LEVERAGING ADAPTIVE WEB WITH ADAPTATION PATTERNS

ZEMIRLINE N / BOURDA Y / REYNAUD C

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud – LRI

11/2009

Rapport de Recherche N° 1529

Leveraging Adaptive Web with Adaptation Patterns

Nadjet Zemirline[1, 2] Yolaine Bourda[1] Chantal Reynaud[2]

[1]SUPELEC/Department of Computer Science, Plateau de Moulon, 3 rue Joliot-Curie,
91192 Gif sur Yvette Cedex, France
{Nadjet.Zemirline, Yolaine.Bourda}@supelec.fr
[2]Université Paris-Sud XI, CNRS (LRI) & INRIA – Saclay Île-de-France / Projet Gemo,
Bât. G, 4 rue Jacques Monod, Parc Orsay Université, 91893 Orsay Cedex, France
Chantal.Reynaud@lri.fr

ABSTRACT

Adaptive web has emerged as a new challenge for the semantic web. One of the aims of the adaptive web is to adapt a set of web resources to users. Over the last few years, the adaptation problem has been studied in the field of Adaptive Hypermedia. Several systems working on closed corpus resources have been developed. Currently, there is a real challenge: integrating resources available on the Web into these systems. More and more metadata describing resources are available on the Web using Semantic Web languages, and can be reused. Our objective is to build an open corpus AHS by, on the one hand, reusing AHS technologies, particularly the adaptation engine which is the heart of these systems and, on the other hand, reusing resources and their descriptions which are available on the Web. Moreover, we want to allow the creator of an adaptive system not only to reuse adaptation strategies that come with the system, but to also be able to specify his own ones.

We address the problem of adaptation specification basing it on user characteristics. Existing systems either based on rules or ECA languages are complex and not easy to understand.

In this paper, we propose a pattern-based approach to express adaptation strategies in a semi-automatic and simple way. This allows the creator of an adaptive system to define elementary adaptations by using and instantiating adaptation patterns. These elementary adaptations can then be combined, allowing to specify adaptation strategies in an easy and flexible manner. We distinguish adaptive navigation according to two main criteria: the selection operations performed in order to obtain resources being proposed to the user and the elements of the domain model involved in the selection process. We present a taxonomy of elementary adaptive navigation techniques. Our approach has been validated using the GLAM adaptation engine. We showed that the GLAM rules can be automatically generated from pattern-based adaptations.

Categories and Subject Descriptors

[**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia - architectures, navigation, theory, user issues; E.1 [**Data**]: Data Structures, graphs and networks;

General Terms

Design, Theory

Keywords

Patterns, Adaptation strategies, AH systems, adaptive web

1. INTRODUCTION

Adaptive web has emerged as a new challenge for the semantic web [4]. Unlike traditional "one-size-fits-all" web systems, adaptive web takes into account interests, goals, and preferences of individual users in order to adapt them to a set of web resources.

Over the last few years, the adaptation problem has been studied in the field of Adaptive Hypermedia [17]. Several systems working on closed corpus resources have been developed [6][9][16][19]. Currently, there is a real challenge: integrating resources available on the Web into these systems and to be able to propose them to the end-user. This means that, among all the possible adaptations in AH systems, we consider only the navigation adaptation.

Our objective is to build an open corpus AHS by, on the one hand, reusing AHS technologies, particularly the adaptation engine which is the heart of these systems and, on the other hand, reusing resources and their descriptions which are available on the Web. Moreover, we want to allow the creator of an adaptive system not only to reuse adaptation strategies that come with the system, but to also be able to specify his own strategies.

We address the problem of the adaptation of the navigation specification according to user characteristics, which is often very dependent on a particular adaptation language. Moreover, in most existing systems either based on rules [6][16], or ECA (Event-Condition/Action) [9] languages are complex and not easy to understand.

This paper presents our efforts to create a framework for expressing adaptation strategies. Based on a semi-automatic approach, it enables the creator of an adaptive system to express adaptation~~,~~ at a high level, regardless of any adaptation language. The resulting adaptation strategies can be automatically translated, thus, executed by any adaptation engine.

In our framework, we broke-down the traditional adaptation problem to a set of elementary adaptation problems. An elementary adaptation problem "*defines what criterion is used in order to propose a particular set of resources among the web's resources*".

For modeling those elementary adaptation problems, we propose a set of elementary adaptation patterns in the tradition of design patterns [11]. These elementary adaptation patterns are presented in a taxonomy based on "the selection operations performed to obtain resources" and "the criterion on which the operation was done" in order to facilitate their use. Instantiating elementary adaptation patterns on specific elements designed by the creator allow defining elementary adaptations in an automatic way.

These elementary adaptations are exploited to specify complex adaptations. One aspect of the originality of our contribution is that by simple associations between a user characteristic and either elementary or complex adaptation, the creator defines adaptation strategies. The most difficult part of the composition process of elementary adaptations is done automatically.

This paper is organized as follows: First, in section 2, we present the main aspects of our approach. Section 3, the elementary adaptation pattern-based approach is described. Section 4, the definition of elementary adaptations and their combination in order to obtain adaptation strategies are detailed. Section 5, we discuss the validation step using the GLAM adaptation engine. In section 6, we describe closely related works. Finally, section 7 concludes the paper.

2. THE APPROACH

Given two models, a domain and a user model, we propose a pattern-based approach to support the definition of adaptation strategies. This approach is based on a set of building blocks which can be reused and instantiated to define specific strategies.

The main steps of the approach are the following:

1- Selection, by the creator, of the elementary adaptation patterns which are necessary to express adaptation strategies (cf. (1) Fig.1).

2- Instantiation, by the creator, of the selected elementary adaptation patterns by considering the elements of his domain model. This step allows the creator to define a set of elementary adaptations (cf. (2) Fig.1).

3- Composition, of elementary adaptations in order to build more complex adaptations in an automated way, and specification by the creator of associations between user characteristics and adaptations corresponding to adaptation strategies (cf. (3) Fig.1).

