Journal of Web Engineering, Vol. 2, No.1&2 (2003) 027-057

A PROGRESSIVE ACCESS APPROACH FOR WEB-BASED INFORMATION SYSTEMS

MARLENE VILLANOVA-OLIVER
Laboratory LSR-IMAG, Grenoble, France
marlene.villanova@imag.fr

JEROME GENSEL
Laboratory LSR-IMAG, Grenoble, France
jerome.gensel@imag.fr

HERVE MARTIN
Laboratory LSR-IMAG, Grenoble, France
herve.martin@imag.fr

Received March 9, 2003
Revised September 19, 2003

Web-based Information Systems (WIS) are used for processing and diffusing large amounts of information over the Internet. It is therefore crucial to adapt to users both the content and the presentation of information in order to save them from some disorientation or cognitive overload syndromes. For this purpose, we introduce the notion of *Progressive Access* which aims at giving users through the WIS functionalities a flexible and personalized access to data, by stratifying their information space. These stratifications are described by a Progressive Access Model (PAM). We present here the PAM and its connections with four other models (data model, functional model, hypermedia model, user model). We show how these five models can be exploited for the design of an adaptable WIS which integrates the progressive access approach. Design and generation of such systems are supported by a platform called KIWIS we also present in the paper and illustrate by an example.

Key words: Web-based Information Systems, design, adaptability, progressive access
Communicated by: D Schwabe

1. Introduction

Web-based Information Systems (WIS) constitute a category of Web applications. The particularity of these Web applications, unlike catalog web sites for instance [1], is that they allow to collect, structure, store, manage and diffuse information, like traditional Information Systems do, but over a Web infrastructure. Generally, WIS have to deal with large amounts of information, with distinct sources and formats, and are characterized by the complex services (based on the querying and/or the update of some databases, for instance) they propose to their users. Concerning the design of WIS, the methods have to cope with the multiple facets of WIS, namely the content (data), the functionalities (services) and the hypermedia features (structure, navigation and presentation). A consensus emerges from the literature (see for instance [2][3][4]) showing that the specification and design of such a Web application generally relies on several descriptions concerning the application domain, the composition and appearance interface (often based on the dynamic generation of Web pages), the navigation in the

hypermedia structure of information. To some extent, it also relies on the description of the interaction between the content and the functionalities [5] and on the description of adaptation features [6].

Concerning adaptation, one first step in this direction is achieved by systems which allow their users to customize the interface by choosing among several options the configuration they prefer [7]. More generally, a WIS is said to be *adaptable* when the user gets the impression that the system has been specially designed for her/him. A*daptability* (and its dynamic version we call *adaptivity,* in accordance with the definition given by Stephanidis *et al.* [8]) refers to the ability a WIS has to provide its users with some relevant information with regard to their rights, needs, individual characteristics or material configurations (WAP, browser, etc.), in terms of both content and presentation. In commercial systems adaptability is most of the time a hidden feature on which the user has no control since the "raison d'être" of these systems is to be the more attractive as possible.

Relying on the Web and, consequently, on a hypermedia structure, a WIS allows its users to navigate through a wide information space. However, this benefit can turn out to be a drawback for users, causing what is called the "lost in hyperspace syndrome", but also a cognitive overload [9]. The disorientation syndrome is experienced by users who get lost and forget their initial goal when browsing an information space having a complex hypermedia structure. Also, users can be subject to a cognitive overload when they have to face a too massive and difficult to understand quantity of information. These two drawbacks are highly prejudicial to the use of the system and consequently to the life of the WIS. We claim that to alleviate users from this cognitive overload must be one of the tasks performed by of a so-called adaptable WIS. For this reason, information must be organized, managed, and displayed in a personalized way. The measurement of the quality of a WIS does not only rely on its graphical appearance, but also and mainly on its usefulness and on the way its users are efficiently given access to some relevant information.

In this paper, we introduce the notion of *Progressive Access* as a contribution to adaptability in WIS. This notion aims at structuring both the information and the functionalities provided by a WIS so that its users get a higher flexibility in their access to data. The objective is to provide each user with a personal information space, organized in different levels of detail through which she/he can navigate. We call *stratification* such an organization of information and/or functionalities. Basically, a stratification applies to a set of elements, called *Maskable Entity* (ME), and consists of an ordered set of *Representations of Maskable Entity* (RoME). Each RoME is a set of elements of a ME. A RoME defines the set of elements of the ME which are accessible at a given level of detail. Thus, from a level to another, the user can navigate and then *gradually* access to information or functionalities. By giving a progressive access to data through the stratification layers, our approach intends to spare users from both disorientation and cognitive overload risks.

The paper is organized as follows. In section 2, we first give some definitions related to the notion of Progressive Access. Then, we present a UML model, called Progressive Access Model (PAM), which allows designers to integrate a progressive access approach into a WIS. We illustrate through an example how to apply the PAM to an object data model. In section 3, the four complementary models (a data model, a functional model, a hypermedia model and a user model) we propose for the design of progressive access based WIS are described. We present the main steps of the design methodology we recommend for such adaptable WIS. Section 4 is dedicated to the presentation of KIWIS, a platform for the design and the generation of WIS. Kiwis implements and validates the progressive access approach. An example of a WIS developed using KIWIS is described in Section 5. We compare our approach with some related work in Section 6, before we conclude.

2. The Progressive Access Model

The central idea behind the notion of *progressive access* is that the user of a WIS does *not* need to access *all* the information (and functionalities) *all* the time. Our objective is to give the possibility to build a WIS which has the capacity to deliver *progressively* to its users a personalized information through a functional space which is also personalized. Whether information or functionalities are considered, the principles of the progressive access remain the same: first, information (respectively functionalities) considered as essential for them is (are) provided, and then, if needed and available, some complementary information (respectively functionalities) can be reached through navigation mechanisms. In this section, we give some definitions related to the notion of *progressive access*. Then, we present a model called the Progressive Access Model (PAM). The PAM is the central model to be included in a WIS so that it supports a progressive access approach.

2.1 Progressive Access: Definitions

2.1.1 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. The RoME of a ME are subsets 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. Moreover, 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 here defined as the set of descriptions of variables (or slots, or fields) which compose the structure of the ME. Whatever its nature – extensional or intensional –, each RoME of a ME is associated with a *level of detail*. We call RoME_i 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 for this ME. The Figure 1 shows a ME with three associated RoME. As shown in Figure 1, the union of the RoME defined for a ME may not contain all the elements of this ME, leaving some elements invisible (see for instance the star element). This shows that it is possible to exploit the progressive access approach also for some confidentiality purposes.

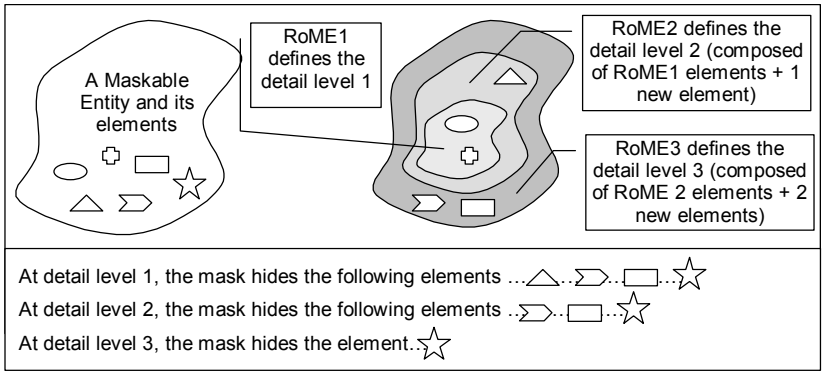


Figure 1 A Maskable Entity with three RoME corresponding to three levels of detail.

We define two functions for the progressive access, which allow to navigate from a RoME to another.

– from an intensional (respectively extensional) $RoME_i$, at level of detail *i*, the *masking function* gives access to the intensional (respectively extensional) $RoME_{i-1}$ at level of detail i-1:

$$masking(RoME_i)=RoME_{i-1}, \text{ where } 2\leq i\leq max.$$

– from an intensional (respectively extensional) $RoME_i$, at level of detail *i*, the *unmasking* function gives access to the intensional (respectively extensional) $RoME_{i+1}$ at level of detail i+1:

$$unmasking(RoME_i)=RoME_{i+1}, \text{ where } 1\leq i\leq max-1.$$

We give in the next two subsections the rules for defining valid RoME, in case of both extensional and intensional RoME. These rules impose that the $RoME_{i+1}$ – whether it is extensional or intensional – associated with the level of detail *i+1* ($1\leq i\leq max-1$) contains at least one more element than $RoME_i$. It is important to notice that extensional RoME can be seen as different ordered *masks* defined on the extension of a ME, while intensional RoME can be seen as different ordered *masks* defined on the intension of a ME.

2.1.2 Extensional RoME

Let $\mathcal{M}$ be a ME whose extension is defined as $E(\mathcal{M}) = \{e_1, e_2, ..., e_n\}$ where each $e_k \in \{e_1, ..., e_n\}$ is an element of $\mathcal{M}$.

– each extensional $RoME_{EXTi}$ defined for $\mathcal{M}$ is non-empty (*i.e.* $RoME_{EXTi} \neq \emptyset$), $\forall i \in [1, max]$

– $RoME_{EXT1} = \{e_m, \ldots, e_p\}$ with $\{e_m, \ldots, e_p\} \subset E(\mathcal{M})$ (*i.e.* $RoME_{EXT1} \neq E(\mathcal{M})$)

– for each extensional $RoME_{EXTi}$ and $RoME_{EXTj}$ defined for $\mathcal{M}$ and so that j = i+1, $\forall i \in [1, max-1]$, $RoME_{EXTj} = RoME_{EXTi} \cup \{e_r, ..., e_s\}$ where $\{e_r, ..., e_s\} \subseteq (E(\mathcal{M}) \setminus RoME_{EXTi})$

2.1.3 Intensional RoME

Let $\mathcal{M}$ be a ME whose intension is defined as $I(M) = \{a_1: t_1, a_2: t_2, ..., a_n: t_n\}$ where each $a_k \in \{a_1, ..., a_n\}$ is a variable (slot or field) of the structure of each element of $\mathcal{M}$, and each $t_k \in \{t_1, ..., t_n\}$ is the type associated with the variable a_k.

– each intensional $RoME_{INTi}$ defined for $\mathcal{M}$ is non-empty (*i.e.* $RoME_{INTi} \neq \emptyset$), $\forall i \in [1, max]$

– $RoME_{INT1} = \{a_m: t_m, ..., a_p: t_p\}$ with $\{a_m: t_m, ..., a_p: t_p\} \subset I(\mathcal{M})$ (*i.e.* $RoME_{INT1} \neq I(\mathcal{M})$)

– for each intensional $RoME_{INTi}$ and $RoME_{INTj}$ defined for $\mathcal{M}$ and so that j = i+1, $\forall i \in [1, max-1]$, $RoME_{INTj} = RoME_{INTi} \cup \{a_r: t_r, ..., a_s: t_s\}$ with $\{a_r: t_r, ..., a_s: t_s\} \subseteq (I(\mathcal{M}) \setminus RoME_{INTi})$

2.1.4 Stratification

We call *stratification* the process (and its result) which aims at defining a particular organization of a ME as a set of RoME. More precisely, an extensional (respectively intensional) stratification is a sequence of extensional (respectively intensional) RoME ordered by the set inclusion relation. It is worth noting that it is also possible to combine and compose these two kinds of stratification: an extensional (respectively intensional) stratification can be applied to an intensional (respectively extensional) stratification seen as a ME [11].

2.1.5 Operations for reorganizing a stratification

In order to reorganize an existing stratification, we propose several of operations which allow either to modify one RoME or the sequence of RoME which constitutes a stratification. The list of these operations is composed of:

- the addition and the deletion of an element of a Maskable Entity in a RoME;
- the addition and the deletion of a RoME in a given stratification;
- the merging, the splitting and the moving of a RoME within a given stratification.

These operations can be used not only for building a stratification during the WIS design phase, but also to take into account, at runtime, the evolving needs of an end-user in terms of progressive access. In both cases, these operations have to comply with the rules which define a well-formed stratification. For instance, when adding a new element to a RoME defined at the detail level *i*, one has to check that this element actually belongs to the available set of elements of the Maskable Entity, that this element does not belong to a RoME defined at a lesser level of detail, and furthermore one has to add this element to each RoME of the stratification having a level of detail greater than *i*, in order to satisfy the inclusion constraint which holds between the RoME of a stratification.

