborate with their colleagues. Web-based groupware systems, which allow collaborative work through a web system, have now to cope with these technologies. However, although these technologies have evolved, they still have several limitations (see [12][11]). We may cite, for instance, the reduced display size and the limited memory and battery life capacities of mobile devices such as PDAs, or the limited bandwidth of wireless networks. Additionally, thanks to these technologies, users are not anymore constrained to access the system using a unique device or from a unique location. The same user now may access the system and collaborate with her/his colleagues from different locations (*e.g.* from home, from the airport, from the office…) or using different devices (a laptop, a PDA, a cellphone, etc.). Consequently, we cannot predict the circumstances under which the user will access the system. This mobility, as well as the limitations of mobile technologies, requires systems with some adaptation capacities. Systems should adapt

the informational content, the presentation and/or the services to the users accordingly to the technologies and to the circumstances in which the users are acting.

The adaptation of a system according to the circumstances in which a user is accessing it is usually associated with the notion of *context-aware computing* [2][5]. Context-awareness is the capability of perceiving the user situation in all its forms, and of adapting in consequence the system behaviour, *i.e.*, the services, the data and the interface. The adaptation is therefore the goal of context-awareness [3]. Commonly, context-aware systems limit the notion of context, referring to the situation in which the user is acting, to the concepts of user's location and device (see [2][15][18]). For instance, some systems propose to adapt the presentation of a given information according to the capabilities of the user's device (such as in [15]), others propose the selection of a particular informational content according to the user's location (such as in [2][11]). Nevertheless, this approach presents some drawbacks. In the one hand, the user's preferences related to the delivered information are often ignored. We believe that taking into account the user's preference may improve the adaptation process. This is particularly interesting for groupware systems in which users play roles that demand an adapted informational content. In the other hand, groupware systems should consider, as part of the user's context description, other elements related to the collaborative process (for instance, the notions of group, role, activity, shared object, and so on), since users on these systems are involved in such a process.

In this paper, we propose a new approach for adapting the informational content delivered by web-based groupware systems to a mobile user. In this approach, adaptation is performed by a twofold filtering mechanism, which filters the available information based on the user's current context and on the user's preferences for this context. We have proposed in [13] an object-oriented model representing context which takes into account both the physical context (location, device, etc.) and the collaborative context (collaborative process, group, role, etc.). In this paper, we use this model to represent the user's context in order to adapt the information content delivered by a Web-based groupware system to the user's current context. The user's preferences are represented through a set of pre-defined profiles, which allow the organization of the delivered information into several levels of details. This organization is based on the Progressive Access Model (PAM) we have proposed in [20]. In this paper, we propose new operations to navigate through these levels. These operations ("next", to present the content of the next level, and "previous", to show the previous one) are particularly concerned with the use of mobile devices and their limitations. The proposed filtering mechanism is therefore performed in two steps: first, it analyses the user's current context and selects, among the user's pre-defined profiles, those which match the user's current context. Second, it applies the progressive access model according to the selected profiles, filtering and organizing the available information.

This paper is organized as follows: Section 2 presents the models used by the proposed filtering mechanism (the context model, the user's preferences model, and the PAM). Section 3 presents the filtering mechanism, detailing each step, whilst Section 4 discusses some preliminary results. We conclude in Section 5.

2 Related Models

In this section, we introduce the models we propose as a base to the context-aware filtering process (see Section 3). This process uses the following interconnected models: (*i*) a *context model* to represent the user's context; (*ii*) the *progressive access model* to select and organize the informational content; (*iii*) a model to describe this *informational content* (*i.e.*, the awareness information) and the *user's preferences* model. These models are described in the sections below.

2.1 An Object-Oriented Model of the User's Context

Any groupware system, in order to exploit the user's context, has to represent somehow the notion of context. However, in order to represent this notion, we need to understand and define it. When looking at the context-aware computing literature, one may perceive that there is no single definition (see, for instance, [5][6][16]). In one of the pioneers works, Schilit *et al.* [18] define context as "the location of use, the collection of nearby people and objects, as well as the changes to those objects over time". A largest view is given by Dey [5], who defines context as "*any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the applications themselves*". In this work, we adopt this definition since it applies particularly for designing context-aware systems.

Inspired by this definition, we have proposed [13] an object-oriented representation of context, which focuses on a mobile use of Web-based groupware systems. Based mainly on a set of UML diagrams, this model represents both the user's physical context (including the concepts of *location*, *device* and *application*) and the user's collaborative context (which includes the concepts of *group*, *role*, *member*, *calendar*, *activity*, *shared object* and *process*). We claim that a groupware system should take into account the collaborative context, since the users of such systems are also involved in some collaborative process. Consequently, some information related to the group, such as its composition, its activities, its goals, etc., can be considered as relevant for such users, and consequently should be included into the user's context. The Fig. 1 shows all the concepts that constitute this model.