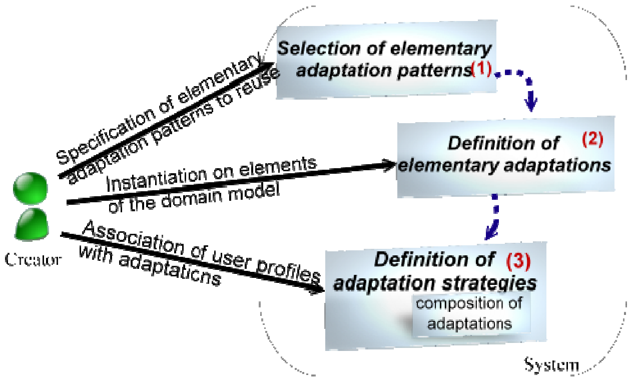
Selection of elementary adaptation patterns (1)
Specification of elementary adaptation patterns to reuse
Definition of elementary adaptations (2)
Instantiation on elements of the domain model
Creator
Association of user profiles with adaptations
Definition of adaptation strategies (3)
composition of adaptations
System

Fig.1 Main aspects of the approach

We propose the following definition of an adaptation strategy:

Definition 1: “*An adaptation strategy defines what resources are to be proposed and how they will be proposed for a set of users who share the same characteristics.*”
Indeed, often the user has multiple characteristics. Defining an adaptation strategy consists in associating user characteristics to adaptations.

This paper addresses each step of the approach.

3. ELEMENTARY ADAPTATION PATTERNS

We propose the following definition for elementary adaptation patterns, based on the definition of design patterns [11].

Definition 2: *An elementary adaptation pattern describes a generic solution of a generic elementary adaptation problem.*

3.1 Description of Elementary Adaptation Patterns

Similar to the description of the design patterns proposed in [11], the characteristics ‘Name, Intent, Solution and Constituents” have been retained to describe elementary adaptation patterns. We define below each of these characteristics when they are used to describe elementary adaptation patterns.

An elementary adaptation pattern is characterized by:

- **Name**: the name of the elementary adaptation pattern described.

- **Intent**: the intent is a short statement about an elementary adaptation problem. It answers the following questions: what is the elementary adaptation pattern supposed to do? i.e. what is its goal? Indeed, it indicates the way the resources are selected and the way they are presented: the proposal of the set of resources are all being considered in the same way, or being ordered or completed with recommendations or preferences.

- **Solution**: the solution includes two elements:

  - **Expressions:** expressions denote a set of resources to be proposed to the user, and the conditions having to be satisfied. These conditions can be represented in one or more logical expressions. Those to be considered simultaneously are gathered in the same expression while excluded conditions are expressed in different expressions. An expression is a conjunction of predicates expressed on the elements of a domain model. In the presence of multiple sets of resources (defined through multiple expressions) we need to specify the way they have to be considered. This is done using meta-expressions.

  - **Meta-expressions:** a meta-expression is a binary relation between two expressions. We propose three relations defining, respectively, an order, and a recommendation or a preference on the sets of resources (defined by the expressions):

    - E_1 *has priority over* E_2 means that the set of resources denoted by E_1 must be proposed before the set of resources denoted by E_2. The set of resources denoted by E_2 will be proposed once all resources denoted by E_1 are consulted.

    - E_1 *is recommended rather than* E_2 means that both sets of resources are proposed, but the set of resources denoted by E_1 will be suggested when the set of resources denoted by E_2 will not.

    - E_1 *is preferable to* E_2 means that the set of resources denoted by E_1 will be proposed when the set of resources denoted by E_2 will not, except if the set of resources denoted by E_1 is empty,.

Consequently, in that case, only one set of resources will be proposed to the user.

- **Constituents**: they describe the elements of the domain model used in all the expressions described in the solution pattern.

3.2 Typology of Elementary Adaptation Patterns

Here, we propose a typology of elementary adaptation patterns for the adaptation of navigation for open corpus AHS. It exploits available web resources and it doesn't modify them. Unlike, the Brusilovsly typology [17], which is proposed in closed corpus AHS (for instance: hiding links).

Adaptation consists in proposing a subset of the available resources. For instance, we need to select resources and to define the way they are proposed either by ordering, recommending them, or by expressing preferences relative to them.

Proposed resources can be obtained by performing four operations. Each operation is done according to a particular criterion. Here are the different operations that can be performed on the available resources:

- *Selection only* means that some resources are chosen in order to be proposed to the user according to a criterion.
- *Ordered selection* means that the resources to be proposed are selected and sequenced according to a criterion. This defines a partial order between resources. For instance: in the e-learning domain, resources can be selected and ordered using the pre-requisite relation between resources.
- *Recommended selection* means that the resources to be proposed are selected and some of them are recommended according to a criterion when others are not. We can use different colors to distinguish between recommended resources, for instance, we can recommend definitions rather than exercises. The user will be able to access both definitions and exercises and a typographical indication may be used to express which resource is recommended more.
- *Preferred selection* means that the resources to be proposed to the user are selected and some of them are preferable to others according to a criterion. Only one set of resources is proposed to the user. The set of preferable resources is proposed as a priority. The other resources will be proposed only if this first set is empty.

The criterion cited above can be based on either the classes (e.g. Definition), or characteristics related to classes (e.g. the format of a resource) or relations between classes (e.g. the prerequisite relations between resources). All available resources are described as instances of the class Resource or of one of its specializations. So, relations in which the class Resource is involved both as the domain and the range, generate links between instances of the class Resource (i.e. links between specific resources), and define a graph on which navigational paths can be specified.

Often, in systems like AH, we don't consider the resources only, but also concepts associated with them. A concept is an abstract notion to which resources are related, and concepts can be organized in a hierarchy. A concept is linked by a bidirectional relation to one or more resources. Hence, navigational paths can be defined either as:

- Instances of the class Concept if there is at least one relation in which the class Concept is involved as the domain and the range.
- Instances of the class Resource if there is at least one relation in which the class Resource is involved as the domain and the range.

We consider both depth-first-search and breadth-first-search expressed either of being on the graph of concepts or of being on the graph of resources.