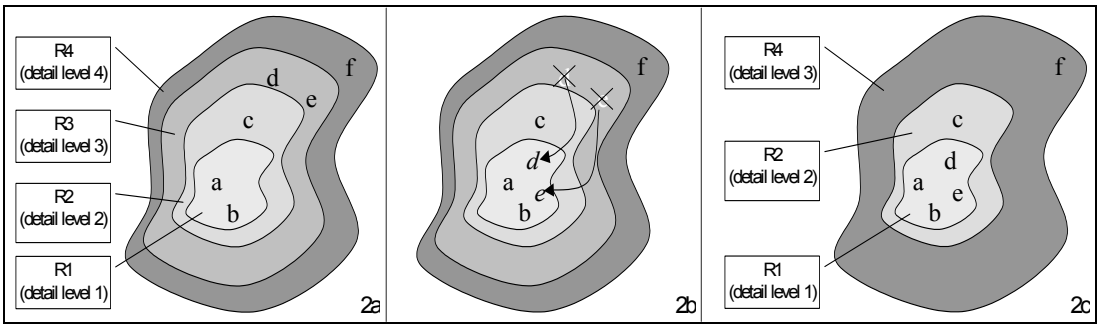


Figure 2. Merging of two RoME into one.

We here describe the *merging* operation which enables to gather two or more RoME (called sources) into one (called target). The idea is to transfer (or to add, using the addition of an element of a ME in a RoME) each element of every source RoME into the target RoME. Then, every source RoME is deleted (using the deletion of a RoME). Figure 2 illustrates the merging of two RoME, R1 and R3 into R1 in a given stratification S (see Figure 2a). Elements of R3 (namely, *d* and *e*) are transferred into R1which now gives access to the elements *a*, *b*, *d*, and *e* (see Figure 2b). Then, R3 is deleted. The new stratification S now holds three levels of detail (see Figure 2c) and contains three RoME R1, R2 and R4.

2.2 Description of the Progressive Access Model

2.2.1 Generic Description

We present in this section the Progressive Access Model (PAM) which can be used to describe stratifications over an information space or a set of functionalities. In return, this model gives the rules for specifying well-formed stratifications. The description of the PAM we give here exploits UML stereotypes [12] which extend the UML vocabulary: modeling elements are associated with a new

meaning introducing the progressive access approach. Figure 3 shows the main classes and associations that constitute the minimal PAM. The instantiation of the PAM leads to the definition of a set of stratifications for a set of maskable entities. Moreover, it allows to associate a set of stratifications with a (group of) user(s). The generic model presented in Figure 2 is valid whether the stratification is intensional or extensional. Please note that the model presented here is simplified (notably, attributes and operations of classes are not shown [11]).

Each notion (namely, *Stratification*, *RoME*, *Maskable Entity*, *Elements of Maskable Entity*) presented in the previous section is implemented by a corresponding stereotyped class of the PAM, as shown in Figure 2. When instantiating the PAM, the multiplicities attached with the relations warranties that the created instances comply to the rules of a well-formed stratification. This way, an instance of the class "*Maskable Entity*" is, at least, composed of two instances of the class "*Element of Maskable Entity*". An instance of the class "*Stratification*" is represented 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.

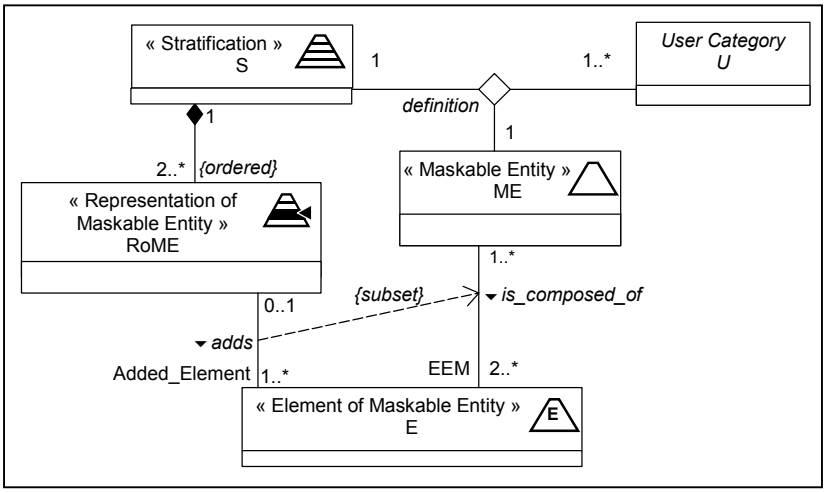


Figure 3. The Progressive Access Model described in UML

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 has to be redefined in order to maintain some useful information about users with regard to the WIS objectives. This is the case, for example, in the User Model we present in section 3.4. Here, this class allows us to introduce the adaptation features of the progressive access approach. The association "definition" allows the designer to specify as many stratifications as needed for a given Maskable Entity. This way, each user can benefit from a personalized progressive access. The multiplicities defined for this association guarantee that:

– only one instance of "Stratification" S is defined for one couple of instances (U, ME),

– only one instance of "Maskable Entity" ME is associated with one couple of instances (U, S),

– at least one instance of "User Category" is defined for one couple of instances (S,ME) (*i.e.* several users can share the same stratification defined for a Maskable Entity).

2.2.2 An application of the PAM to a UML Class Diagram

The description of the PAM we have presented is generic enough to match any formalism used to describe the underlying data model of the application. It is important to notice that the progressive access approach we propose can apply to a non-structured data model, as well as to a semi-structured (XML) or a structured (object or relational) one. For instance, in [10] we showed how to couple the PAM with an object-based knowledge representation system.

In this paper, we consider that the data model of the WIS is expressed using an object-oriented formalism and, more precisely, that it is described by a UML class diagram. In order to apply the PAM to such a data model one needs first to identify what are the potential Maskable Entities (ME) that can be stratified and then to specify, for each identified ME, what are the corresponding RoME. Basically, this two step process consists in producing models similar to those presented in Figure 4. Each model of the Figure 3 is a specialization of the PAM presented in Figure 2 and describes the extensional or intensional stratifications which can be performed on a particular type of Maskable Entity. Here, we focus on three kinds of ME which can be distinguished on a UML schema:

– the extension of the class diagram itself, seen as a set of classes, associations and classes-associations, can be stratified extensionally by creating RoME from this set of components (see Figure 4a). Therefore, stratifying the data model consists in defining different levels of detail at which the whole *population* (*i.e.* all the instances/tuples which constitute the extensions of all the classes/associations) is more or less completely presented to users.

– the extension of a class (or an association, or a class-association) seen as a set of instances (or/and tuples) can be stratified extensionally. In that case, the population linked to each class (association or class-association) can be more or less completely presented to users. The Figure 4b shows how the extension of a class is stratified by defining RoME from the set of instances attached to this class.

– The intension of a class (or an association or a class-association) seen as a structured set of variables (or/and roles) can be intensionally stratified. This kind of stratification allows to present the population of a class (respectively an association or a class-association) using a more or less detailed sub-structure of the class (respectively association or class-association). This consists in showing a more or less great number of variables (or/and roles) when displaying the content of each instance (respectively tuples). As shown in Figure 4c, RoME are made of class variables. This illustrates the case of a class to be intensionally stratified.

In the next subsection, we illustrate the instantiation of each of the three models presented in Figure 4 through an example.

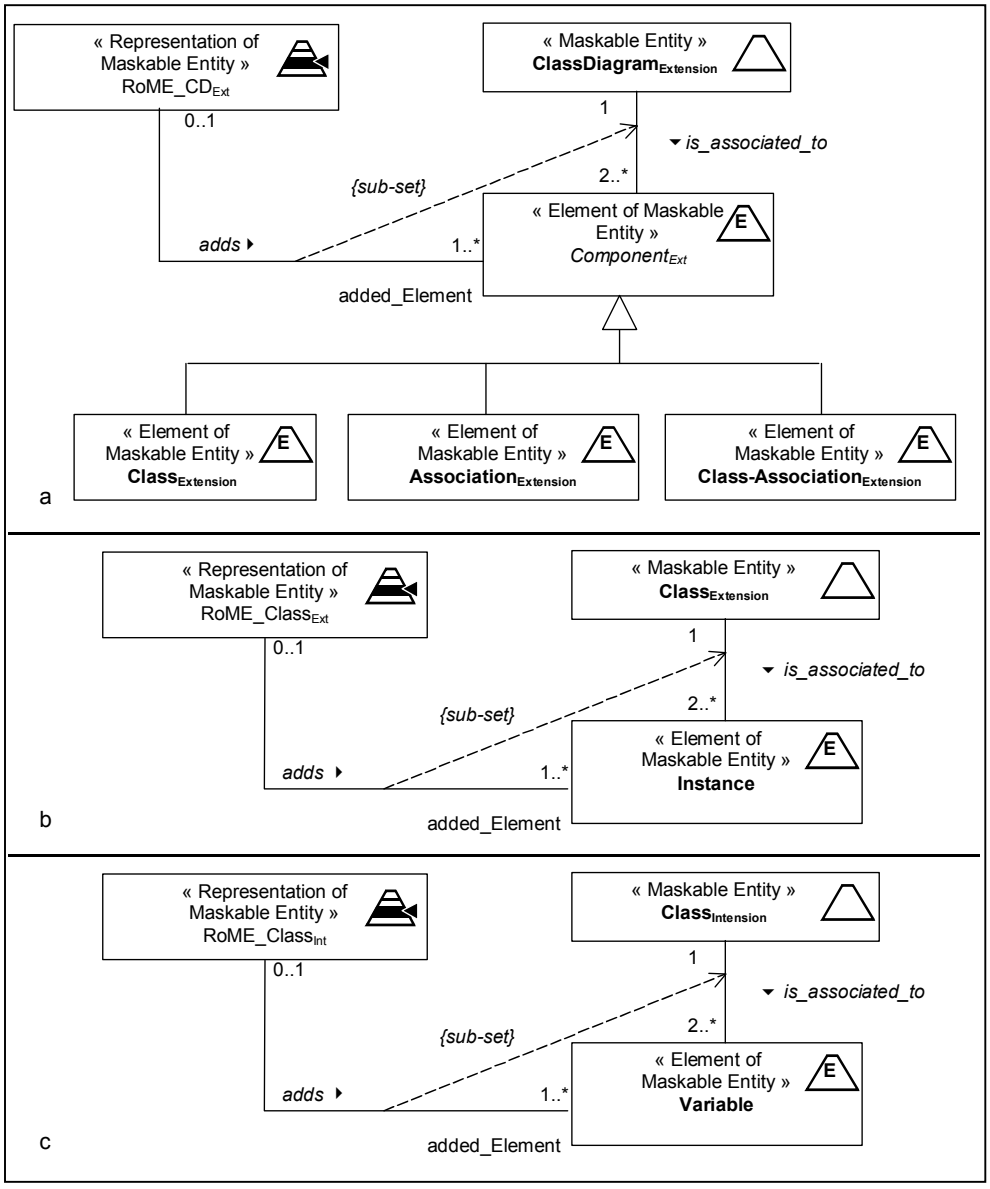


Figure 4. Three potential ME and their corresponding RoME for a UML class diagram. (a) the Data Model considered as a ME extensionally stratified – (b) a Class of the Data Model considered as a ME extensionally stratified – (c) a Class of the Data Model considered as a ME intensionally stratified.

2.3 Examples of stratification

2.3.1 The Data Model

The Data Model we present here is rather simple but should be sufficient to illustrate our approach. This model is an excerpt of a larger model used in a Geographic and Historical Information System managing data on river flooding we have developed in the SPHERE project [13]. The part of the

model we have kept for this paper is in charge of the management of *reports* written by *experts* (*geographers* or *historians*) who analyze *documents* about *flooding* events (see Figure 5).

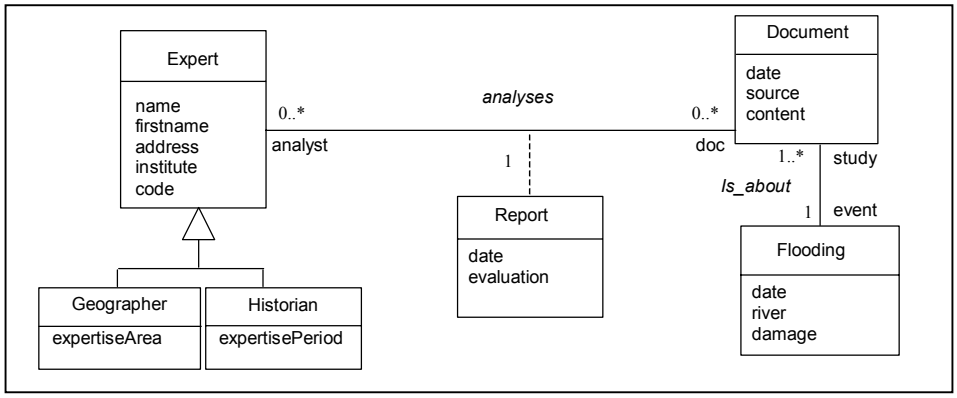


Figure 5. The Data Model used for illustrating some stratification possibilities

2.3.2 Extensional stratification of a class diagram

The Figure 6 illustrates an extensional stratification that is performed on the Data Model presented in Figure 5. This stratification has three levels of detail (three RoME), each being represented by a grey box. At the lowest level of detail (RoME 1), only the extensions of the classes *Expert* and *Document* (*i.e.* instances of these classes) and the extension of the association *analyses* (*i.e.* tuples of this association) are accessible. At the intermediate level (made of the elements found in RoME 2 + RoME 1), the extension of the class-association *Report* becomes available too. Finally, the highest level of detail gives access to the instances of the *Flooding* class and the tuples of the association *is_about* that gather instances of the *Document* and *Flooding* classes. Therefore, the lower (respectively higher) the level of detail chosen by the user is, the more the available information space is reduced (respectively expanded).