In this model (see Fig. 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 is not an isolated information but does belong to a more complex representation of the user's situation. For instance, we consider that a member belongs to the group through the roles she/he plays in this group, and that each group defines a process, which is composed by a set of activities (or tasks, also composed by subtasks), and so on. A complete description of these associations is given in [13].

The context of a user (member of a group) is then represented in this model by an instance of the class context description, which is linked by composition to instances of the class context element and its subclasses (see Fig. 1). The Fig. 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), which uses a groupware system with collaborative edition and asynchronous communication (through annotations) services. Let us suppose that Alice is accessing this groupware system through her PDA in order to consult the latest notes about a report that her team is writing. When Alice requests these notes, her current context can be represented by the context description object represented in the Fig. 2b. 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 ‘*report2005’* through the ‘*notes’* application, etc. All these associations, as well as the context elements objects connected by them compose the current Alice’s context.

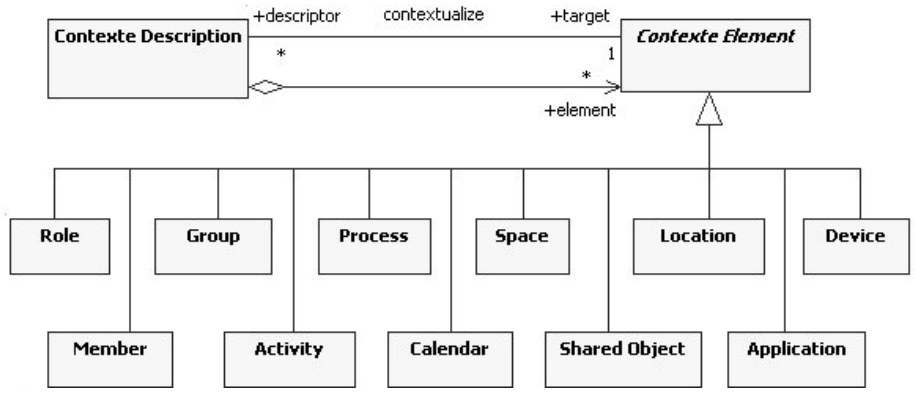

Fig. 1. A context description is seen as a composition of context elements.

These classes form the schema of a knowledge base (KB), which allows us to describe the context of a user accessing the groupware system. This KB encompasses three kinds of knowledge. First, the classes and associations related to the system and the working environment are defined and instantiated. For instance, considering a collaborative editor, this knowledge refers to the definition of the classes and instances corresponding to the edition process and its component activities, the group’s members, the application (services) the system offers, etc. Second, the KB stores the descriptions of potential contexts established for the different users (cf. section 2.3). Third, the KB keeps the instances of context description that represent the current context of the active users. These instances represent a knowledge that is created and dynamically updated by the system during each user’s session, according to her/his behaviour. This knowledge can be discarded once the user is no longer active. In the opposite, the instances corresponding to the other kinds of knowledge are permanently stored in the KB, and they may evolve, following the evolution of the work performed by the group. We exploit the elements of this model in the filtering process (cf. section 3).

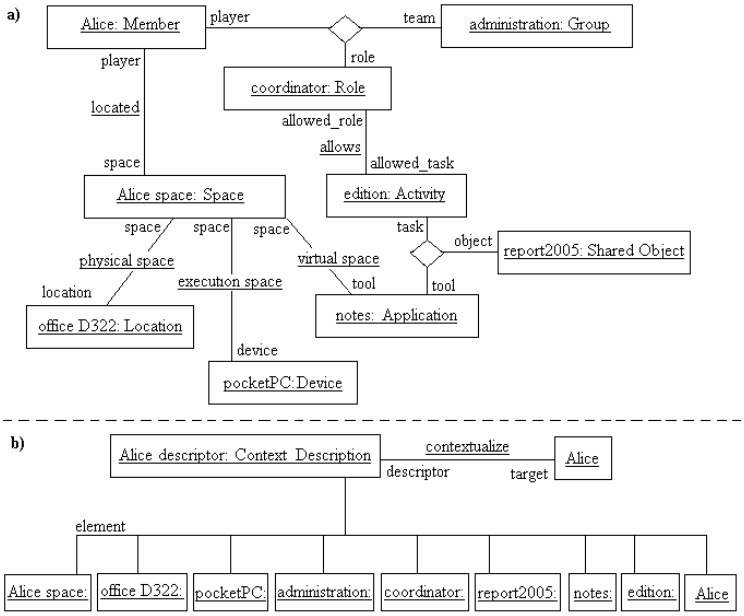

Fig. 2. Example of a context description for a given user (Alice).