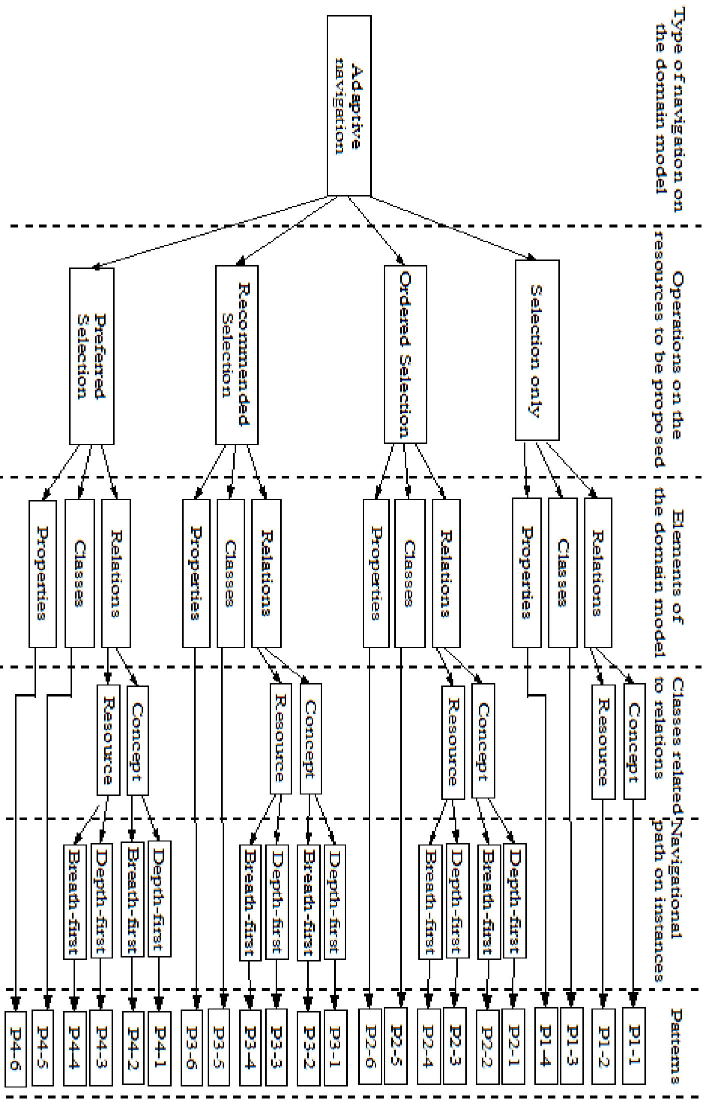

Fig.2 Typology of elementary adaptation patterns.

Fig.2 presents the typology of elementary adaptations that we propose. It is built according to (1) the selection operations performed in order to obtain the proposed resources, (2) the elements of the domain model involved in the selection process (3) the type of navigation on the resources or concept graph. This typology is presented as a tree. Each leaf of the tree is an elementary adaptation pattern referenced by a code that the name has been built as follows:

The selection operation - the element of the domain model - the resources or concepts graph - the type of navigation.

3.3 Some Elementary Adaptation Patterns

Due to lack of space, we describe here only three elementary adaptation patterns proposing different operations on resources and based on different elements of a domain model.

In the following, we use:

- *r*: to denote a variable that is a resource. It is an instance of the class Resource or of one of its specializations.
- *goal*: to denote a particular resource that the user wants to reach.

We recall that resources are the concrete objects having to be presented to the user.

3.3.1 Order of resources to be proposed

Here is the description of the elementary adaptation pattern “A2-5: Ordered - Selection - Classes”

- **Name**: Ordered - Selection - Classes
- **Intent**: This pattern proposes ordered resources belonging only to the following subclasses of the class Resource: $Class_i$ i = 1..n and i<j.
- **Solution**:
  - Expressions
    - E_1: *instanceOf (r, $Class_1$)*
    - E_2: *instanceOf (r, $Class_2$)*
    - ..
    - E_n: *instanceOf (r, $Class_n$)*
  - Meta-expressions
    - E_i has priority over E_j, i<j, i = 1..n and j = 1..n.

  E_i indicates that only the resources of the class $Class_i$ are selected.
- **Constituents**:
  - r: a variable which represents an instance of the class Resource or of one of its specializations.
  - $Class_i$: a variable which represents a sub-class of Resource.

3.3.2 Recommendation of resources to be proposed

Here is the description of the elementary adaptation pattern “A3-3: Recommended - Selection - Resource”.

- **Name**: Recommended Selection-Relation-Resources-Depth First
- **Intent**: This pattern proposes resources linked by the relation relation-precedence and recommending those related to the current document.
- **Solution**:
  - Expressions
    - E_1: *relation-precedence* (r, goal) ^ relation-precedence (r, currentDocument)*
    - E_2: *relation-precedence* (r, goal)*
  - Meta-expression
    - E_1 is recommended rather than E_2.

  E_1 denotes the resources that are linked to the current document with the relation relation-precedence and lead to the goal.

  E_2 denotes the resources that lead to the goal using the relation relation-precedence.
- **Constituents**:
  - r: a variable which represents an instance of the class Resource or of one of its specializations.
  - relation-precedence: a variable which represents a relation defined between instances of the class Resource or of one of its specializations. This relation is transitive.
  - relation-precedence*: the transitive closure of the relation relation-precedence.
  - goal: a variable which represents a resource that the user wants to reach.
  - currentDocument: a variable which represents the resource being currently studied by the user.

3.3.3 Preference of resources to be proposed

Here is the description of the elementary adaptation pattern “A4-6: Preferred-Selection - prop”.

- **Name**: Preferred - Selection - Property
- **Intent**: this pattern proposes resources that satisfy some values of the property prop with prop = $value_1$ are proposed first, if no resources are available prop=$value_2$ are proposed and so on.
- **Solution**:
  - Expressions
    - E_1: *prop (r, $value_1$)*
    - E_2: *prop (r, $value_2$)*
    - ..
    - E_n: *prop (r, $value_n$)*
  - Meta-expressions
    - E_i is preferable rather than E_j i < j, i = 1..n and j = 1..n.

  E_i indicates that only the resources with prop = value $_i$ are selected.
- **Constituents**:
  - r: a variable which represents an instance of the class Resource or of one of its specializations.
  - prop: a variable which represents a property of the class Resource or of one of its specializations.
  - value $_i$: a variable which represents a value among the allowed values of the property “prop”.