This example also illustrates how the progressive access approach can be used for confidentiality matters. For instance, in the stratification presented in the Figure 6, the two sub-classes of the class *Expert*, *Geographer* and *Historian* (see the whole Data Model of Figure 5) remain masked. Even at the highest level of detail, the user can *not* access to the information which is specifically related to the extensions of these classes. Moreover, the example shows that all the classes of a class hierarchy may not all belong to the same level of detail (or RoME). In other words, the class specialization is ‘virtually' cut by the stratification. In [11], we have proposed some rules of visibility for the stratification of an object-based schema. Basically, in the presented example, two different rules might have been specified for determining the extension to be shown:

- Rule 1: the instances of the sub-classes Geographer and Historian are never shown, and consequently neither are all the tuples involving instances of these classes. The only information which is accessible and stratified is related to instances of the Expert class which are neither classified as Geographer nor Historian.
- Rule 2: the instances of the sub-classes Geographer and Historian are shown as if they were instances of the class Expert. Here, the structure of the super-class Expert is used to present the whole population attached to the class hierarchy: the variables defined in these sub-classes are omitted when displaying their instances.

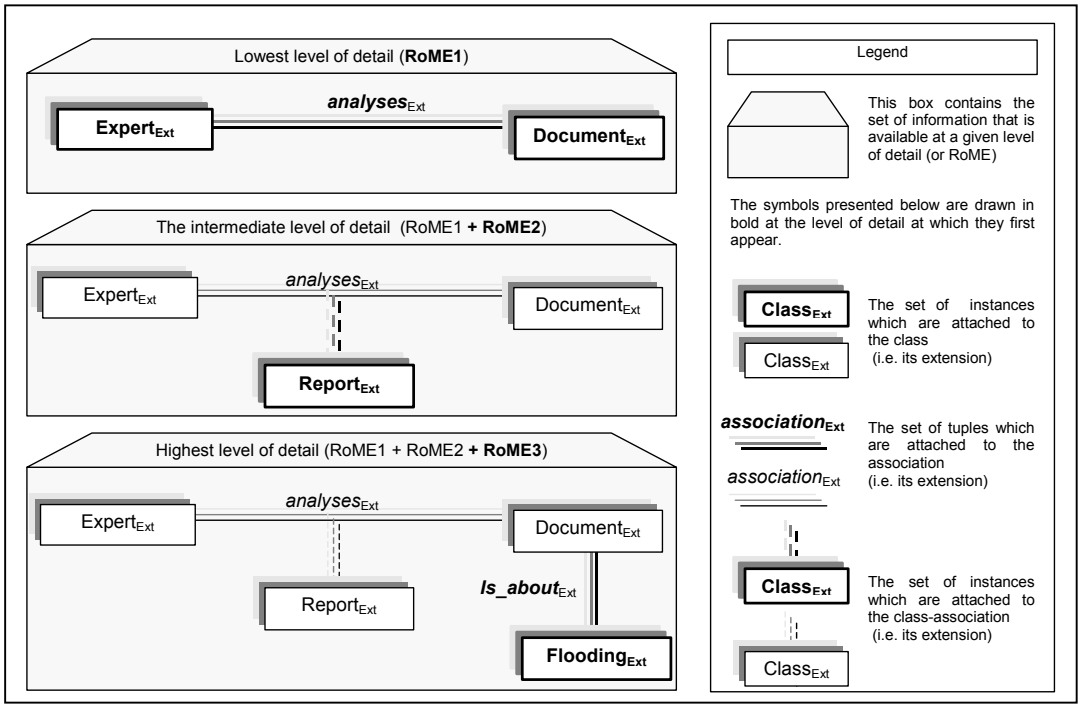


Figure 6. Example of a three-level extensional stratification described for the Data Model

2.3.3 Extensional stratification of a class

We give here an example of an extensional stratification performed for a class. This corresponds to a partition of the set of instances of the class. Through such a stratification, a progressive access to the set of instances of the class can be provided. Considering the class *Geographer*, we propose to stratify its population using the value of the area of expertise as a criterion for building RoME. This information is given by the variable *expertiseArea* of the class (see Figure 5). Let us suppose that at the first level of detail, only the geographers who are experts of a given river are presented. At a second level, the instances of the class for which the variable *expertiseArea* equals the corresponding river basin are visible. The third level finally adds to the information space the rest of the population, i.e. the geographers who are experts of any other area.

2.3.4 Intensional stratification of a class

Let us now consider the class *Historian* described by the following variables: *name*, *firstname*, *address*, *institute, code* and *expertisePeriod*. The Figure 7 shows how the intension of this class, considered as a ME, is stratified for two distinct users, User1 and User2. As described in section 2.2.2, the intensional stratification relies on the definition of RoME from the set of variables which compose the structure of the class.

At the first level of detail, User1 is given a representation of the class using only variables *name* and *firstname* ($RoME_1$ contains only these two variables). For all instances of Historian that are shown, User1 gets these two pieces of information. When querying the content of such an instance, now if she/he 'unmasks'[1] the stratification, she/he is given access to the *institute* and *code* information

[1] This illustrates one of the two specific navigation mechanisms proposed for the progressive access in section 2.1.1.

which correspond to the two variables added by $RoME_2$. Note that the stratification defined for User1 presents only two levels of detail and that two variables (*address* and *expertisePeriod*) are kept masked for this user. The second stratification described for User2 is totally different, because User2 needs and rights are different from User1's ones.

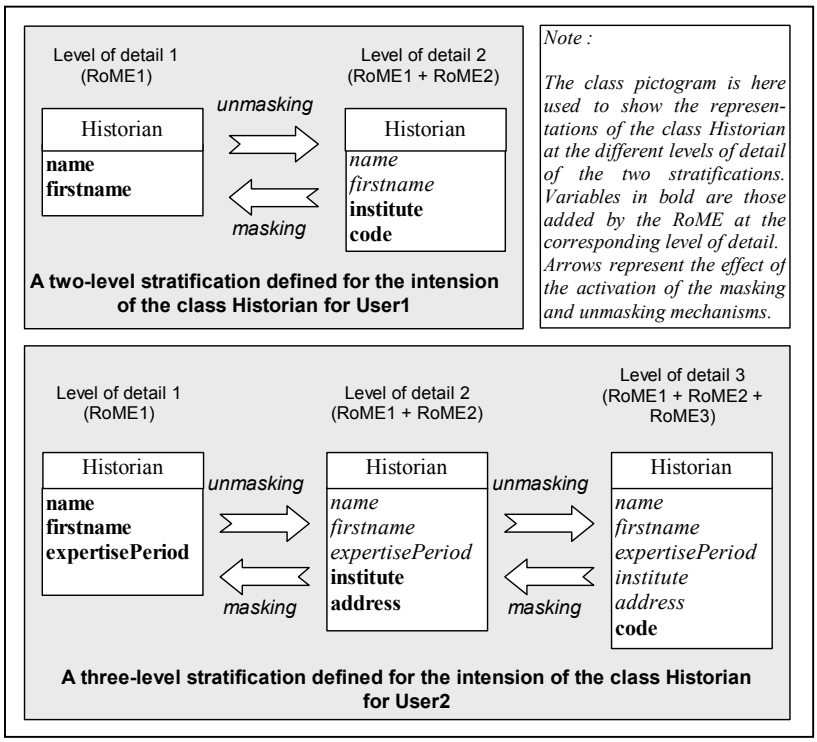


Figure 7. Examples of two stratifications of the intension of the same class defined for two distinct users.

3. Four complementary models

The PAM presented in the previous section can be seen as the central model of a five model architecture that allows to set-up a progressive access to both the information space and the functionalities of a WIS. First, the data of the domain of application targeted by the WIS are described by a *Data Model*. Also, we consider that the set of services or functionalities offered by the WIS are described through a model called the *Functional Model*. Since our objective is to personalized the progressive access for possibly each user, a *User Model* is also needed in order to characterized the different groups of users. Thus, it is possible to associate one stratification with a particular user or group of users. Finally, considering that information and functionalities provided by the WIS by means of Web pages which are generated dynamically and accessible through a Web browser, a *Hypermedia Model* is required for the description of both the composition and the appearance of these Web pages, but also for the specification of the navigation features. Consequently, we envision the design of WIS as a multi-faceted activity which relies on the five models listed above. In this section, we introduce the four complementary models of the PAM used in our design approach: the *Data Model*, the *Functional Model*, the *User Model*, and the *Hypermedia Model*. We present each of these four models, then we describe the links between each of them and the PAM.

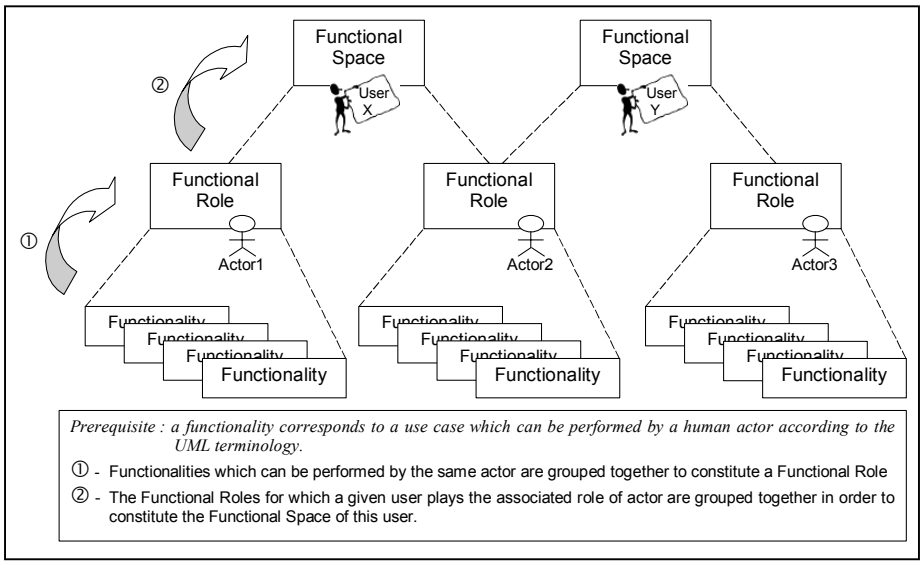
3.1 The Data Model

The *Data Model* describes the objects of the real world which constitute the domain of application targeted by the WIS. This model is conceptual and puts the emphasis both on the concepts of the domain of application and on the relations which hold between these concepts. It is worth noting that the Data Model is described independently from any progressive access notion. We suppose here that this Data Model is expressed using an object-oriented formalism and, more precisely, that it is described by a UML class diagram (as for example in Figure 5). We stress that the design of the Data Model has to rely on an analysis of the user's needs which can be described using some UML use cases. Also, use cases are a good starting point to design the Functional Model.

3.2 The Functional Model

The Functional Model describes the tasks a user can perform with the WIS. The peculiarity of this model stands in the organization of the available tasks. This organization relies on three levels supported by concepts of different granularity: the *functionality*, the *functional role* and the *functional space* (see Figure 8).

Figure 8. The three levels of the functional dimension of a Progressive Access WIS.



The design of the Functional Model is driven by users needs. We suppose that a previous step of requirement analysis for the WIS has led to the description of use cases. At the lowest level of this three level structuration of the functional dimension of the WIS, the functionalities are described. Functionalities are identified from use cases where identified actors are persons using the system (as opposed to use cases where actors are, for instance, components of the application, or other systems, etc.). At an intermediate level, the functionalities defined for a same actor (for instance, Actor1 in Figure 8) are grouped together in a functional role. At the highest level, a functional space, defined for every user, gathers the various functional roles a user can play. A single user can fulfill the role of several actors. The functional pace of a given user includes all the functional roles she/he can play. For instance, User X owns a functional space which contains the functional roles defined for Actor1 and Actor2 (see Figure 8).

