RAPPORT DE RECHERCHE

L R I

EXPRESSING ADAPTATION STRATEGIES USING ADAPTATION PATTERNS

ZEMIRLINE N / BOURDA Y

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

01/2011

Rapport de Recherche N° 1540

CNRS – Université de Paris Sud
Centre d'Orsay
LABORATOIRE DE RECHERCHE EN INFORMATIQUE
Bâtiment 490
91405 ORSAY Cedex (France)

Expressing Adaptation Strategies using Adaptation Patterns

Nadjet Zemirline, Yolaine Bourda, *Member, IEEE,* Chantal Reynaud

Abstract—Today, there is a real challenge to enable personalized access to information. Several systems have been proposed to address this challenge including Adaptive Hypermedia Systems (AHSs). However, the specification of adaptation strategies remains a difficult task for creators of such systems. In this paper, we consider the problem of the definition of adaptation strategies at a high level. We present two main contributions: a typology of elementary adaptation patterns for the adaptation of navigation; and a process to generate adaptation strategies based on the use and the semi-automatic combination of patterns. We also describe how the generated adaptation strategies can be integrated into existing AHSs. A prototype has been implemented and an experiment in the e-learning domain has been conducted with a group of volunteers. This experiment shows that our pattern based approach for defining adaptation strategies is more suitable than those based on "traditional" AH languages.

Index Terms—AHSs, Adaptation strategies, Patterns.

1 Introduction

The concept of Adaptive Hypermedia Systems (AHSs) has existed for years now [21], and it has amply proved its utility particularly in education [6], [7], where students have access to personalized resources according to their knowledge, preferences and goals. However, till today, AHSs are not authored as many as desired, and this is mainly due to the difficulty of their authoring process [26].

In fact, authors have to define a *domain model* structuring available resources, a *user model* describing user characteristics and an *adaptation model* in the format understood by the used adaptation engine [14]. In this paper, we focus particularly on the authoring process of the adaptation model, which is most often the less intuitive part to be authored in an AHS by non technical persons.

Indeed, authors have to specify an adaptation model, in which they describe resources to propose for users having distinct characteristics and different knowledge in a personalized manner, in order to achieve their specific goals. This is done through the definition of multiple adaptation strategies. By an adaptation strategy, we mean that *an author specifies which resources have to be proposed and how they will be proposed to a set of users who share the same characteristics*. Thereby, authors of an AHS face numerous challenges when defining their adaptation strategies.

The first challenge concerns the expression of adaptation strategies. Multiple solutions have been proposed [13], [15] to make it easier, but they were related to a particular AHS and failed to answer the second and the third challenge, e.g, the author graph tool for AHA! [15] uses visualization in order to support creators and works only for AHA!.

The second challenge concerns the reuse of adaptation strategies from one system in another one, and the expression of adaptation strategies independently of any AHS. To do so, a new paradigm has been proposed: "write once, use many" [28]. This paradigm endorses expressing adaptation at a high level, independently of all AHSs and then translating this adaptation into a particular AHS. However, proposed adaptation languages [9], [26], [25] failed to answer the third challenge.

The third challenge concerns the granularity in writing adaptation strategies. It targets to avoid to write several times the common parts of adaptation strategies. To do so, adaptation languages using constructors have been proposed [9], [25], but till today, an adaptation strategy is considered as a whole block and can not be easily reused.

This paper addresses these three challenges. It concentrates on the ease of defining adaptation strategies at a fine granularity, and on the facility of reusing existing adaptation strategies. In a first time, we focus only on the expression of adaptation strategy for the adaptive navigation, where users are forced to navigate among the proposed navigation paths. This can be either by imposing them a particular order or by recommending them resources [21].