2.2 The Progressive Access Model

The central idea behind the notion of *progressive access* is that a user does *not* need to access *all* the information *all* the time. The goal is to make the system able to deliver *progressively* a personalized information to its users: first, information considered as essential for them is provided, and then, some complementary information, if needed, is available. The Progressive Access Model we have proposed in [20] is a generic model, described in UML, which allows the organization of a data model in multiple levels of details according the user's interests. This model is based on some basic definitions that we present below (more details can be found in [20]).

The notion of progressive access is related to the one of *Maskable Entity*. A *Maskable Entity* (*ME*) is a set of at least two elements (*i.e.* |ME| ≥2) upon which a progressive access can be set up. The progressive access to a ME relies on the definition of *Representations of Maskable Entity* (*RoME*) for this ME. These RoME are subsets of the ME ordered by the set inclusion relation. Two kinds of RoME, *extensional* or *intensional*, are distinguished. *Extensional RoME* are built from the *extension* (*i.e.* the set of elements) of the ME. In the case where the ME is a set of structured data having the same type, *intensional RoME* can be built from the *intension* of the ME. The intension of a structured ME is defined as the set of descriptions of variables, which constitute the structure of the ME. Whatever its nature – extensional or intensional –, each RoME of a ME is associated with a level of detail. Thus, RoME_i is defined as the RoME of a ME corresponding to the level of detail *i*, where $1 \leq i \leq max$, and *max* is the greatest level of detail available for this ME.

The Fig. 3 shows a ME with three associated RoME. Some rules impose that a $RoME_{i+1}$ – whether it is extensional or intensional – associated with the level of detail *i+1* ($1 \le i \le max-1$), contains at least one more element than $RoME_i$. A stratification for a ME is a sequence of RoME ordered by set inclusion as illustrated in the Fig. 3. Please note that several and different stratifications can be defined for a given ME.

The progressive access relies traditionally on two operations that allow to switch from a RoME to another within a given stratification:

- from a $RoME_i$, at level of detail *i*, the *m ask operation g*ives access to the $RoME_{i-1}$, and to its possible predecessors as well, at level of detail i-1:

  $mask(RoME_i)=RoME_{i-1}$, where $2 \le i \le max$
- from a $RoME_i$, at level of detail *i*, the *unmask operation* gives access to the $RoME_{i+1}$ at level of detail i+1:

  $unmask(RoME_i)=RoME_{i+1}$, where $1 \le i \le max-1$

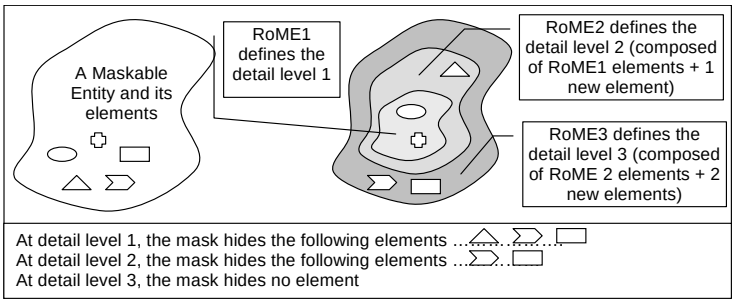

Fig. 3. A Maskable Entity with three RoMEs.

When applying the *unmask* operation, the content of the next level is added to the content of the previous levels. In other words, the user will have all the previous information as well as the information selected by the next level. When the user is using a mobile device, this way of access can reveal itself inappropriate. For instance, let us consider a user consulting the address book on her/his cellphone. In this case, the ME is the address book, and we suppose a stratification S composed by two levels: $RoME_1$ = {contact's name, contact's phone} and $RoME_2$ = {contact's address}. Thus, at the first level, the user can consult the contacts' names and their phone numbers (content of the $RoME_1$). By applying the *unmask* operation, the user goes to the second level, at which she/he will get the contact's address (content of the $RoME_2$) as well as the contacts' names and phones, which is the content of the $RoME_1$. Considering the limited display size common to cellphones, the information of the $RoME_1$ that remains when presenting the $RoME_2$ can be seen as useless, since the user has already seen it. In this case, it is more interesting to present to the user only the content of the current level, hiding the elements from the previous one.