Such an organization of the functional dimension allows to provide each user with a personalized functional space. A functional space is composed of functional roles which are themselves composed of functionalities. Among the different kinds of functionalities, if we focus on the consultation functionalities, we can see them as the set of data they return as result of the corresponding query. Then, a functional space, a functional role and a (consultation) functionality can each be considered as a Maskable Entity and, consequently, be referenced by the PAM using the description presented in Figure 3. Then, a progressive access can be offered on the functional dimension of a WIS as explains below:

– considered as set of two or more functional roles, a functional space can be stratified. This way, a user can access first to the functional role she/he plays the more frequently. Then, using the unmasking mechanism, her/his functional space can be expanded and then give access to some additional functional roles. For instance, in Figure 8, considering User X's functional space, one can imagine that at the first level only the functional role corresponding to the Actor1 could be accessible. The functional role of the Actor2, also played by this user but less frequently, could be reached at a second level if needed.

– considered as a set of two or more functionalities, a functional role can be stratified at its turn. The principles are quite similar to those adopted for the stratification of a functional space. Here, the different levels of detail offer a progressive access to the functionalities attached to a precise functional role. The definition of the stratification could be based on various criteria, for instance giving priority to the last functionalities used.

– considering the set of two or more elements which compose the result of the query linked to a (consultation) functionality, a stratification can also apply. It is worth noting that the result of a query is a set of information extracted from the Data Model. This way, our approach allows to provide users with a progressive access to the information supported by the Data Model. Going back to the example of Information System introduced in section 2.3, let us consider a functionality which allows users to consult information about historians. We suppose that this functionality is based on a query on the Data Model (see Figure 5). The result of this query, expressed as 'SELECT * FROM Historian' in a SQL-like syntax, is the set of instances attached to the class Historian. Being extracted from the extension of a class, this result set can be extensionally and/or intensionally stratified. The example of intensional stratification given in section 2.3.3 for the class Historian illustrates the way the result of a query can be stratified.

In the next subsection, we present the Hypermedia model which describes the interface proposed to the users of the WIS when accessing their stratified functional space.

3.3 The Hypermedia Model

The *Hypermedia Model* is composed of two sub-models: a *Structuration and Navigation Model* and a *Presentation Model*. The former describes the structuration of the hypermedia and the paths of navigation according to the progressive access. The later is dedicated to the composition and the appearance of the Web pages.

3.3.1 The Structuration and Navigation Model

The structure of the hypermedia (the visible part of the WIS) is derived from the *Functional Model* by the *Structuration and Navigation Model*. This structure is composed of *contexts* and *nodes* as illustrated by the legend of Figure 9. Contexts correspond to the three notions of the Functional Model

which can be seen as ME and possibly stratified: *functional space*, *functional role*, and *functionality*. A context is composed of three kinds of nodes: *stratification nodes, RoME nodes* and *element nodes*. Navigation in the WIS can be performed either *within* a context, using the masking and unmasking mechanisms, but also between two contexts provided that the source context is both an element of a RoME and a ME. As shown by Figure 9, the Structuration and Navigation Model derives from the Functional Model and the navigation paths are based on the logic of the progressive access.

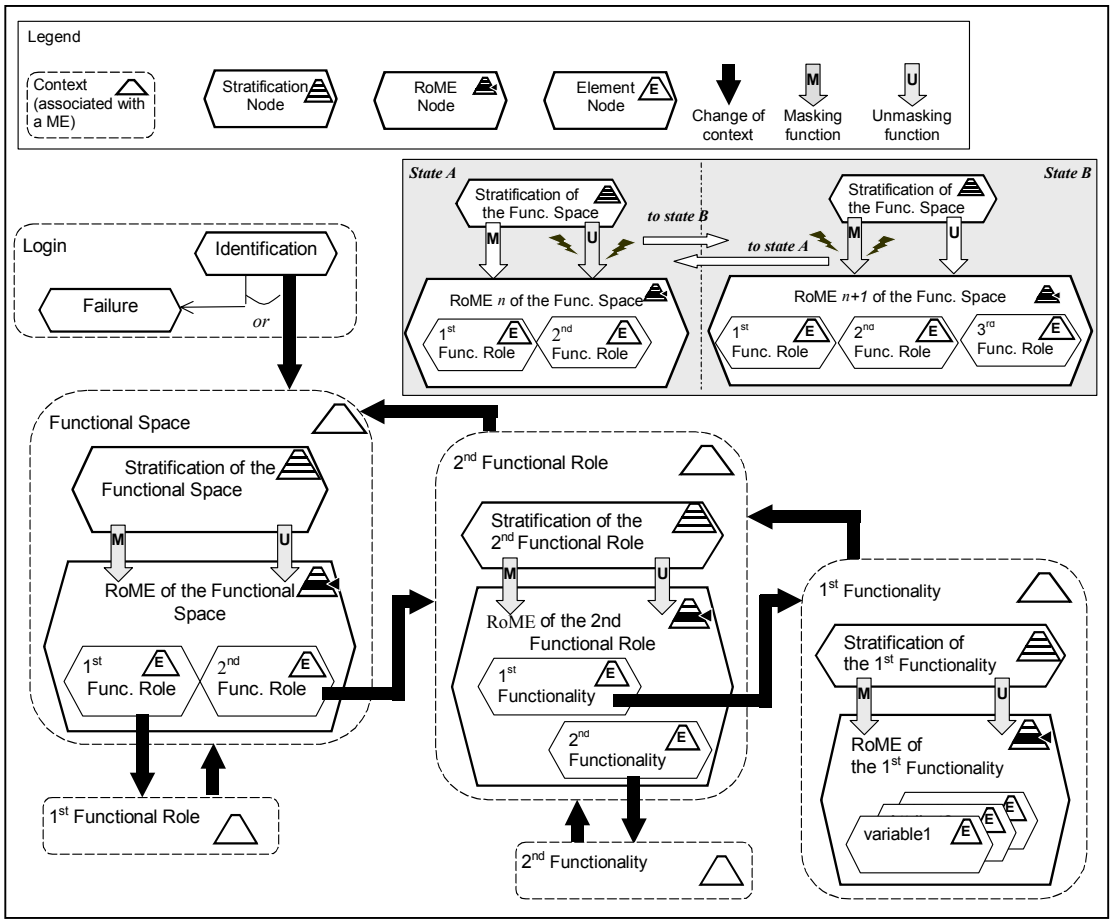


Figure 9. The Model of Structuration and Navigation

Seen as a ME, a functional space is stratified into RoME that are sets of functional roles. Also, a functional role can be stratified into RoME that are sets of functionalities. Finally, a (consultation) functionality can be stratified into RoME that are sets of class variables involved in the result of the query associated with this functionality. On Figure 9, gray arrows correspond to masking (M) and unmasking (U) navigation links within a context. On the right upper side of the figure, the effect of the activation of these links in the context of a functional space is shown. Starting from a state A corresponding to a level *n* in the stratification of the functional space, the activation of the unmasking function leads to a state B. This state B corresponds to the level *n+1* in the stratification of the functional space. This way, the user accesses progressively to the different functional roles she/he can

fill. Conversely, when activating the masking function from state B, the state A is reached giving access to a smaller number of functional roles.

Black arrows symbolize a change of context. Once the user has chosen a functional role, she/he is given access to the set of functionalities associated with this role. Again, she/he can choose the target functionality by navigating progressively within this set. Then, when the target functionality is reached, she/he can use the masking/unmasking mechanisms to get more or less detail in the result displayed by the activation of the functionality. Besides these links derived from the structuration of the functional space, it is also possible to define additional contexts and nodes (see, for instance, the context Login in Figure 9) or to link two contexts which correspond to elements of the same RoME, for instance two functionalities often used consecutively.

3.3.2 The Presentation Model

The Presentation Model is based on the definition of both a *composition charter* and a *graphical charter*. The composition charter describes a set of Web pages derived from the Structuration and Navigation Model. Each page has at least one region called *body* and possibly other regions containing a title, a foot page, a tool bar, or a navigation bar. The body of a page is composed of *frames*. A frame owns some *components*. The content of a component can be the result of a functionality but also some scripts (cgi, applet…) to be executed. In order to provide users with Web pages that take into account the progressive access, a default composition charter is proposed. It is composed of three *page templates* (see Figure 10), each corresponding to a context. At the top of Figure 10, a page template for a functional space is presented. A page template for a functional role stands in the middle of Figure 10, and at the bottom of Figure 10, is located the page template which corresponds to a functionality.

The body of a page template has two frames, one for the stratification, the other for the RoME. The stratification frame gives some information about the stratification, and contains two ‘button’ components for activating the masking or unmasking mechanisms. The RoME frame displays information about the RoME which corresponds to the current level of detail reached by clicking on the masking and unmasking buttons. This frame encompasses sub-frames, each of them displays each element of the current RoME. By default, the RoME with the lowest level of detail is presented. A frame associated with an element of RoME which can also be seen as a ME, contains a component which allows to pass from a context to another. The navigation bar helps in localizing the current context in the hierarchy of contexts of a functional space. The tool bar allows the user to quit the session or accede to her/his profile. The graphical charter defines the look of each Web page of the WIS, giving some spatial layout rules and some visual characteristics. In order to maintain a certain homogeneity in the visual appearance of the WIS Web pages, we provide by default three graphical charters, each being associated with a page of the composition charter. In the example treated in section 5, Figure 18 and Figure 19 present two different templates and graphical charters for a same functionality.

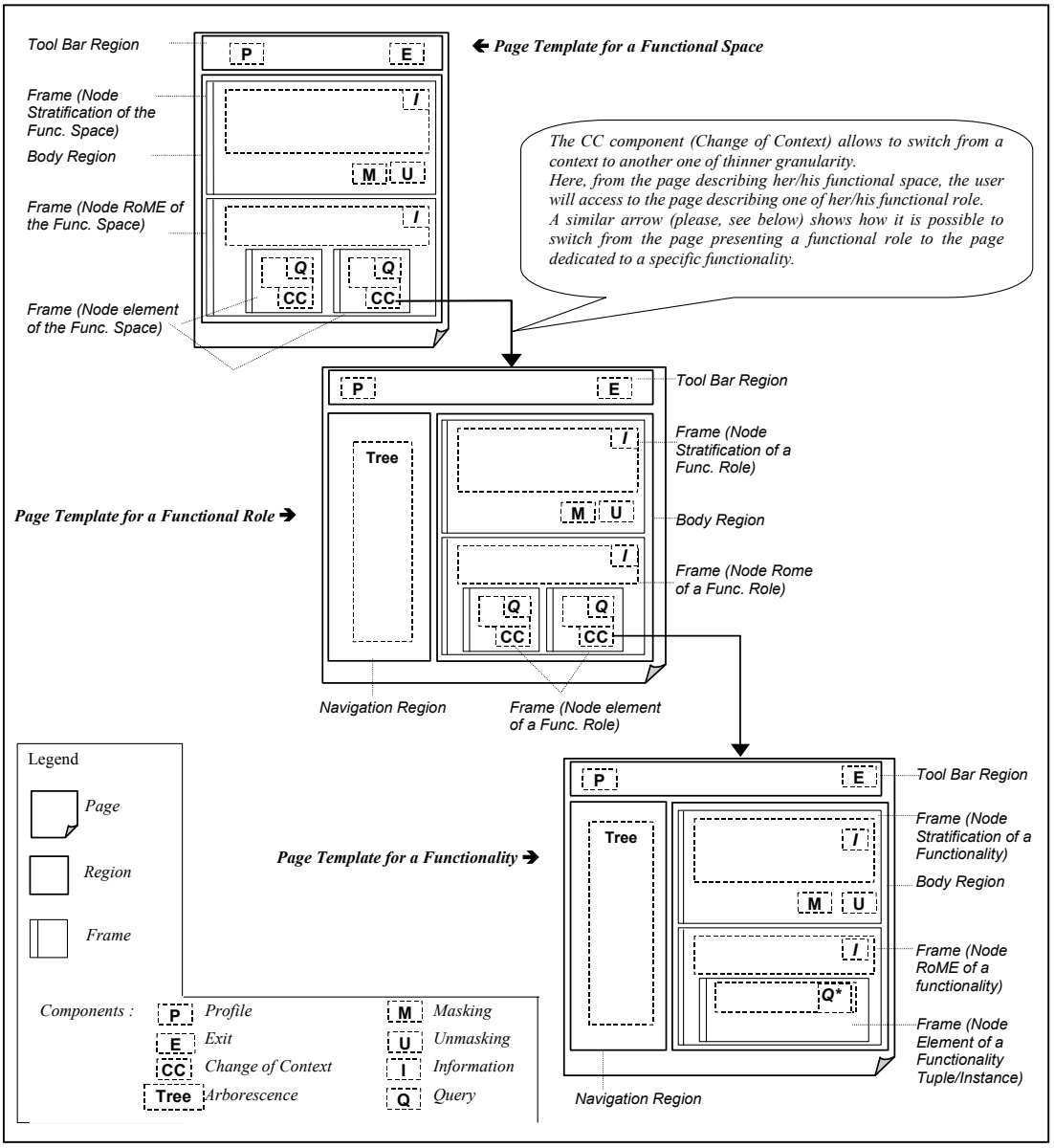


Figure 10. Page Templates of the composition charter

3.4 The User Model

In a UML-based design approach, the user is considered as an actor, an external entity who interacts with the system but who has no internal representation. When adaptation to users is concerned, the system needs to store some information about users. This is the reason why some Web design methods propose an internal representation of the users [14][15][16]. We also adopt this approach.