4. DEFINING ADAPTATION STRATEGIES

Each elementary adaptation pattern allows the creator to define one possible manner to propose resources. For that, it has to instantiate the elementary adaptation patterns on a specific application domain model. For instance, if he wants to propose definitions before exercises, he has to reuse and instantiate pattern 2.5.

However, often adaptations are more complex and need to be expressed using multiple elementary adaptation patterns. For instance, the creator may want to propose definitions before exercises, all proposed resources being only video ones. In that case, in addition to pattern 2.5, the creator will have to reuse pattern 1.4. That means that the instantiation of the two reused patterns will have to be combined.

Besides, the selected resources to be proposed to the user must be according to his particular profile. A user must consult only resources he is interested in and these resources must be understandable. This is precisely the goal of an "adaptation strategy" which aims at linking user characteristics to adaptations.

The process we used to build adaptation strategies is described below. First, we introduce adaptations in section 4.1. Then, we describe the combination process of adaptations in section 4.2. Finally, we present adaptation strategies in section 4.3.

We note $E_{i,j}$: the expression number j of the adaptation i.

4.1 Definition of Elementary and Composed Adaptations

Elementary adaptation patterns are generic and always need an adjustment phase to specific problems. Here, we propose the definition of an elementary adaptation.

Definition 3: *An elementary adaptation is obtained after instantiation of an elementary adaptation pattern on a particular domain model*.

An elementary adaptation is characterized by the characteristics "Name, Intent, Solution and Constituents" that are instantiated on elements (classes, relations and properties) of a particular application domain model.

The generation of the elementary adaptation is done automatically (not detailed in this paper). The creator has only to select the elementary adaptation pattern and to specify the elements of the domain model on which the elementary adaptation should be expressed.

Here is an example of an elementary adaptation (A_1). It is an instantiation of the elementary adaptation pattern: A2-5 "Ordering-Selection-Classes".

- **Name**: Ordering - Selection - Definition - Exercise.
- **Intent:** This pattern proposes ordered resources belonging only to the following subclasses of the class Resource: Definition before Exercice.
- **Solution**:
  - ➢ Expressions
    - ▪ E_{1-1}: *instanceOf (r, Definition)*
    - ▪ E_{1-2}: *instanceOf (r, Exercise)*
  - ➢ Meta-expression:
    - ▪ E_{1-1} has priority over E_{1-2}
- **Constituents**:
  - ➢ r: a variable which represents an instance of the class Resource or of one of its specializations.
  - ➢ Definition: a direct or indirect specialization of the class Resource.
  - ➢ Exercise: a direct or indirect specialization of the class Resource.

Often, multiple elementary adaptation patterns need to be reused to express adaptation, and then they are combined. The result of the combination process is a composed adaptation.

Definition 4: "*A composed adaptation is obtained by combining elementary adaptations*".

From a structural point of view, a composed adaptation has the characteristics: "Name, Intent, Solution and Constituents". It is not different from an elementary adaptation. In the following, we use "*adaptation*" to refer to either an elementary or a composed adaptation.

The following section presents the combination process of adaptations.

4.2 Combining Adaptations

An adaptation has multiple characteristics. The combination process of a set of adaptations consists in combining together each of the characteristics of this set. We propose:

- A manual process to combine the characteristics "Name, Intent" as it needs natural language processing (not detailed in this paper).
- An automatic process to combine the characteristics "Solution, Constituents". The combination of the characteristic "Constituents" is simple. Constituents coming from the different adaptations are gathered together in order to obtain a unique set of "Constituents".

Here, we describe the combination process of the Characteristic "Solution". It includes expressions and meta-expressions:

- As expressions denote the selected sets of resources, we propose combining them together. However, those sets can be disjoint or not, i.e. resources of a similar nature that don't satisfy the same constraints lead to different sets of resources and can't be combined. So, during the combination we differentiate between disjoint sets of resources and those not disjoint. Two combination steps are proposed.
- As meta-expressions denote how the selected sets of resources are organized, we propose combining them together and they are combined differently according to each combination step.

For instance, the adaptation "Ordering-Selection-Definition-Exercise" can be combined with the adaptation "Selection only - format", but it can't be combined with the adaptation "Selection only - Example".

Given a set of adaptations, our objective is to make one composed adaptation that includes all the initial sets of adaptations. To achieve that, we propose two successive steps:

Step 1: Build different sets of adaptations, each set being composed of adaptations based on excluded criteria.

Step 2: Build one adaptation from the sets of adaptations built in step 1.

4.2.1 *Step One of the Combination*

Adaptations are defined by exploiting the elements of the domain model which are relations, classes and properties.

- Relations are used to define different search types in a graph of resources or of concepts. For instance, adaptations based on the "prerequisite" or "Part-Of" relations define different adaptations for searching in a graph of resources.
- Classes are used to define a classification of resources. For instance, adaptations based on definitions and exercises, or adaptation based on examples and Details. They define distinct adaptations exploiting the classification of resources.
- Properties define characteristics of resources. Values of properties are constraints that have to be respected by the proposed resources. For instance, the format must be text or image or video, the knowledge level must be high or low. Each property is independent from the other. However, adaptations defined on values of the same property define distinct adaptations.

We partition the set of adaptations having to be combined in several subsets, each subset being composed of adaptations based on excluded criteria. We group all adaptations exploiting relations, whatever they are, in the same set. We group all adaptations exploiting classes and gather adaptations expressed on the same property (there are as many sets as properties).

After, we build one adaptation from the multiple adaptations belonging to each subset. The expression part of this resulting adaptation is the union of the expressions of the subset adaptations and the meta-expressions part is the union of the meta-expressions of the subset adaptations.

The result of the first step is a set of adaptations, one per subset. The second step aims at combining them.

4.2.2 *Step Two of the Combination*

Let P be the number of adaptations obtained after the first step. Let A_i with i = 1..p, be an adaptation composed of n expressions and m meta-expressions. Let A_c be the adaptation built from the combination of the p adaptations according to the second step of the combination process.

The combination of expressions of adaptations in step 2 consists in building conjunctive formulae from the expressions of the p adaptations in order to obtain the expression part of the resulting adaptation A_c. Let:

- The set of expressions of A_1: $E_{1\text{-}j1}$ with $j_1 = 1... n_1$.
- …
- The set of expressions of A_p: $E_{p\text{-}jp}$ with $j_p = 1... n_p$.