Therefore, we propose here two extra operations that can apply in mobile environments. These operations, *next* and *previous*, intend to allow the transition from a RoME to another one within a given stratification. Unlike *unmask* and *mask* operations, *next* and *previous* present the information of each level separately. These new operations do not preserve the information of the previous levels when presenting the next or previous one. These operations are defined as follow:

- first of all, let us define a basic operation called *proper(i)*, which isolate the elements included at a level *i*:

  proper($RoME_i$) = $RoME_i$ if i=1, otherwise *proper*($RoME_i$) = $RoME_i$ – $RoME_{i-1}$, where $2 \leq i \leq max$.
- then, from a $RoME_i$, at level of detail *i*, the *previous operation g*ives access *only* to the elements of the $RoME_{i-1}$ at level of detail i-1:

  previous(RoMEi)= proper(i-1), where $2 \leq i \leq max$.
- and from a $RoME_i$, at level of detail *i*, the *next operation* gives *only* access to the elements of the $RoME_{i+1}$ at level of detail i+1:

  next(RoMEi)= proper(i+1), where $1 \leq i \leq max-1$.

Therefore, the Progressive Access Model (PAM) is defined as a generic UML class diagram (see Fig. 4) which allows the description of well-formed stratifications for maskable entities. Fig. 4 shows all the definitions mentioned before, notably the stratification, represented by the class "*Stratification*", which is seen as an aggregation of, at least, two *ordered* instances of the class "*Representation of Maskable Entity*". An instance of this class is linked by the association *adds* to one or more instance(s) of the class "*Element of Maskable Entity*" which are the elements of the ME added by the $RoME_i$ at the level of detail *i*. The dependency relation ({*subset*}) ensures that the added elements belong to the set of elements corresponding to the ME.

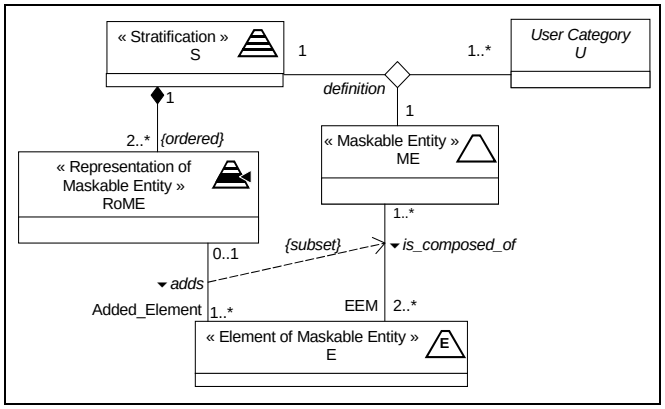

Fig. 4. The Progressive Access Model described using UML stereotypes.

An important characteristic of the PAM is the ternary association called *definition* which links the classes "*Stratification*" (S), "*Maskable Entity*" (ME) and "*User Category*" (U). The class "User Category" is an abstract class which allows the connection with any user model so that each described user can benefit from a personalized progressive access to a given maskable entity. In this work, we consider as *maskable entities* the informational content delivered by a groupware system, which is represented through a superclass *event* (cf. section 2.3).

2.3 Modelling the Informational Content and the User's Preferences

In order to simplify the filtering process, we have modelled the informational content that can be delivered to the user through a superclass named *event*. An event may contain any useful information (for the group's members) about a specific topic related to the collaborative process. The idea is to describe events that are interesting for users in a mobile situation, such as the presence of a member of the group in a given location. In this work, we are particularly interested by the *awareness information*. Awareness in groupware systems refers to cooperating actors taking heed of the context of their joint effort [19]. It refers to the knowledge and the understanding a user has about the group itself and her/his colleagues' activities, providing a shared context for individual activities in the group [7]. Awareness information is pointed out by CSCW community as crucial for the success of a cooperative work [19], since it helps in forming a common ground for individual and cooperative actions. However, active groups tend to produce large amounts of awareness information, and users may feel overloaded by it. This is particularly true when considering mobile users, who are subject to multiple constraints (physical and environmental constraints), such as devices with limited capacities (a cellphone, for instance), or inadequate environments (like working in a rail station). Thus, mobile users need awareness information that is adapted to their current situation, *i.e.*, adapted to their current context. In this work, each event represents a piece of awareness information.

The set of events is defined by the system designer, which should define them according to the interests the users may have when working in the groupware system. The *event* class should be specialized by the system designer when developing the groupware system. This class contains some attributes that we consider as essential when describing awareness information (see Fig. 5): an event name (for instance, "Alice's presence"), a description (*e.g.* "Alice is online from the office D322"), some details about it (*i.e.* a more detailed description, like "Alice is online since 8h30am from the office D322, 3rd floor"), a time interval in which it occurs (for instance, "from 8h30am until now"), and some media describing its content (e.g., a photo of Alice). We also consider the event as referring to one or more elements of the context model (see concerns association in Fig. 5), since it may carry information related to these elements (for instance, the event referring to the Alice's presence is linked to the object of the Member class that refers Alice). Additionally, each event instance is associated with a context description instance, representing the context in which the event is or has been produced (for instance, the location object related to the office D322).