We introduce a distinction between groups and individual users which is justified by the functional approach as traditionally encountered in the field of Information Systems. More precisely, the User Model is closely related to the Functional Model as illustrated on the Figure 11. We associate one

Group to exactly one *Functional Role*. Therefore, all the *Users* who belong to a *Group* do play the same Functional Role. A user belongs to one or more groups, as she/he can play one or several functional role(s). A user also owns a personal *Functional Space*, which is defined as a set of functional roles fulfilled by her/him. It is possible to deduce the composition of a given functional space from the list of the groups to which belongs the user. Also, when the functional roles a user fulfills are known, the set of functionalities she/he can perform can be obtained.

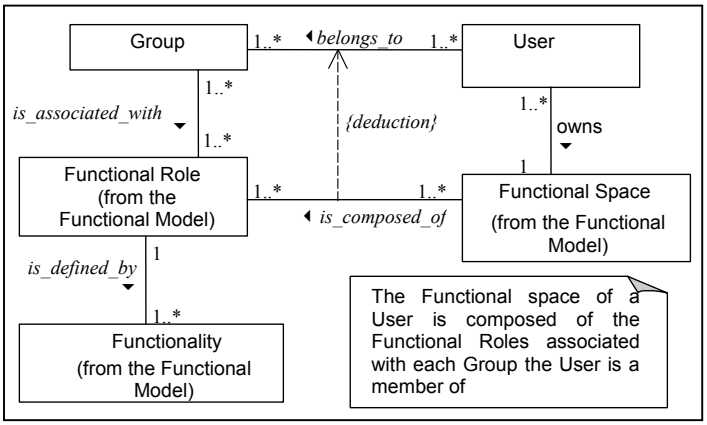


Figure 11. The functional Model and its links with the User Model

In the *User Model*, we propose the class *Group* (respectively *User*) describes the profiles of groups of users (respectively individual users). It can be noticed that these two classes are modeled as sub-classes of the class *User_Category* introduced by the PAM (see Figure 3). Then, the User Model can be seen as an extension of the PAM. Such an extension can be specific to the application domain of the WIS. Notably, this model allows designers to integrate some characteristics for representing users which are independent from any progressive access feature but domain dependant instead (for instance, a level of expertise towards some concepts of the application domain). In this paper, we present a more specifically progressive access oriented User Model.

Beside the traditional variables which describe the name and the description of a group, but also the identification and authentication features for a user, three profiles are considered as sub-classes of the class *Group* (respectively *User*):

– the *G_ProgressiveAccess* (resp. *U_ProgressiveAccess*) class which contains information about the stratifications established for this group;

– the *G_Preferences* (resp. *U_Preferences*) class which describes the preferences of the group (resp. user) in terms of the presentation of the hypermedia;

– the *G_Behavior* (resp. *U_Behavior*) class which describes the behavior of the group (resp. user) within the WIS and whose information can be used for dynamic adaptation of both the content and the presentation.

Each of these three classes represents a profile used for adaptation. The progressive access profile is exploited by the Functional Model and, by derivation, by the Structuration and Navigation Model, in order to build the hypermedia structure of the pages presented by the WIS. The preferences are exploited by the Hypermedia Model for building the personalized pages from the graphical charter associated with the group or the user. The behavior profile is first exploited to store the actions

performed by the group or the user on the WIS. Such information can then be used by the Progressive Access Model in order to reorganize the stratifications associated with the group or the user, using primitives described in [11]. This way the WIS is able to adapt itself dynamically to its users.

3.5 The links between the Models

The four models presented in this section have been defined for integrating the principles of the progressive access into each of the dimensions (content, functionalities, hypermedia features) of a WIS. In order to reach this goal, we widely exploit the possibilities offered by the PAM, the reference model of our approach. The Figure 12 shows the relations that hold between this model and the four other ones. We now describe these relations.

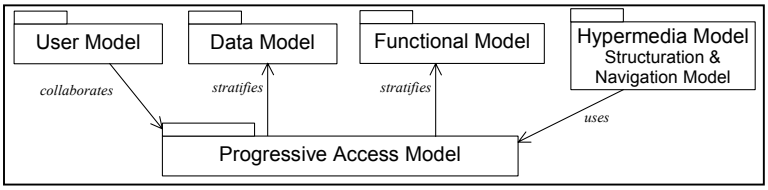


Figure 12. Links between the four design models and the PAM

In section 2.2.2, we have mentioned that an object data model together with its components (classes, associations and classes-associations) can be considered as Maskable Entities and, therefore, can be stratified. In our design approach, the PAM is exploited by the Data Model in a similar way. The Data Model can remain partially stratified since only information involved in some functionalities actually needs to. Indeed, the concept of *query result* modeled in the Functional Model is a set of elements obtained from the Data Model. Seen as a Maskable Entity, this result is stratified into different RoME. In that case, the application of the PAM to the Data Model is performed under the control of the Functional Model.

The concepts and the principles of the PAM also directly apply to the Functional Model. We have shown how the concept of *Functional Space*, considered as a ME, can be stratified into different RoME offering the progressive access to the functional roles which compose this Functional Space. At its turn, each functional role is seen as a ME. The stratification of such a role offers a progressive access to the functionalities it encompasses. Finally, the principles of the progressive access apply to the result of each functionality.

The Hypermedia Model maintains another kind of relation with the PAM. Its Model of Structuration and Navigation *uses* the concepts described by the PAM. This model is a first step towards the implementation of the logic of progressive access into a hypermedia.

Finally, the relation between the PAM and the User Model can be seen as a collaboration. On the one hand, the descriptive capacities of the User Model are enriched by the integration of the concepts and the principles of the PAM. It is possible to extend a user profile by taking into account her/his expectations regarding the content and the presentation of the delivered information. On the other hand, the PAM is given a supplementary dimension of adaptation. The User Model allows to customize every stratification. This increases the expressive power of the PAM.

3.6 Overview of the Design Methodology

The dependency relationships which hold between the five models of our proposition partially determine the methodological guidelines which are presented in this section. Such dependency relationships are more precisely described in [11]. Dependencies result from the fact that, basically, the instantiation of a model requires that the instantiation of some others has been previously achieved.

Upstream from the design process, an analysis of the functional requirements is recommended as a preliminary step. Based on the description of use cases, the requirement analysis helps in identifying the essential information for specifying the Data Model and the Functional Model. At the end of the chain, the generation of the SIW terminates the process. Between these two stages, seven design steps take place:

- Step 1: Definition of the Data Model. It consists in creating the classes and the associations which represent the application domain.
- Step 2: Definition of the Functional Model. This step requires the creation of the Functionalities of the WIS which have been identified during the requirements analysis and are described by use cases. A functionality is given a name and a description of the task it performs. This step also includes the specification of the query on the Data Model which corresponds to the functionality. Functional Roles are then described (by a name, a description) and each of them is linked to its set of functionalities.
- Step 3: Definition of the User Model (1/3). The definition of the User Model is a particular step which is divided into 3 sub-steps. In this first sub-step, the groups are identified and each of them is associated with a functional role. The individual users identified by the designer are also created and declared as members of the adequate group(s).
- Step 4: Identification of the Progressive Access Requirements. In order to ease the description of the stratifications, we propose some specific notations allowing the expression of the users requirements in terms of progressive access (*i.e.* what are the information or functionalities they want to access at the first level of detail, at the second, etc.). These notations are based on use cases and on UML sequence diagrams. They are described together with some specific directives for their use in [11].
- Step 5: Definition of the User Model (2/3). The progressive access profiles are defined for groups and users. Stratifications for functional roles and functionalities are created using the specifications of step 4. These stratifications are referenced by each profile. Stratifications relative to functional spaces of known users are defined. Stratifications concerning functional roles and/or functionalities that the designer needs to adapt to a user are specifically redefined.
- Step 6: Definition of the Hypermedia Model. This step begins with the derivation of the Model of Structuration and Navigation (one per group). It gives the skeleton of the hypermedia which will be proposed for each group of users. Every model is translated into a Presentation Model by applying either the composition charter provided by default and presented in the section 3.3.2 or another charter created by the designer. The definition of the Hypermedia Model continues with the definition of the appearance of each page template. Again, the designer can either link them to the default graphical charter, or specify a new graphical charter.
- Step 7: Definition of the User Model (3/3). This third sub-step consists in updating the preferences profiles. The designer references in each group's profile and, if needed, in each user's profile, the chosen composition charter and the chosen graphical charter.

These seven steps form a methodology for designing WIS adopting a progressive access approach. One of its benefits is that it indicates in which order the five models presented before are to be instantiated.

We have implemented the five models in a platform, called KIWIS. This platform is dedicated to the design and the generation of WIS and implements the principles of the progressive access. KIWIS assists the WIS designer by decomposing the design process into the seven steps presented above. Once these steps are achieved, KIWIS, as a Web server, constitutes the executive environment of the generated WIS. The KIWIS system is described in the next section.

4. The KIWIS System

4.1 Software Architecture

KIWIS is a platform for designing and generating adaptable WIS which implements the progressive access approach. KIWIS is written in Java. The architecture of KIWIS (see Figure 13) relies on freeware tools and on five packages of Java classes, called *Managers*, which form the kernel of KIWIS. KIWIS uses the AROM system [17], an object-based knowledge representation system which adopts UML class diagram-like notations. AROM is used to describe the application domain of the WIS. AROM knowledge bases are translated in XML data files. KIWIS is itself a WIS relying on the http server Apache and the servlet engine Tomcat. Dynamic web pages appearance is obtained by assembling XML and XSL files using the publishing framework Cocoon. Queries associated with the functionalities are sent to the Kweelt XML query module.

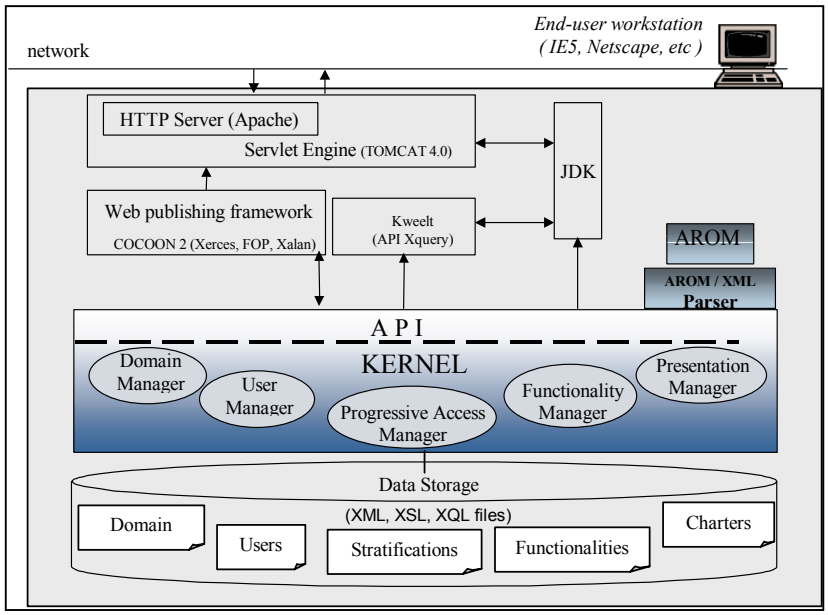


Figure 13. The architecture of the KIWIS platform.

We present now the five modules of the KIWIS kernel (see Figure 13), each managing one of the five models presented above:

- The Domain Manager manages the data of the Domain Model (in XML format). It communicates with the AROM system which describes the application domain of the generated WIS. Using a XML/AROM parser, data of the application are translated from a AROM knowledge base into XML files.
- The User Manager manages the User Model. It controls the creation of new users, the deletion or modification of their profiles. It also performs an authentication when a user logs on the generated WIS. The User Model is stored as two XSD (XML schema) files which correspond to the Group and User classes. Instances of theses classes are profiles described by XML files.
- The Progressive Access Manager manages the Progressive Access Model. It performs the creation, modification or deletion of any stratification defined for a user (group or individual) on the XML data of the application domain. Stratifications are described by a XSD file and instantiated as XML files.
- The Functionality Manager manages the Functional Model. It is in charge of creating, modifying or deleting the functional spaces, functional roles and functionalities of each generated WIS. The Functional Model is expressed using a XML schema file. Functionalities are XML files validated by this XSD file.
- The Presentation Manager manages the Hypermedia Model and the composition and graphical charters in the XSL format.