The expression part of A_c is built from: $E_{c,h} = E_{1\text{-}j1} \wedge E_{2\text{-}j2} \wedge \ldots \wedge E_{p\text{-}jp}$ with $h=1\ldots n_1 * n_2 * \ldots * n_p$.

The combination of meta-expressions of adaptations in step 2 consists in determining the meta-expressions on the $E_{c,h}$ expressions. This process exploits the meta-expression of the P adaptations obtained from step 1 of the combination. They are obtained as follows:

Let $E_{c,h1}$ and $E_{c,h2}$ be two expressions belonging to the adaptation A_C, as:

- $E_{c,h1} = E_{1\text{-}j1} \wedge E_{2\text{-}j2} \wedge \ldots \wedge E_{n\text{-}jn}$ with $j_1 = 1...n_1$, $j_2 = 1\ldots n_2, \ldots j_n = 1\ldots n_n$
- $E_{c,h2} = E_{1\text{-}k1} \wedge E_{2\text{-}k2} \wedge \ldots \wedge E_{n\text{-}kn}$ with $k_1 = 1\ldots n_1$, $k_2 = 1\ldots n_2 \ldots j_n = 1\ldots n_n$

We deduce meta-expression between $E_{c,h1}$ and $E_{c,h2}$ from existing meta-expressions defined between the expressions of the P adaptations.

A meta-expression is a binary relation between two expressions, which is anti-symmetric. The generation of a relation and its anti-symmetric entity creates a conflict. For instance: *$E_{1\text{-}1}$ has priority over $E_{1\text{-}2}$* and *$E_{1\text{-}2}$ has priority over $E_{1\text{-}1}$*. The generation of two meta-expressions between two identical expressions also makes for a conflict. For instance: *$E_{1\text{-}1}$ has priority over $E_{1\text{-}2}$* and *$E_{1\text{-}1}$ is recommended rather than $E_{1\text{-}2}$*.

In order to facilitate the combination process for the creator, we propose retaining only one meta-expression of those in conflict and we propose a selection criterion for each type of conflict.

We choose to process these conflicts automatically. Combined adaptations are ordered. We propose a default order but the creator is free to change it according to his needs. The default order that we propose is the following:

- Criterion 1: Navigational path of the graph.
- Criterion 2: Type of resources.
- Criterion 3: Characteristics of resources.

While a criterion includes multiple adaptations, those adaptations are arranged randomly.

This first solution allows resolving the two types of conflicts, but we choose a finer criterion to resolve the second type of conflict, which is to consider meta-expressions at different levels. This means that the different priorities are allocated to the meta-expressions according to the represented relations: (1) Priority (2) Recommendation (3) Preference. Once more, it is a default order which can be modified if needed.

The goal in this step is to deduce the meta-expressions of the adaptation A_C. For that, we perform the following steps:

The meta-expressions are ordered according to which type of relation they belong. The deduction process is incremental: we add meta-expressions in a set that do not generate conflicts. When a meta-expression generates a conflict of the first type, it is automatically not considered and the deduction process will continue. When it generates a conflict of the second type we retain only one meta-expression according to the order defined in the associated solution of the second conflict.

4.2.3 *Example of a Combination of Adaptations*

Here is an example illustrating the combination process. Due to lack of space, we will give only an extract from each adaptation.

Let A_1, A_2, A_3 be three adaptations having to be combined:

- A_1 is the elementary adaptation Ordering – Selection – Definition - Exercise described in 4.1.

- A_2 is an elementary adaptation defined by instantiating the elementary adaptation pattern A2-6:
  - **Name**: Ordered - Selection - Format.
  - **Intent**: this adaptation proposes resources that satisfy some values of the property format with format = text before format = image.
  - **Solution**:
    - Expressions:
      - E_{2-1} : *format (r, text)*
      - E_{2-2} : *format (r, image)*
    - Meta-expression:
      - E_{2-1} has priority over E_{2-2}
- A_3 is an elementary adaptation defined by instantiating the elementary adaptation pattern A3-6:
  - **Name**: Recommended - Selection - difficulty-level
  - **Intent**: this adaptation proposes resources that satisfy some values of the property difficulty-level with format = high rather than format = low.
  - **Solution** :
    - Expressions:
      - E_{3-1} : difficulty-level *(r, low)*
      - E_{3-2} : difficulty-level *(r, high)*
    - Meta-expression:
      - E_{3-1} is recommended rather than E_{3-2}

Step 1 of the combination. The adaptations A_1, A_2, A_3 are grouped in different sets according to the elements of the domain model they are expressed on. This step consists in the partitioning of the adaptations. Here, we have three adaptations expressed on three different elements of the domain model: one adaptation A_1expressed on classes and two adaptations A_2, A_3expressed on two different properties, so we get three sets: one adaptation per set.

Step 2 of the combination. The set of adaptations obtained in step 1 are combined together in order to get one composed adaptation A_c.

The expressions in the solution part of A_c are the following:

- $E_{C,1}$= E_{1-1}^E_{2-1}^ E_{3-1}= *instanceOf (r, Definition)^ format (r, text)^* difficulty-level *(r, low)*
- $E_{C,2}$=E_{1-1}^E_{2-1} ^ E_{3-2}=*instanceOf (r, Definition)^ format (r, text)^* difficulty-level *(r, high)*
- $E_{C,3}$= E_{1-1}^ E_{2-2} ^ E_{3-1}= *instanceOf (r, Definition)^ format (r, image)^* difficulty-level *(r, low)*
- $E_{C,4}$=E_{1-1} ^E_{2-2} ^ E_{3-2}=*instanceOf (r, Definition)^ format (r,image)^* difficulty-level *(r, high)*
- $E_{C,5}$= E_{1-2}^E_{2-1}^ E_{3-1}= *instanceOf (r, Exercise)^ format (r, text)^* difficulty-level *(r, low)*
- $E_{C,6}$=E_{1-2}^E_{2-1} ^ E_{3-2}=*instanceOf (r, Exercise)^ format (r, text)^* difficulty-level *(r, high)*
- $E_{C,7}$= E_{1-2}^ E_{2-2} ^ E_{3-1}= *instanceOf (r, Exercise)^ format (r, image)^* difficulty-level *(r, low)*
- $E_{C,8}$=E_{1-2} ^E_{2-2}^ E_{3-2}=*instanceOf (r, Exercise)^ format (r,image)^* difficulty-level *(r, high)*

The meta-expressions in the solution part of A_c are obtained by the exploitation of the meta-expressions coming from A_1 then those of A_2, and after, those of A_3. This is because, A_1 is expressed on classes and A_2 and A_3 on properties.