We consider the event class (and its subclasses) as a *maskable entity* when applying the progressive access model. In other words, stratifications can be defined for the *event* class and its instances. We may define *intensional stratifications*, associating the attributes of the event class (its intension) with the levels of detail (for instance, a stratification $S_1^{intensional}$ = {{*name, description*}, {*interval, details*}, {*medias*}} may be defined), as well as *extensional stratifications*, selecting the instances of the event class according the value of their attributes (*e.g.* $S_2^{extensional}$ = {{*event.interval during* DAY}, {event.interval *during* (WEEK)}}).

The user's preferences are represented through the notion of *profiles*. A profile represents the preferences and the constraints the system should satisfy for a given element (a group member, a role, a device…). It includes the description of a *potential context* that may characterize a user's situation (called the *application context*), and defines *filtering rules* that should apply when this situation happens (*i.e.* when the user's current context matches the application context, cf. section 3.1). The filtering rules are based on a set of stratifications and reflect the user's preferences considering the context associated with the profile. In other words, a profile describes what information the user wants to be informed of when in this situation and how this information is organized.

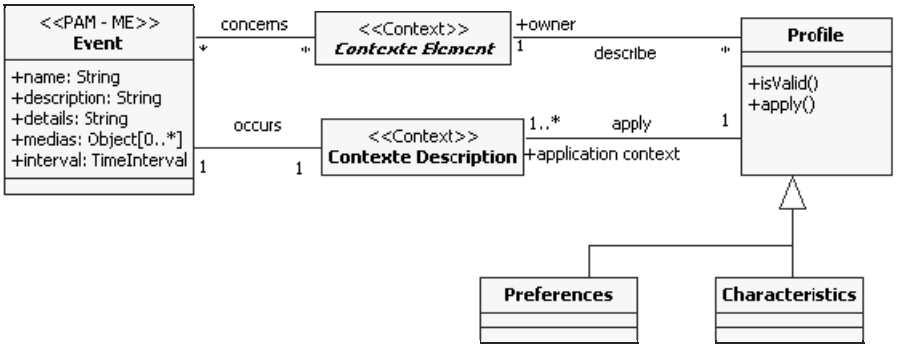

Fig. 5. UML class diagram describing the event and the profile class.

Each profile can be seen as a set composed by:

1. an *owner*, for who/what the profile is defined. The owner is represented by a context element object (association describe in Fig.5), allowing the definition of a profile either for users or for any element of the context model;
2. at least one *application context* to be considered, which represents the potential contexts in which this profile can be applied, *i.e.* the situations in which the profile is valid (association apply in Fig.5) and should be selected by the filtering mechanism (*cf.* Section 3);
3. a list of *event* subclasses whose instances can be selected (association sign up in Fig. 6), representing the informational content considered as relevant for the owner;
4. a set of *stratifications* defined for the owner (see Fig. 6). The stratifications as well as the list of events define the filtering rules related to the profile.

The basic idea behind the profiles is to allow users, system designers and administrators to define the profiles and the application context objects related to them. In other words, each user may determine what information she/he considers as relevant and in which circumstances, as well as how this information is organized in levels of detail. A system designer may create profiles for the supported devices according to the capabilities of each device type, and a system administrator may also define profiles for some particular roles, like the coordinator role, which have special

needs considering the informational content. This last case is particularly interesting for groupware systems, in which the roles represent the rights and the responsibilities of each team member.

Indeed, we have observed that often mobile users do access groupware systems from a limited number of well known situations: from the rail station, from the airport, from home, using a particular PDA, a laptop, etc. By describing these potential situations (the corresponding application context) and defining particular profiles for them, each user (or the team manager) may express her/his needs and preferences for the most common situations, allowing the system to better adapt the delivered content in these situations. For instance, considering a cooperative editor (as the one proposed in [9]), the system designer may pre-define some event subclasses 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 the "comment" events. This profile could be related to the situations in which she is using her PDA (*i.e.* a profile that is valid when she is using this device) and organize the "comment" events according to the stratification S_1^{int} = {{*name, description*}, {*interval, details*}, {*medias*}}.

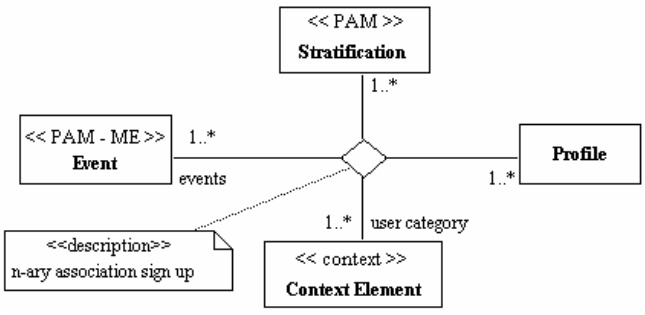

Fig. 6. The association concerning the profiles and the stratifications.