4.2 Functional Architecture

We now describe the communication between the managers in the processing of a functionality activated in a WIS designed and generated with KIWIS. We detail the different steps indicated by the numbers in the Figure 14.

The call to a functionality is first transmitted to the Functionality Manager (step ①). This manager sends a message to the User Manager concerning the query (step ②). The User Manager consults the Users Database and identifies the Progressive Access Profile to be applied (*i.e.* an individual one or, by default, a group Profile). This Profile is then sent to the Progressive Access Manager (step ③) which searches for the corresponding stratification in the Stratifications Database. This information is sent back to the Functionality Manager (step ④) which builds the result in terms of stratified content by obtaining the reply to the query from the Domain Manager (step ⑤). In parallel, the Functionality Manager has notified the Presentation Manager that a functionality is requested (step ②b). The Presentation Manager looks for the page template to be applied to the result, by asking the User Manager which charters to be used (step ③b). Then, the Functionality Manager transmits to the Web page publishing framework Cocoon an XML file which contains the reply to the query (step ⑥) (in terms of stratified content). The Presentation Manager sends the XSL file to be applied (step ⑥b). Cocoon assembles these XML and XSL files and builds the result Web page (step ⑦). This result Web page displays the data according to the first level of detail defined for this functionality and for this user and according to the composition charter and the graphical charter given by the Profile of this user.

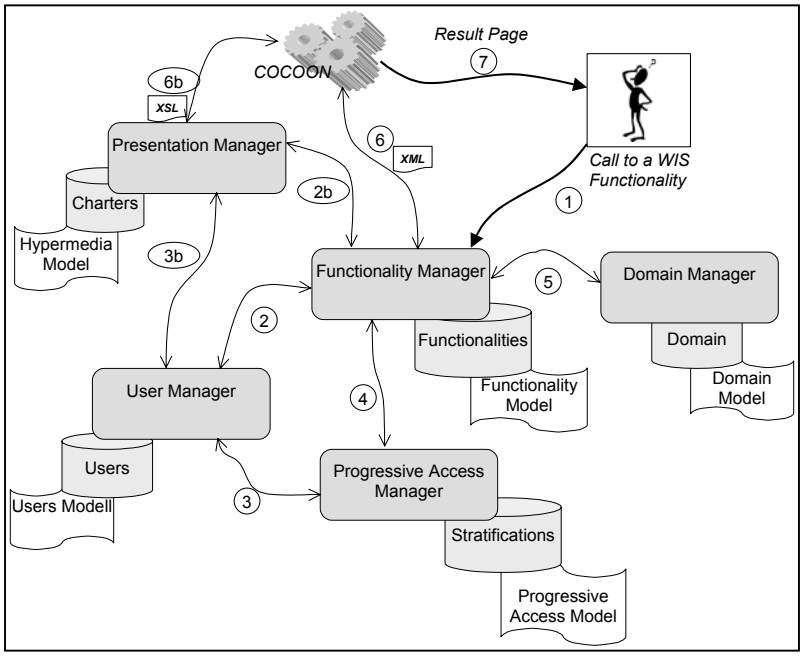


Figure 14. Communications between the managers in the processing of a functionality in a WIS generated with KIWIS

From this displayed result Web page, the user may, if possible, unmask the stratification towards more information. The activation of the unmasking mechanism is performed by the Tomcat servlet engine. A servlet treats the request of unmasking by sending to Cocoon the new parameters for publishing a new page. This consists in re-applying the XSL style sheet on the XML file which now contains the data available at the greater level of detail. The same process is launched if the user wants to reach a lower level of detail by masking some information. In the next section, we illustrate such mechanisms through the presentation of a Web-based Information System (WIS) integrating a progressive access, designed and generated with KIWIS. This example shows the adaptive Web pages generated and displayed by the WIS during its use by a particular user. Screens which allow to design such a WIS following the design methodology described in section 3.6 are not presented here. For an illustration of the use of KIWIS in the design phase, please refer to [11].

5. Example: a WIS designed and generated with KIWIS

In this section, we present an example of a WIS developed in the SPHERE project [13] and whose Data Model has been introduced in section 2.3.1 (see Figure 5). This WIS (called FRISK-IS) is dedicated to the management of a corpus of documents about flooding risks. All the available documents registered in the WIS are evaluated by experts concerned by these natural hazards (Geographers, Historians in that simple case). To begin, a first functional role, called the '*Expert'* role, can be identified. This role is defined by a set of functionalities to be performed like checking the list of documents required for an analysis, downloading a document to be analyzed, submitting an analysis report, modifying an analysis report, consulting the analysis report of another expert, etc. A second functional role identified for this WIS is called the *'Manager'* role. Such a role essentially consists of the management of some information concerning experts. Some functionalities inherent to this role can be listed as adding a new expert, modifying information about an expert, getting the list of the reports in progress, finding an expert to analyze a document, etc.

Let us now suppose that a user of the WIS, called user U, has been identified as a member of the two groups (Expert and Manager). Figure 15 shows an excerpt of the Structuration & Navigation Model instantiated for User U according to the specifications of the progressive access required for this user. In this figure, three context are represented in order to illustrate how the user U can navigate in her stratified and personalized functional space to reach, from this point, a given stratified and personalized role and then to reach a given stratified and personalized functionality of this role. The effects of activating the masking and unmasking mechanisms are also presented.

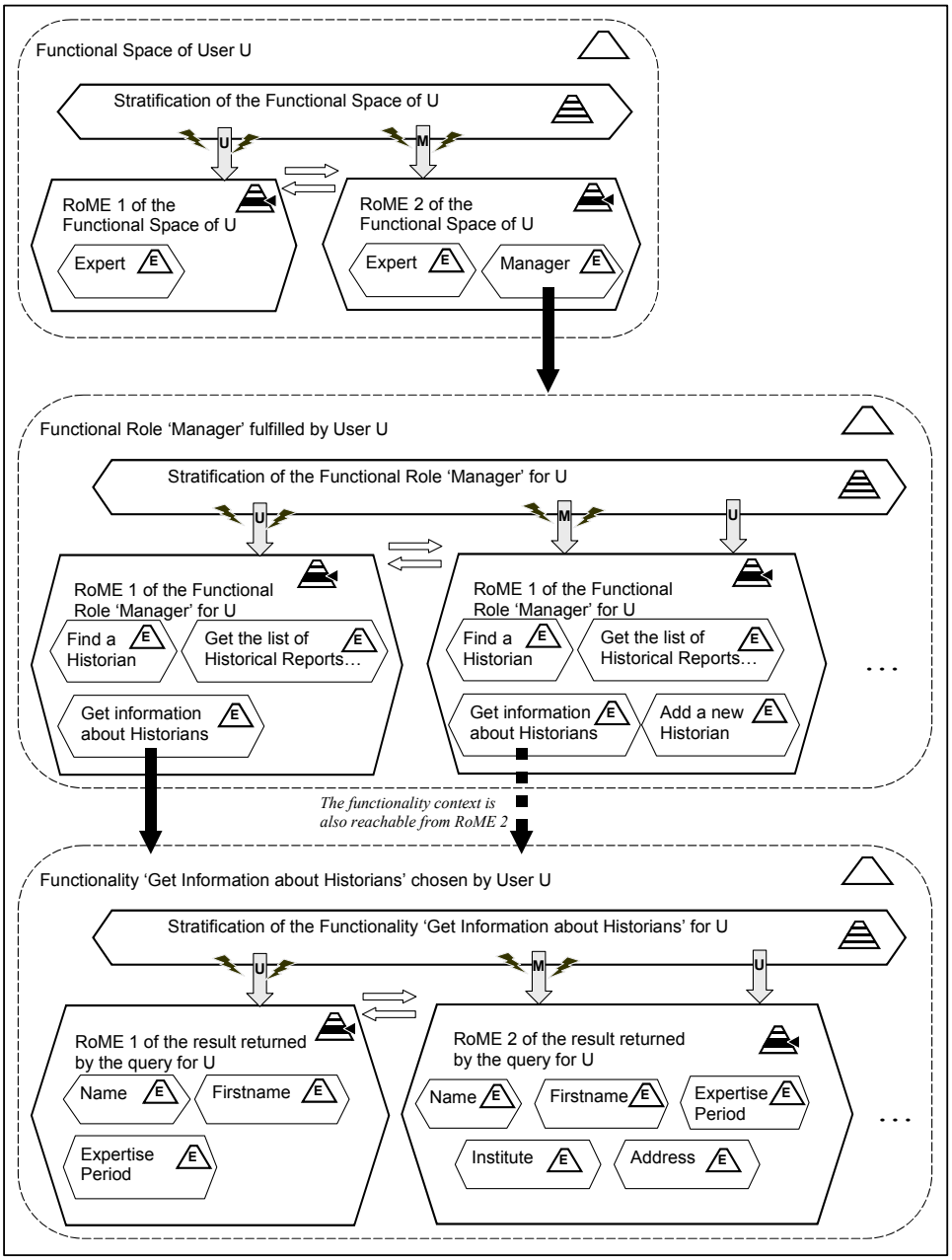


Figure 15. An excerpt of the Structuration & Navigation Model instantiated for User U according to the stratifications defined for her concerning the functional dimension of the WIS

In the remainder of the section, we present in a more detailed way the path followed by user U. Explanations we give are illustrated by some screenshots of the FRISK-Information System.

Figure 16 shows one of the first Web page displayed by the WIS for user U when she enters the FRISK-IS. This page informs the user U about her functional space. In this WIS, roles of the functional space have been grouped in RoME and ordered by frequency of use. In that case, user U accesses to the role Expert at the first level of detail available for her functional space. User U can, if needed, access to some other roles she is authorized to performed on the FRISK-IS, by clicking on the unmasking button. At the bottom of the Figure 16 the effect of the activation of the unmasking function is shown. User U can now access to the functionalities encompassed by the role of Manager. This role is actually less frequently fulfilled by user U, this is why it appears at the second level of detail.

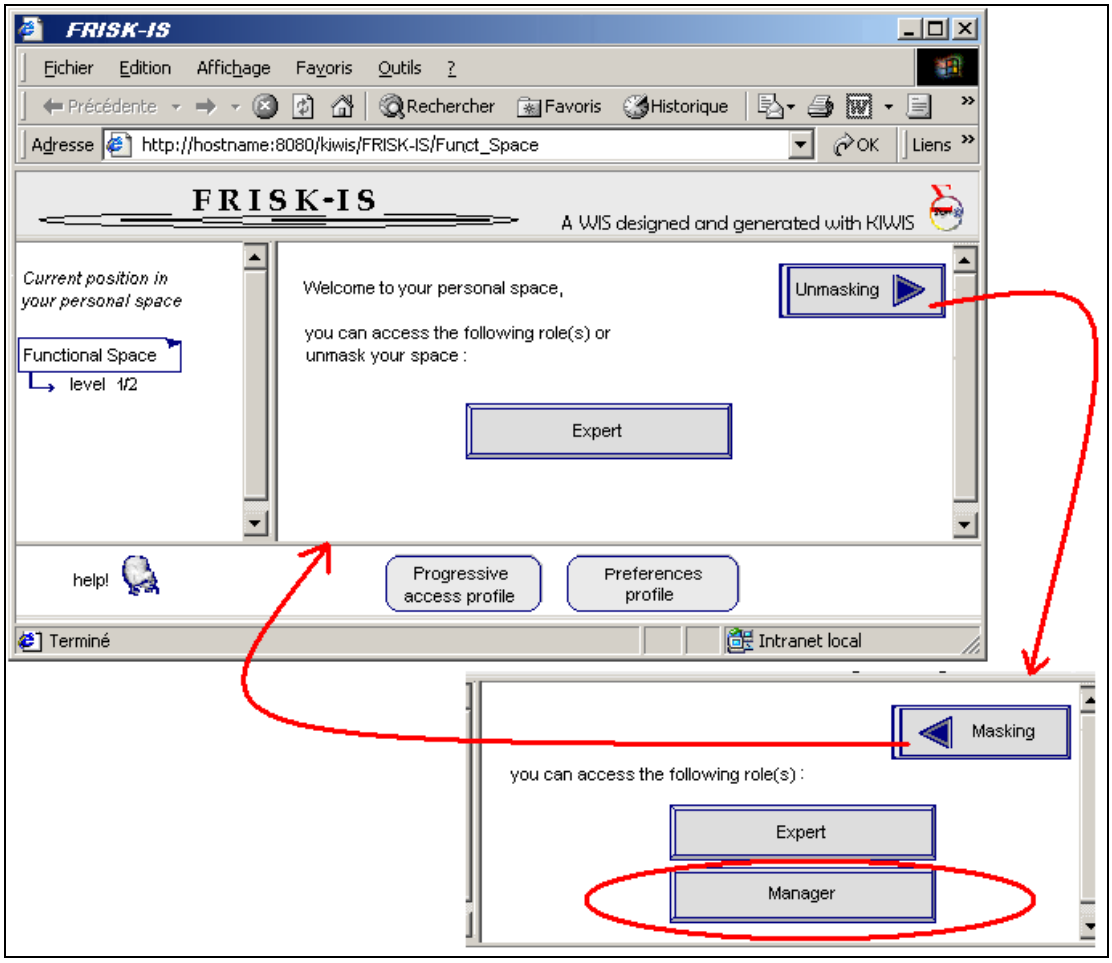


Figure 16. The Web pages concerning the personal functional space of a particular user.