Among the deduced meta-expressions and retained meta-expressions, we have: *$E_{C, 1}$ has priority over $E_{C, 5}$, $E_{C, 1}$ has priority over $E_{C, 3}$, $E_{C, 2}$ has priority over $E_{C, 5}$, $E_{C, 2}$ has priority over $E_{C,3}$, $E_{C, 3}$ is recommended rather than $E_{C, 4}$, $E_{C, 5}$ is recommended rather than $E_{C, 6}$, $E_{C, 7}$ is recommended rather than $E_{C, 8}$.*

However, among the deduced meta-expressions and the un-retained meta-expressions, we have: *$E_{C, 5}$ has priority over $E_{C, 3}$, $E_{C, 5}$ has priority over $E_{C, 4}$, $E_{C, h6}$ has priority over $E_{C, 3}$* and *$E_{C, 6}$ has priority over $E_{C, 4}$, $E_{C, 1}$ is recommended rather than $E_{C, 4}$.*

5. VALIDATION

For validation, we choose the complex system GLAM [16], and we integrate our pattern-based adaptation above GLAM. It is easier to define adaptation using our framework rather than using GLAM, and the adaptation is automatically generated in the GLAM format.

GLAM (Generic Layered Adaptation Model) platform is defined for an entire class of adaptive hypermedia systems. The platform is made up of a generic adaptation model relying on generic user and domain models. Specific systems can be obtained by specializing the GLAM generic user and domain Models.

We found the essential elements of our framework in GLAM: the modeling of resources, of concepts, of relations between either resources or concepts and between concepts and resources, and of properties.

Besides, in the GLAM adaptation, the creator has to manually find all conditions that must satisfy the proposed resources and to manually compose the different conditions, this is expressed using rules in GLAM, so one must write all the needed rules in the system. Then, he has to define which rules will be applied to which user, (which is expressed by meta-rules in GLAM), then he also has to manually write these meta-rules. While using elementary adaptation patterns, the creator just has to choose those to use and on which elements of the domain model to use them. The instantiation of elementary adaptations is done automatically. These adaptations can be used as many times as needed in his adaptation strategies and the composition of the difficult part is done automatically for him.

Moreover, the GLAM rules include repetitive parts. An example of an adaptation had been defined using 30 rules in GLAM [15], while its needs only 12 ‘elementary adaptations’ to do it. There are 9 rules proposing a depth search [15], i.e., the same condition selecting resources according to depth first is written 9 times. Similarly, there are 11 rules selecting the type of resource explanations [15], etc.

5.1 Adaptation Strategies in GLAM

An adaptation strategy, in GLAM is described on two levels:

- A Level based only on domain-related knowledge. It concerns data about the domain model and the position of the user in the domain model. It is exploited using rules.
- A Level based on user-related knowledge. It concerns user characteristics. It is exploited using Meta-rules.

5.1.1 GLAM Rules

Rules are expressed using a condition-conclusion format as *Predicate $_1$ ^… ^ predicate $_n$ $\rightarrow$ Action (resource $_i$, degree)*

The condition part describes the condition that must be satisfied by the resources proposed to the user. Usually, this choice is composed of: a particular navigational path of the domain model, perhaps a type of resources and perhaps restrictions on the resource format using concepts or resource attributes.

The conclusion part describes what will be done by the selected resources. It includes two elements:

- Action: describes the activity that will be proposed to the user for the resource $_i$.
- Degree: can be used in different treatments. In GLAM, it is used to describe the relevance of resources relative to each other. It allows proposing several resources for the user with a code of degrees of relevance (color). It is implemented in five values (very high, high, medium, low, and very low) where each value is associated with a particular color.

Here is an example of one rule in GLAM.

Rule 1: $\forall R \in$ {r/ type(r, explanation)}, abstraction(r, Concept)^ prerequisites_ (Concept, goal) ^ abstraction (currentDocument, Concept2) ^ hierarchy (Concept2, Concept) $\rightarrow$ Read(r, degree)

Its condition part includes two subparts:

- Subpart 1: « *r/type(r, explanation)* » describes resources defined as instances of the Explanation that we are interested in and which would be proposed to the user.
- Subpart 2: « *abstraction (r, Concept)^ prerequisites_ (Concept, goal) ^ abstraction (currentDocument, Concept2) ^ hierarchy (Concept2, Concept)* » characterizes Concept and concept2 compared to the resource r and to the current document. These conditions are those that must be satisfied when we choose to propose resources according to an in-depth navigational path.

5.1.2 GLAM Meta-rules

Meta-rules allow describing mechanisms for the rules selection, scheduling, and excluding in order to select rules for a given user according to his profile. Let R_1, R_2 be two sets of rules, where four types of meta-rules are proposed. Each meta-rule is a binary relationship between rules:

- *Preference meta-rules between R_1 and R_2* means that we prefer to execute R_1 rather than R_2, noted $R_1 > R_2$.
- *Requirement meta-rule between R1 and R2* means that the execution of R_1 needs the execution of R_2, noted $R_1 \supset R_2$.
- *Exclusion meta-rule between R_1 and R_2* means that either R_1 or R_2 is executed, noted $\overline{R_1R_2}$.
- *Order meta-rule between R_1 and R_2* mean that R_1 is executed before R_2. The order meta-rules define a strict order between elements on which they are expressed, noted $R_1 \prec R_2$.

We note that in GLAM, two types of orders can be proposed. A *partial order* expressed using the degrees of desirability and a *total order* expressed using the order meta-rules.

5.2 Automatic Generation of GLAM Adaptation

In this section, we demonstrate how our adaptations,(those that are obtained after instantiation of elementary adaptation patterns and eventually, after their combination) can be translated automatically into the GLAM format.