Additionally, it is worth to note that the multiplicity of the association sign up allows the definition of many stratifications for a given event class, by assigning it to different stratifications or to different profiles (and, consequently, to different *application context*). In the other hand, it is important to observe that a profile may concern any given context element (user, group, role, device…). We believe that, for groupware systems, it is particularly interesting for roles, defining the role's needs considering the delivered awareness information (*e.g.* a team coordinator needs a global view of the team performance, which may be unnecessary for a simple participant). It is equally interesting to define profiles for devices, describing the characteristics and limitations of a given device. Thus, as we can see in the Fig. 5, we specialize the profile class in preferences, describing the preferences of the user or her/his role, and in characteristics, describing the capabilities of the referred.

3 A Personalized and Context-Based Filtering Process

The adaptation approach we propose here is based on a filtering process in *two steps*. The first step selects the profiles the system should apply in order to filter the available information, according the user's current context. The second step consists in applying the filtering rules defined by the selected profiles. These rules are based on a set of pre-defined stratifications, which filters and organizes the set of available events that represent the available informational content. We assume that each user (or the system designer or its administrator) may define several profiles and the situations in which they are valid (*i.e.* the description of each *application context*). Thus, the first step selects, among the available profiles, those the system should apply, and the second step applies the stratifications defined in the selected profiles.

3.1 Step 1: Selecting Profiles According the User's Context

The first step of the proposed filtering process consists in selecting the profiles that are valid with regard to the user's current context. This selection is performed by comparing the *application context* related to the available user's profiles with the user's current context. Please note that these two kinds of context are both instances of the class context description of the context model. For each profile, we test if one of its *application contexts* has the same content or is a subset of the *user's current context description*. In other words, we verify if 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* instance and the *context element* instances associated with it define a graph, where the nodes represent the instances and the edges between them represent the tuples of associations involving these instances. 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 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 (tuples of the same association) 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 report, as in Fig. 2. Let us suppose that Alice has defined two profiles: *(i)* a first one that is applicable only when she is involved in an activity which concerns a given report and when she is working on her desktop device. This means that the *application context* related to this profile includes

the context elements corresponding to the shared object *report2005* and to the device *desktop*. *(ii)* And a second profile that includes in its *application context* an instance referring to her PDA (*i.e.* a profile for the situations in which she is using this device). When Alice finds herself in the situation described by the Fig. 2, only the second profile is selected. The first one is rejected since the context description object representing the application context of this profile does not match the user's context description (the latter does not include a node referring to the desktop device that is present in the former, so it does not contains application context of this profile). The Fig. 7 represents the graphs defined by context description objects related to Alice's current context (Fig. 7a) and her profiles (Fig. 7b and Fig. 7c respectively). In this figure, we observe that the graph defined by the application context of the first profile is not a subgraph of the graph defined by the Alice's context, while the graph defined by the application context of the second profile is a subgraph of the one defined by Alice's context.

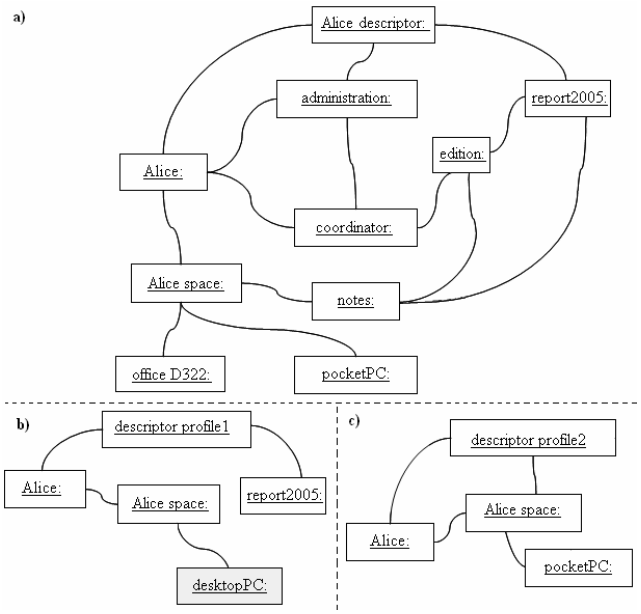

Fig. 7. The graphs defined by a user's current context (a), and the application context of two profiles (b and c).

3.2 Step 2: Applying the Progressive Access Model

Once the profiles have been selected, the second step of the filtering process applies the stratifications defined in the selected profiles. These stratifications filter and organize the available set of events. It is worth noting that, in order to respect the stratification definition, we apply one profile at time by ordering the profiles. This is achieved by an algorithm in four steps: *(i)* it orders, by priority, the selected profiles; *(ii)* for each profile, it selects, among the set of available events, all events whose class corresponds to a class signed up by the profile; *(iii)* it performs the stratifications

associated with this profile, by applying the extensional stratifications before the intensional ones; *(iv)* it delivers the content of the first level of these stratifications.