Let us now suppose that the user U fulfills the role of Manager of the Experts who evaluate documents related to flooding risks. Figure 17 shows the web page displayed by the WIS when user U clicks on the Manager button in the functional space page (see Figure 16). It illustrates the stratification of the functional role Manager and gives access to the set of functionalities attached to

the role Manager following the existing stratification. At the first level of detail, a click on the unmasking button unmasks an additional functionality for this role ('*Add a new Historian*') as shown at the bottom of the Figure 17.

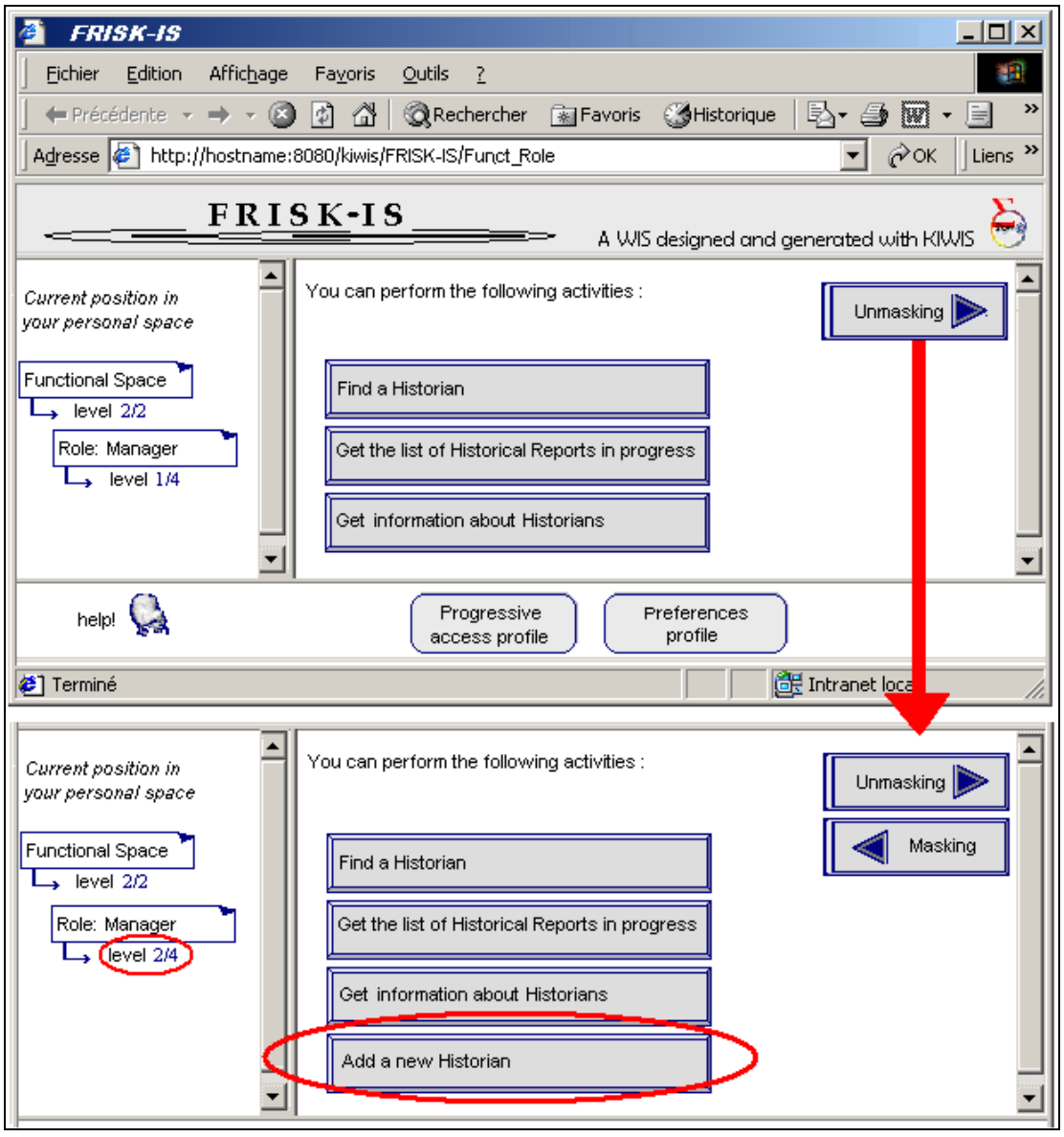


Figure 17. The Web page showing some functionalities a particular user can perform given the role 'Manager'.

Then, let us suppose that the user U chooses the functionality which allows her to get some information about Historian Experts. Figure 18 shows the web page displayed by the WIS when user U clicks on the button corresponding to this functionality in the functional role page (Figure 17). Through this page are now presented to user U the instances of the Historian class of the underlying Data Model which correspond to the result of the query associated with the chosen functionality. These instances are presented according to the preferences of user U. In that case, the three level intensional

stratification defined for the Historian class in Figure 7 applies. Instances of Historians are listed in a table and, at the first level of detail, only the values of the three variables of this class (*Lastname*, *Firstname*, *ExpertisePeriod*) are visible. If user U wants to get more information about these instances, at the second level of detail reached by clicking on the unmasking button, she will access to two more variables (*Institute*, *Address*), as shown at the bottom of Figure 18.

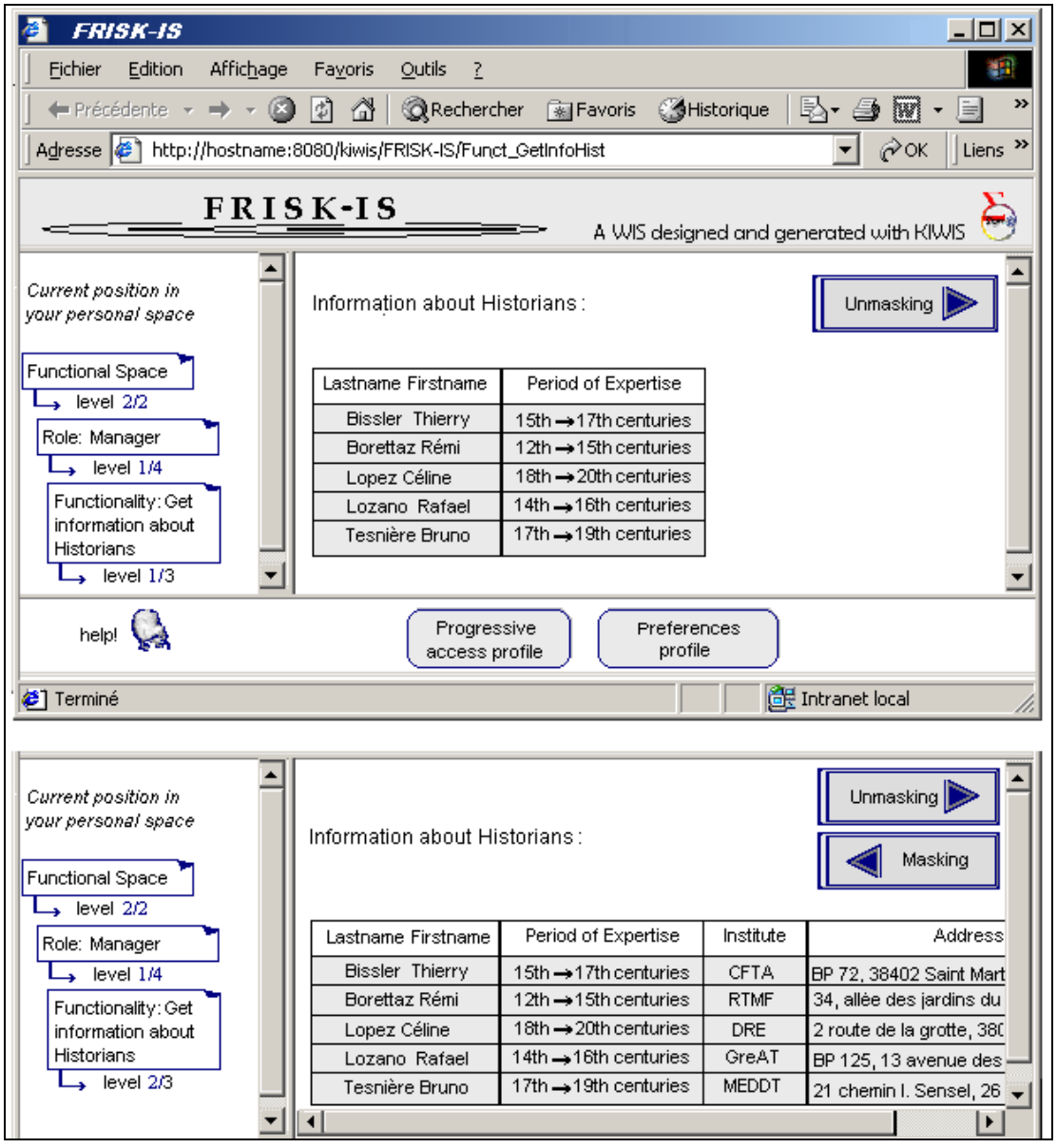


Figure 18. The Web page showing the result of the functionality 'Get information about Historians' for a particular user.

Lastly, on Figure 19 is shown the web page displayed by the WIS for the same functionality, *i.e.* 'Get information about Historians' for another user V. For this user, the presentation of the functionality differs in some points: the spatial layout of the page components (here, three frames) is different, instances of the Historian class are not presented in a table, but consecutively in the Web

page and can be accessed by scrolling. Moreover, for user V, the masking and unmasking functions are available for each instance independently from the others as shown at the bottom of Figure 19.

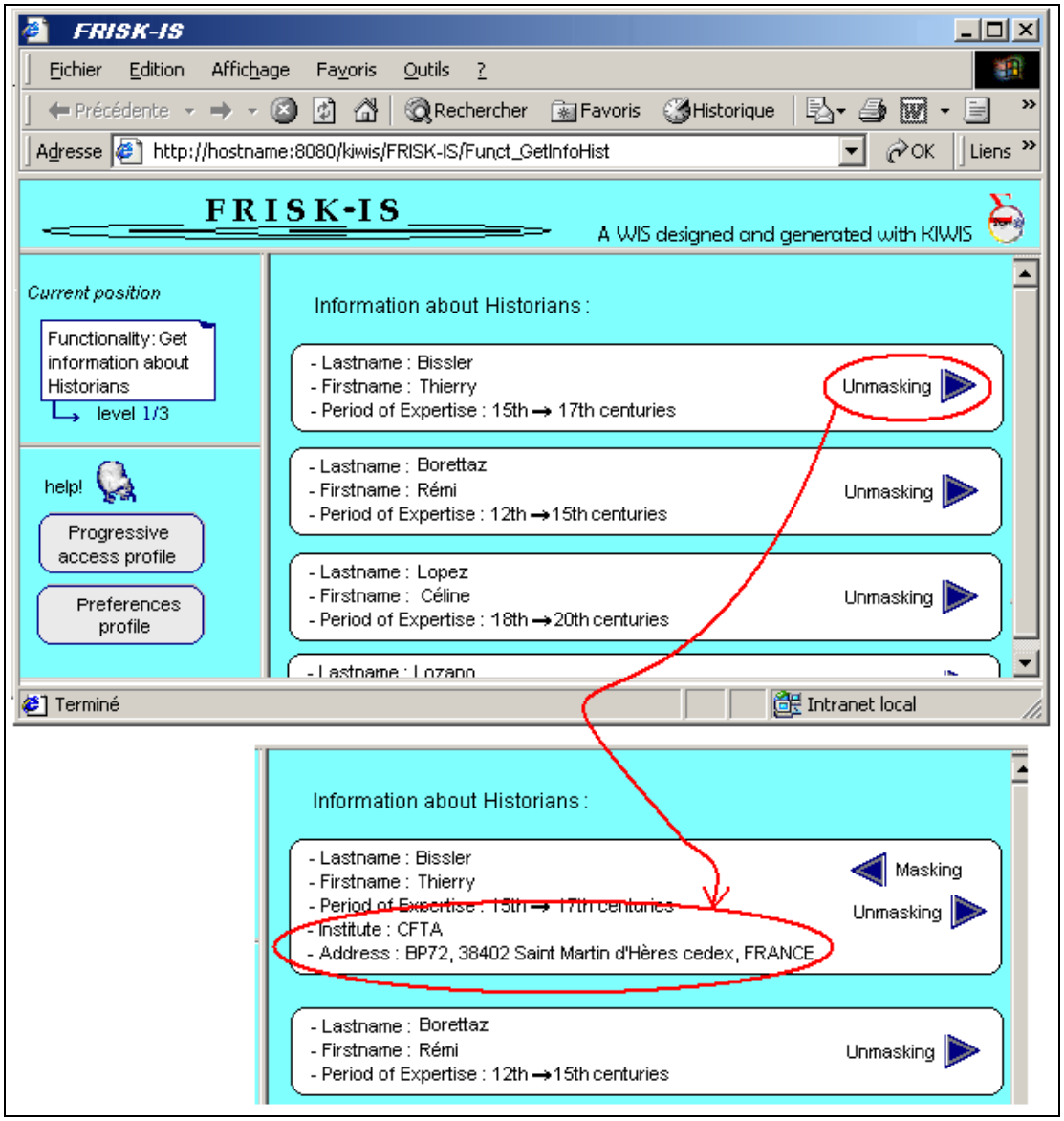


Figure 19. The Web page showing the result of the functionality 'Get information about Historians' for another user.