GLAM had been designed for closed corpus AHS, taking into account several possible actions. But here, in the context of open corpus AHS, we consider only the "read" actions.

The process we used to automatically generate rules and meta-rules in GLAM is described below.

5.2.1 Translation of Expressions in GLAM

For each expression E_i belonging to the set of all the expressions composing an adaptation strategy, we generate a rule R_i as follows:

- The condition part of R_i is made of E_i.
- The conclusion part of R_i is generated with a desirability degree set to "medium".

5.2.2 Translation of Meta-expressions in GLAM

For each meta-expression M_k belonging to the set of all the meta-expressions composing an adaptation strategy, we perform:

- If the kind of M_k is "E_i has priority over Ej", and if R_i (resp. R_j) is the rule obtained from E_i (resp. E_j), we generate the following meta-rule $R_1 \prec R_2$.
- If the kind of M_k is "E_i is preferable to Ej", and if R_i (resp. R_j) is the rule obtained from E_i (resp. E_j), we generate the following meta-rules $R_1 \prec R_2$ and $\overline{R_1R_2}$.
- If the kind of M_k is "E_i is recommended rather than Ej", and if R_i (resp. R_j) is the rule obtained from E_i (resp. E_j), we modify the conclusion part of R_i (resp. R_j) as follows: R_i takes a degree of desirability higher than the desirability degree of R_j and eventually higher than its previous degree of desirability (resp. R_j takes a degree of desirability lower than the desirability degree of R_i and eventually lower than its previous degree of desirability). Desirability degrees are bounded, when two rules have a desirability degree very high, they can't have a higher degree of that, similar for a desirability degree very bad.

6. RELATED WORKS

There are no directly related works about elementary adaptation patterns and their composition, in building complex adaptation strategies, independent of any adaptation languages. However, allowing a creator of an adaptive system to simply specify adaptation strategies in an open corpus context implies addressing some issues. They are discussed in the following paragraphs.

Typology of adaptation

Concerning the classification of the adaptation techniques, there is the reference taxonomy of Brusilovsky [17]. It classifies existing techniques in three groups: adaptive presentation, content adaptation and adaptive navigation support. Both Adaptive presentation and content adaptation techniques manipulate knowledge fragments that can be processed and rendered in a variety of ways depending on user preferences. Whereas adaptive navigation support techniques manipulate the links, i.e., it focuses on aspects of navigational hyperlinks such as adaptation guiding and link hiding. The typology of Brusilovsky relies on the fact that the available resources can be modified and restructured during the adaptation process. Hence, it is suitable for closed corpus AHS, but, it isn't for open corpus AHS where there is no control of the distributed resources.

Generic adaptation languages

Some generic languages (independent of any system) exist to specify adaptation strategies [6][9][16][19][20]. Among them, the LAG language [6] is an implementation of the specification of an adaptation language defined in the LAOS model [7]. It is independent of any AH system. Conversion to WHURLE [19] and to AHA! adaptation engines have been proposed. However, the expression of adaptation strategies is complex; in fact, adaptation has to be specified in a sequential way using a context-free grammar, which is not very suitable for creators.

One of the most well-known AH systems is the AHA! system [9], which is proposed in an open source and mainly used in the educational domain. It supports adaptation in a number of different ways, for instance: adaptation guiding and link hiding. Among the aid it provides, there is a template-based tool called "Graph author" that associates a predefined set of attributes and adaptation rules to each newly created concept by the creator, it also exploits different types of relations defined between concepts. The creator doesn't have to specify the adaptation explicitly; it is done automatically for him from his domain model. However, the creator is obliged to choose between the adaptation strategies provided by the system, and he can't specify his own.

Recently a new Generic Adaptation Language GAL has been developed for describing adaptive hypermedia [20], it proposes abstract constructs (for instance: concepts, attributes) to describe the navigational structure of a web application. GAL wants to be independent from any adaptation engines and plans to generate concrete adaptation rules for the AHA! adaptation engine. However, the description of the navigation adaptation is difficult to specify, because the creator has to write the GAL program according to the formal description of GAL, no aid is proposed for creators.

In conclusion, none of these languages propose a simple way for the creator of an adaptive system to create his own adaptation strategies.

Hypertext and adaptation patterns

Some design patterns for building personalization in web applications have been proposed [10]. They are based on some recurrent design structures found, but they are for the designers and developers of adaptive systems and not for the creators of a particular adaptive hypermedia using an adaptive system.

In the e-learning domain, adaptation design patterns have been proposed [7][12]. Garzotto et al. [12] have proposed some patterns for designing adaptation strategies in Adaptable Educational Hypermedia Systems. These patterns define matching learning styles with application design solutions. Also, Cristea et al. [7] had proposed a taxonomy for AEH design patterns. However, they just identify different types of patterns according to a learning style and there were no reel formalization and no support for an automatic generation to a particular adaptation language. Moreover, they consider that one strategy can be expressed using one pattern ~~that~~ and this can be more difficult to reuse.

Open corpus adaptive systems

In the adaptive hypermedia community, research in the integration of open corpus content into the consideration of adaptive systems has been put in perspective for several years (mostly in the field of education [2]). Most of the existing systems are built upon an existing Adaptive Hypermedia System (e.g., [14] on top of [4]). Multiple issues are to be faced in order to develop open corpus based adaptive systems ([7][15][16]): automatic hypertext creation, indexation of open corpus resources, content preparation… But none of them face the problem of the definition of adaptation strategies, by a creator, in a simple and understandable way. The issue of adding knowledge to available resources (to describe them, to index them or to add metadata to them, depending on the community you belong to) is crucial. The adaptation process relies on the description of available resources. Manual indexing approaches ([14]) are very time-consuming but, the result is generally of good quality. And for some kinds of metadata (for example, the difficulty of educational resources), there is no other solution. Automatic indexing is cheaper and faster ([14]). Another approach is to exploit communities of users by taking into account the use of the resources by the AH users and then to attach information to these resources ([1]). In order to take advantage of both solutions, hybrid approaches ([18]) are being developed. Even if the description of the problem of available resources is not completely solved, we consider (for this work) that we can have access to the required metadata.