The priority of a profile is defined by a similarity measure between the application context of the profile and the user's current context. This measure evaluates the matching degree of the application context with the user's context. In other words, it estimates the proportion of elements of the graph defined by the user's context that have equals elements in the graph defined by the application context. 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 instances composing 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 C_p; otherwise $x = | X | / | C_u |$ with $X = \{ x \mid x \; equals \; y, x \in C_u, y \in C_p \}$

As an illustration, let us consider once again the example of a collaborative editor. The designer of such a system may have defined three event classes: *user session*, *document changed* and *new comment*. Let us consider now that a user (*Alice*) has two selected profiles. The first profile signs up the first event class, defining one stratification S_1^{int} = {{name, description}, {interval, details}, {medias}}. The second profile signs up the second and the third event classes, defining two stratifications, S_2^{int} = {{name, interval}, {description}} and S_3^{ext} = {{event.interval during DAY}, {event.interval during WEEK}}. Considering that the first profile has 0.2 as priority order, and the second has 0.8, the filtering process will apply first the stratifications S_2 and S_3 to the *document changed* and *new comment* instances, and then it will apply S_1 to the *user session* instances. As a result, the filtering process will make available for the user *Alice*, at the first level, the *document changed* and *new comment* events instances whose *interval* corresponds to a period of the current day, presenting only their attributes *name* and *interval*.

At the end of the proposed filtering process, an organized (in levels of details) set of events will be available for delivering to the mobile user. This set will probably better suit the current mobile user's context since it accords the user's preferences defined for this context.

4 Preliminary Results and Discussion

We have implemented the proposed filtering process using the AROM system [17] and a framework for awareness support called BW-M [14]. AROM is an object-based knowledge representation system, which adopts classes/objects and associations/tuples as main representation entities. Using AROM, we have created a knowledge base (KB) in which we keep the instances of the context model as well as the profiles and the events. Using the BW-M framework, we have implemented the filtering process using the algorithms presented in Section 3.

Using these algorithms, we have performed some tests simulating situations that represent the use of a collaborative web system which provides shared repository and synchronous/asynchronous communication features. The simulation uses five users and fifteen profile definitions, and we have evaluated different types of pattern matching algorithm acting on the *equals* and *contains* operations, and on the similarity

measure *Sim*. We have evaluated two versions of the *equals* operator (one that considers only perfect match – objects must be exactly equal – and another that allows to define a minimum set of similar attributes in the objects) and analyzed the effects of these versions on the *contains* operator and on the *Sim* measure. We have also tested two versions of the *contains* operator (one that compares only equal instances, and another that compares the graph from two instances, even if they are not equal).

These tests have showed the validity of our filtering process with regard to a user's current context, the profiles being applied as expected in most cases and the total amount of delivered information being reduced in all cases. These tests have also pointed out some critical aspects. First, using distinct versions of the *equals* and *contains* operators leads to different results: the most satisfactory ones are those obtained with the most flexible versions (*equals* with predefined limit, and *contains* with the comparison of different instances). Then, the definition of the *application context* related to a profile is more or less critical given the version of the operator. In fact, defining a detailed application context causes the non selection of the profile (or the attribution of a lower priority to the profile with the *Sim* measure) in most cases when using the first version of the operators. On the other hand, defining a minimal application context causes its selection in almost all cases, especially when using the second version of the operators. However, the use of the most flexible versions of the *equal* operator does not affect the priority of these profiles, as those with detailed application context are always better ranked than those with a reduced application context.

5 Conclusions

In this paper, we have presented a context-based filtering process that proposes to adapt the information delivered to mobile user by filtering it according to the current user's context and to the user's preferences for this context. This approach differs from other approaches in context-aware systems, such as [2][10][15][18], by the use of these preferences, and it differs from traditional approaches on groupware systems, such as [8][19], by the use of context knowledge. We believe that, by allowing a direct participation of the user into the adaptation process, this process may provide results that are more adequate to the user's expectations. We also differ from other works, such as [2][15], by a largest vision of what is context, as our approach takes into account both the user's physical context and the user's collaborative context. Moreover, our context model is mainly concerned by representing context information, instead of concerning its acquisition process, such as in [5].