6. Related Work

In the last fifteen years, several methods have been proposed for the design of applications ranging from Web sites (hypermedia and hypermedia based on the Web) to WIS. Basically, the main difference between these two kinds of application stands in the complexity of services they offer [1][18]. Whereas services are almost non-existent in Web sites, WIS provide their users with functionalities like traditional information systems do. In the literature, some methods (for instance,

OOHDM [19], WSDM [20], AWIS-M [14] or UWE [15] put the emphasis on functional aspects more than others do (e.g. HDM [21] or RMM [22]). The former generally adopt a *user needs driven approach* we have also chosen. It should be noticed that, although WebML [23] is not explicitly presented as a method relying on an analysis step of the functional requirements, WebML also addresses the functional dimension through the concept of *operation units* [5].

All these approaches envision the design activity as a multi-faceted one, which aims at producing various models corresponding to the different dimensions of a WIS. When studying these methods, one can notice that these models are often named differently, are more or less central in the whole proposition, and some models are sometimes split (respectively merged) into more specific sub-models (respectively one global model). Nevertheless, the methods generally agree on, at least, three models we also adopt in our design approach:

- a data model which allows to describe the concepts of the domain,
- a model which concerns the structuration and navigation features of the hypermedia,
- a presentation model which aims at defining the appearance of the Web pages.

Moreover, when considering the set of these propositions, one can identify:

- models for the analysis of requirements such as the User Interaction Diagrams in OOHDM, the Users Objectives Model in AWIS-M or UML use cases in UWE. In our approach, we also exploit UML use cases and sequence diagrams which are extended in order to fit the progressive access approach.
- User models which aim at supporting adaptation to users. These models generally maintain information about users characteristics and/or preferences. Such information is exploited in an adaptation process which concerns (a part of) the WIS dimensions. For instance, preferences about presentation are stored in a user profile in Hera [16]. AWIS-M takes into account both the different levels of knowledge of the users (concerning the application domain, the computer environment, etc.) and a description of their goals in the system. UWE proposes a user model which has to be extended according to the adaptability requirements, which is also possible in our user model (by adding new profiles for instance).
- Adaptation models in Hera, UWE, WebML and AWIS-M. The later is based on a mathematical model whereas the formers exploit rule mechanisms. In our approach, there is no independent and specific adaptation model since the PAM itself expresses both information and rules for adaptation.

Concerning the progressive access, the possibility to define short or long version of information is suggested in WebML, which appears to be the most similar approach to our proposition. Nevertheless several differences can be found. First, the different versions of information in WebML are not built with the objective of offering a progressive access, but in order to choose between one *or* the other version to be presented. Thus, there is no notion in WebML which could be compared to the masking and unmasking mechanisms for navigating between different levels of detail. Second, the definition of multiple representations of information in WebML is an option offered to WIS designer but does *not* constitute a guideline for the design of WIS as the Progressive Access approach does. Finally, the definition of multiple versions of information in WebML concerns the *entities* of the *structural model*, which roughly correspond to the classes of the Data Model in our proposition. The Progressive Access approach then applies to a wider diversity of concepts which are both relative to the data (i.e. schema

or classes, but also associations, and results of requests), and to functional aspects (i.e. functional spaces and functional roles).

Our proposition deals with the management of the quantity of information delivered by WIS having a hypermedia dimension. The progressive access aims at reducing the disorientation syndrome and the risk of cognitive overload. To our knowledge, there is no method in the literature which addresses this issue by proposing a notion similar to the progressive access, integrates it in a WIS design approach from the conceptual level and investigates the impact of such a notion on each dimension of the WIS (content, functional, hypermedia).

7. Conclusion and Future Work

We have defined and presented in this paper the notion of Progressive Access as a contribution for adaptability in Web-based Information Systems (WIS). This notion helps in limiting the cognitive overload and the disorientation a user may feel when, browsing in a WIS, she/he is confronted to a too large amount of information. Progressive Access consists in stratifying the information space in different levels of detail through which the user can navigate by means of masking and unmasking mechanisms. We have proposed five models to implement a progressive access in a WIS. The Progressive Access Model (PAM) describes the entities of the application domain which can be (un)masked. The PAM is general enough to apply to different representation formalisms. Around this central model, four models provide WIS users with a progressive access approach: the Domain Model describes the entities of the application domain targeted by the WIS. The Functionality Model describes the tasks users can performed within the WIS. It has three granularity levels: functional space, functional role and functionality. Each of these levels is linked to the PAM giving the user a progressive access to the WIS functionalities. From the specifications of the Functionality Model, a logical representation of the hypermedia of the WIS is derived by the Structuration and Navigation Model which constitutes the first part of the Hypermedia Model. The second part of this model is a Presentation Model which describes the layout of the Web pages displayed by the WIS through a composition charter, and the look of these pages through a graphical charter.

Consequently, the design of a WIS which provide its users with a progressive access to information, requires to instantiate the five models we have presented. This task can be considered as complex. In order to guide a WIS designer in this task, we have proposed a seven step design method which leads to the complete design of a progressive access based WIS. We have implemented these methodological guidelines into a system called KIWIS. KIWIS is a platform for the design and the generation of adaptable WIS, whose architecture encompasses the five models for progressive access. We have experienced the progressive access approach in the SPHERE project [13], in which, using KIWIS, we have built an Information System dedicated to geographic and historical data on river flooding. Masks and stratifications have shown to be of a rather intuitive use for different categories of users (hydrologic experts, city hall employees, etc.) who consult data about the same topic (floods) but at different levels of detail and with different centers of interest.

We are currently extending the notion of progressive access towards dynamic adaptability. The idea is to propose a description of the behavior of users together with decision rules which will be dynamically (at runtime) exploited by the WIS in order to redefine (re-organize) the existing stratification, if it appears that the different levels of detail defined so far are not so well-suited. For instance, when the system detects that the user always accesses the third level of detail, never using the first two ones, then the current stratification should be changed so that this third level of detail be merged with the first. Already, we have defined a set of operations which enable to reorganize a

stratification. These operations should now be integrated into event/condition/action rules in charge of dynamically reorganizing the stratifications at runtime. Here, events could be linked to the observation (by means of tracking techniques, for instance) of the behavior of the user, conditions could involve some decision thresholds, and finally actions could be materialized by these reorganization operations.

The characteristics of the user's equipment (PDA, laptop…), the network activity, the time available for the WIS consultation, etc., are also features that we intend to take into account in order to improve the adaptation of the hypermedia dimension of the WIS.

Acknowledgements

We would like to thank the anonymous reviewers for their very helpful comments.

REFERENCES

[1] Mecca G., Merialdo P., Atzeni P. & Crescenzi V., The ARANEUS Guide to Web-Site Development, ARANEUS Project Working Report, AWR-1-99, March 1999.

[2] Isakovitz T., Bieber M. & Vitali F., Web Information Systems, Communications of the ACM, 41(7), July 1998, pp. 78-80.

[3] Fraternali P., Tools and Approaches for Developing Data-Intensive Web Applications: A Survey, ACM Computing Surveys, 31(3), September 1999, pp. 227-263.

[4] Baresi L., Garzotto F. & Paolini P., From Web Sites to Web Applications: New Issues for Conceptual Modeling. In Proceedings of the International Workshop on The World Wide Web and Conceptual Modeling, co-located with the 19th International Conference on Conceptual Modeling, Salt Lake City (USA), October 2000.

[5] Ceri S., Fraternali P., Bongio A. & Maurino A., Modeling data entry and operations in WebML, WebDB 2000, Dallas, USA, 2000.

[6] Brusilovsky P., Methods and Techniques of Adaptive Hypermedia. In Brusilovsky P., Kobsa A. & Vassileva J. (eds.), Adaptive Hypertext and Hypermedia, Kluwer Academic Publishers, 1998, pp. 1-43.

[7] Rosenberg M., The personalization story, ITworld.com, 11th may 2001. http://www.itworld.com/Man/2676/ITW010511rosenberg/

[8] Stephanidis C., Paramythis A., Akoumianakis D. and Sfyrakis M., Self-Adapting Web-based Systems: Towards Universal Accessibility, 4th Workshop on User Interface For All, Stockholm, Sweden, October, 1998.

[9] Conklin J., Hypertext: An introduction and survey. IEEE Computer 20(9), 1987, pp. 17-41.

[10] Villanova M., Gensel J. & Martin H., Progressive Access to Knowledge in Web Information Systems through Zooms, 7th International Conference on Object Oriented Information Systems (OOIS2001), Calgary, Canada, August 27-29, 2001, pp467-476.

[11] Villanova-Oliver M., Adaptabilité dans les systèmes d'information sur le Web : modélisation et mise en œuvre de l'accès progressif, Thèse de Doctorat, Institut National Polytechnique de Grenoble, décembre 2002 (in French). Available at http://www-lsr.imag.fr/Les.Personnes/Marlene.Villanova/THESE/

[12] Object Management Group, Unified Modeling Language Specifications, Version 1.4, Sept. 2001. http://www.omg.org

[13] Davoine, P.A., Martin, H., Trouillon, A., Cœur, D., Lang, M., Bariendos M., Llasat C.: Historical Flood Database for the European SPHERE Project: modelling of historical information, 21th General Assembly of the European Geophysical Society, Nice (2001)

[14] Gnaho C., Web-based Informations Systems Development – A User Centered Engineering Approach, in Murugesan S. & Deshpande Y. (Eds.), Web Engineering: Managing Diversity and Complexity of Web Application Development, LNCS 2016, 2001, pp. 105-118.

[15] Koch N., Software Engineering for Adaptative Hypermedia Systems – Reference Model, Modelling Techniques and Development Process, Ph.D Thesis, Fakultät der Mathematik und Informatik, Ludwig-Maximilians-Universität München, December 2000.

[16] Frasincar F., Houben G-J. & Vdovjak R., An RMM-Based Methodology for Hypermedia Presentation Design, Proc. of the 5th East European Conference on Advances in Databases and Information Systems (ADBIS 2001), LNCS 2151, Vilnius, Lithuania, September 25-28, 2001, pp. 323-337.

[17] M. Page, J. Gensel, C. Capponi, C. Bruley, P. Genoud, D. Ziébelin, D. Bardou and V. Dupierris, A New Approach in Object-Based Knowledge Representation: the AROM System, IEA/AIE 2001, Budapest, Hungary, June, 2001.

[18] Conallen J., Building Web applications with UML, Addison-Wesley Longman, 2000.

[19] Rossi G., Schwabe D. & Guimarães R., Designing Personalized Web Applications, Proc. of the 10th International World Wide Web Conference (WWW10), Hong Kong, China, May 1-5, 2001.

[20] De Troyer O.M.F. & Lejeune C.J., WSDM : a User-Centered Design Method for Web Sites, Proc. of the 7th International World Wide Web Conference (WWW7), Brisbane, Australia, April 14-18, 1998.

[21] Garzotto F., Mainetti L., & Paolini P. HDM2: Extending the E-R Approach to Hypermedia Application Design, in Elmasri R., Kouramajian V., & Thalheim B. (Eds), Proc. of the 12th Int. Conference on the Entity-Relation Approach (ER'93), Arlington, TX, December 1993, pp. 178-189.

[22] Isakowitz T., Stohr A. & Balasubramanian E., RMM : A methodology for structured hypermedia design. Communications of the ACM 38(8), pp. 34-44, August 1995.

[23] Ceri S., Fraternali P. & Bongio A., Web Modeling Language (WebML): a modeling language for designing Web sites, Proc. of the 9th International World Wide Web Conference (WWW9), Amsterdam, Netherlands, May 15 - 19, 2000.