7. CONCLUSION AND FUTURE WORKS

In this paper, we propose a solution enabling the creator of an adaptive system to easily define adaptation strategies, which is not the case in most of the existing systems which are complex and not easy to understand.

Our approach relies on the creator to start the process. He has to select a set of elementary adaptation patterns from the catalog of elementary adaptation patterns. Then, the creator selects the elements of the domain model on which the system can automatically instantiate the elementary adaptation patterns. Obtained elementary adaptations can then be combined, in order to produce composed adaptations. The most difficult part of the composition process is done automatically. Either an elementary or a composed adaptation can be associated to a one-user characteristic. Hence, an adaptation strategy is defined by taking into account multiple- user characteristics.

We conducted a first validation using the GLAM platform. The validation showed that using elementary adaptation patterns, we can define all adaptations allowed in GLAM and the GLAM rules can be automatically generated from pattern-based adaptations.

Our future work, therefore, will be in the direction of proving that our system can be plugged in on top of multiple and different systems. This means that, using elementary adaptation patterns we can automatically generate adaptations in other adaptation languages, like LAG, AHA!.

We also plan to do scale-up testing. We are currently implementing a graphical tool allowing the creator to define adaptation strategies based on our approach and a set of elementary adaptation patterns.

8. REFERENCES

[1] Brusilovsky, P., Chavan, G., Farzan, R.: Social Adaptive Navigation Support for Open Corpus Electronic Textbooks. In Proceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004). P. De Bra and W. Nejdl (Editors), Springer, Lecture Notes in Computer Science 3137, pp 24-33.

[2] Brusilovsky, P.: Adaptive navigation support. In: Brusilovsky, P., Kobsa, A., Neidl, W. (eds.): The Adaptive Web: Methods and Strategies of Web Personalization. Berlin Heidelberg, USA, New York (2007), Springer, Lecture Notes in Computer Science, Vol. 4321. pp. 263-290.

[3] Brusilovsky, P., Henze, N.: Open Corpus Adaptive Educational Hypermedia. In: Brusilovsky, P., Kobsa, A., Neidl, W. (eds.): The Adaptive Web: Methods and Strategies of Web Personalization. Berlin Heidelberg, USA, New York (2007), Springer, Lecture Notes in Computer Science, Vol. 4321. pp. 672-696.

[4] Brusilovsky, P., Kobsa, A., Nejdl, W. (Eds.): The Adaptive Web, Methods and Strategies of Web Personalization. (2007) Lecture Notes in Computer Science, Vol. 4321 Springer

[5] Conlan, O., Wade, V., Evaluation of APeLS – An Adaptive eLearning Service based on the Multi-model, Metadata-driven Approach. In Proceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004). P. De Bra and W. Nejdl (Editors), Springer, Lecture Notes in Computer Science 3137, pp 291-295.

[6] Cristea, A.I., Calvi, L., The three Layers of Adaptation Granularity. In: the international Conference on User Modelling, USA, Pittsburgh (2003) Springer, pp. 145-155.

[7] Cristea, A., Garzotto, F.: designing patterns for adaptive or adaptable educational hypermedia: a taxonomy. In Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications, 2004, pp. 808-813.

[8] Cristea, A.I., Mooj, A.: LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators. In The Twelfth International World Wide Web Conference WWW'03, Budapest, Hungary (2003)

[9] De Bra, P., Smits, D., Stash, N.: Creating and Delivering Adaptive Courses with AHA! In: the first European Conference on Technology Enhanced Learning, Crete (2006) Springer, pp. 21-33.

[10] Danculovic, J., Rossi, G., schwabe, D., Miaton, L.: Patterns for Personalized Web Applications. In European Conf. Pattern Languages of Program- EuroPLoP (2001), pp. 34-43.

[11] Gamma, E., Helm, R., Johnson, R., Vlissides, J.M.: Design Patterns: Elements of Reusable Object-Oriented. In Software Addison-Wesley Professional Computing Series, 2003 (Hardcover)

[12] Garzotto, F., Retalis, S., Papasalouros, A., Siassiakos, K. : Patterns for designing adaptive/Adaptable educational hypermedia. In Journal (208) Advanced Technology for Learning (2004)

[13] Greenberg, J., Metadata Extraction and Harvesting: A Comparaison of Two Automatic Metadata Generation Applications. In the Journal of Internet Cataloging (2004) vol. 6(4), pp. 59-82.

[14] Henze, N., Nejdl, W., Adaptation in open corpus hypermedia. International Journal of Artificial Intelligence in Education 12,4, (2001), pp. 325-350.

[15] Jacquiot, C. Modélisation logique et générique des systèmes hypermédias adaptatifs. PhD thesis, Department of Computer science, France, Supelec (2006)

[16] Jacquiot, C., Bourda, Y,. Popineau, F., Delteil, A., Reynaud. C.: GLAM: A generic layered adaptation model for adaptive hypermedia systems. In: the 4th International, Germany, Heidelberg (2006) Springer, pp. 131–140.

[17] Knutov, E.,De Bra, P., Pechenizkiy, M. :AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques. In: New Review of Hypermedia and Multimedia, (2009) pp. 15-38.

[18] Levacher, K., Hynes, E. Lawless, S., O'Connor, A., Wade, V. A Framework for Content Preparation to Support Open-Corpus Adaptive Hypermedia. In International Workshop on Dynamic and Adaptive Hypertext at the 20th ACM Conference on Hypertext and Hypermedia, Hypertext 2009, pp. 13-24.

[19] Moore, A., Brailsford, T.J. Stewart, C.D.: Personally tailored teaching in WHURLE using conditional transclusion. In: the 12th ACM Hypertext Conference, Denmark, Arhus (2001), pp.163-164.

[20] Sluijs, K., Hidders, J., Leonardi, E., Houben, G.: GAL: A Generic Adaptation Language for describing Adaptive Hypermedia. In: 1st International Workshop on Dynamic and Adaptive Hypertext: Generic Frameworks, Approaches and Techniques, USA New York (2009) pp. 13-24.

[21] Steichen, B., Lawless, S., O'Connor, A., Wade, W.,: Dynamic Hypertext Generation for Reusing Open Corpus Content. In: the 20th ACM Conference on Hypertext and Hypermedia', USA, New York (2009) pp. 119-129.