Additionally, we have also proposed two new operations (*next* and *previous*) for navigation purposes in the Progressive Access Model. These operations are particularly interesting for a mobile use, since they limit the information presented to the user in a given moment. By using these new operations and the Progressive Access Model, we not only filter the awareness information (such as in [14]), but we also organize it according to its relevance for the user. This represents a clear improvement, since mobile users usually do not dispose of enough time to analyze all information. Organizing the delivered information by relevance becomes then necessary, as pointed out by Billsus *et al.* [1] and by Coppola *et al.* [4].

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 studying how to simplify the profile definition in order to reduce problems that may be caused by an incorrect definition.

6 References

[1] Billsus, D., Brunk, C.A., Evans, C., Gladish, B., Pazzani, M.: Adaptative interfaces for ubiquitous web access. Communications of the ACM, 45 (5), May 2002, ACM Press (2002) 34-38.

[2] Burrell, J., Gray, G.K., Kubo, K., Farina, N.: Context-aware computing: a text case. In: Borriello, G., Holmquist, L.E. (eds.): 4th Int. Conf. on Ubiquitous Computing, LNCS 2498. Springer (2002) 1-15

[3] Chaari, T., Laforest, F., Celentano, A.: Design of context-aware applications based on web services. Technical Report RR-2004-033, LIRIS, Lyon, France (2004) http://liris.cnrs.fr/publis/rr_html

[4] Coppola, P., Mea, V.D., Gaspero, L. Di and Mizzaro, S.: The concept of relevance in mobile and ubiquitous information access. In: Frestani, F., Dunlop, M. and Mizzaro, S. (eds.): Int. Workshop on Mobile and Ubiquitous Information Access: Mobile, LNCS2954, Udine, Italy, Sept. 2003. Springer-Verlag (2004) 1-10.

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

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

[7] Dourish, P., Bellotti, V.: Awareness and Coordination in Shared Workspaces, ACM Conference on Computer-Supported Cooperative Work, ACM Press (1992) 107-114.

[8] Fernández A., Haake J.M. and Goldberg A.: Tailoring group work. In: Haake J.M. and Pino J.A. (eds.): Int. Workshop on Groupware (CRIWG 2002), LNCS 2440, Springer-Verlag, (2002) 232-242.

[9] Guerrero L.A., Piño J.A, Collazos C.A., Inostroza A., Ochoa S.F.: Mobile support for collaborative work. In: Vreede, G.-J., Guerrero, L.A. and Raventós G.M. (eds.): X International Workshop on Groupware (CRIWG'04), LNCS 3198, Springer-Verlag (2004) 363-375.

[10] Ibach P., Tamm G. and Horbank M.: Dynamic Value Webs in Mobile Environments Using Adaptive Location-Based Services. Proceedings of the 38th Hawaii International Conference on System Sciences, Big Island Hawaii, January 03-06, IEEE Computer Society (2005) 208-217.

[11] Islam, N. and Fayad, M.: Toward ubiquitous acceptance of ubiquitous computing'. Communication of ACM 46 (2), ACM Press (2003) 89-92.

[12] Jing, J., Helal, A.S., Elmagarmid, A.: Client-server computing in mobile environments. ACM Computer Surveys, vol. 31, n. 1 (1999) 117-157

[13] Kirsch-Pinheiro, M., Gensel, J., Martin, H.: Representing Context for an Adaptative Awareness Mechanism. X Int. Workshop on Groupware (CRIWG'04), LNCS 3198. Springer (2004) 339-348

[14] Kirsch-Pinheiro, M., Villanova-Oliver, M., Gensel, J. and Martin, H.: BW-M: A Framework for Awareness Support in Web-Based Groupware Systems. Proceedings of the 9th International Conference on CSCW in Design, Coventry, UK, May 2005 (2005) 240-246.

[15] Lemlouma, T., Layaïda, N.: Context-Aware Adaptation for Mobile Devices. IEEE Int. Conf. on Mobile Data Management, IEEE Computer Society (2004) 106-111

[16] Mostéfaoui, G.K., Pasquier-Rocha, J. and Brézillon, P.: Context-aware computing: a guide for the pervasive community. Proceedings of the IEEE/ACS International Conference on Pervasive Services (ICPS'04), IEEE Computer Society (2004) 39-48.

[17] 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 International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems, (IEA/AIE2001), LNAI 2070, Springer-Verlag (2001) 113-118.

[18] Schilit, B., Adams, N. and Want, R.: Context-aware computing applications. IEEE Workshop on Mobile Computing System and Applications, IEEE Computer Society (1994) 85-90.

[19] Schmidt, K.: The problem with 'awareness': introductory remarks on 'Awareness in CSCW'. Computer Supported Cooperative Work 11 (3-4), Kluwer Academic Publishers (2002) 285-298.

[20] Villanova, M., Gensel, J., Martin, H.: A progressive access approach for web based information systems. Journal of Web Engineering (JWE), vol. 2, n. 1 & 2 (2003) 27-57