We perceive an adaptation strategy as a combination of elementary parts. Each part corresponds to an elementary adaptation and is bound to a user characteristic. A part can belong to different complex adaptation strategies depending on user characteristics. Our work takes up this idea. The notion of elementary adaptation patterns that we propose, is an abstraction of such elementary parts. Elementary adaptation patterns are independent from any application domain, but limited in

Nadjet Zemirline and Yolaine Bourda are with SUPELEC Systems Sciences (E3S) - Computer Science Department, France
Nadjet Zemirline and Chantal Reynaud are with the Universit Paris-Sud XI, CNRS (LRI) & INRIA - Saclay le-de-France / Projet Leo, France

a first time to express adaptive navigation. We propose a typology for the elementary adaptation patterns and a semi-automatic process to combine them (the most difficult part is done automatically).

The paper is organized as follows. It presents in section 2 close work on the expression of adaptation, and demonstrates the intuition of our work in section 3 with an example. Section 4 reviews the main aspects of our proposal. Section 5 presents the description of elementary adaptation patterns and their organization in a typology, and section 6 describes how elementary adaptation patterns can be used to define adaptation strategies. In section 7, we discuss how the generated adaptation strategies can be integrated on the top of existing AHSs. Finally, we conclude the paper in section 8.

2 Related works

Most often, during the authoring process of adaptation on domain and user models, authors ask themselves two questions [3]: *what kind of adaptation they can provide for users?* and *how to produce the desired adaptation?* The two questions are answered in that order. For deciding what kind of adaptation they can provide, authors may refer to existing typologies on adaptation (cf. Section 2.1), while for producing adaptation, they need to consider what are the most appropriate adaptation engine and the languages understood by each of them (cf. Section 2.2).

On the other hand, as there are more and more resources available on the web, recent works enable authors not only to define adaptation on their sets of resources but also on those available on the web. So, we present works about integrating adaptive technologies on open corpus (cf. Section 2.3).

2.1 What kind of adaptation could be provided?

The well-known Brusilovsky taxonomy [4] is undoubtedly the most used typology of adaptation. It describes several methods of adaptation that can be combined together. These methods are organized into three non disjoint groups: adaptive presentation, content adaptation and adaptive navigation support. This typology relies on the fact that the available resources can be modified and restructured during the adaptation process. Hence, it is not suitable when there is no control of the distributed resources.

As we focus in this paper on the expression of adaptive navigation, we get a particular interest of methods included in the adaptive navigation support group. The group includes 4 methods:

- *direct guidance*: supervises users step by step. It is done by proposing to users one link at a time.
- *adaptive ordering*: defines the priority of all the links of a particular page.
- *link hiding and removal*: hides, removes or disables links to users (e.g, AHA! [15] hides links that are not relevant to users).
- *adaptive link annotation*: suggests links to users. The suggestions are often expressed using visual cues (e.g, WHURLE [24] makes suggestions using colours).
- *link generation*: creates new links on a page.

2.2 How authors can express their adaptation?

We have grouped existing solutions to express adaptation in three main categories.

Adaptation languages accompanied by their adaptation engine. Adaptation strategies written by these adaptation languages are often expressed in condition-action or event-condition-action rules [15], [24], [22]. However, authoring adaptation using rules is not easy to perform and is time consuming. Thereby, aids have been proposed to make the expression of adaptation easier. E.g, the *author graph tool* for AHA! [15] which uses visualization in order to support authors: for each new created concept, the tool associates a set of attributes and adaptation rules. Regardless, authors are captive to a particular system. Indeed, adaptation strategies expressed in a system cannot be used outside this system, it has to be rewritten.

Generic adaptation languages accompanied by translators to existing adaptation engines. Some generic languages (independent of any system) have been proposed to specify adaptation [9], [25]. Among them, the LAG language [9], which is an implementation of the specification of the adaptation language defined in the LAOS model [10]. It includes conversion to the WHURLE [24], Blackboard [1] and AHA! adaptation engines. However, LAG is like a programming language, which is not very suitable for non technical authors (an example is given in section 3). Recently a new Generic Adaptation Language (GAL[1]) has been developed to describe adaptive hypermedia [25]. It argues to gather all functionalities of existing adaptation engines and to be an intermediate language between existing authoring environments and adaptation engines. For that, GAL plans to include translators from existing authoring environments to GAL and from GAL to existing adaptation engines. It describes the navigational structure of a web application using abstract constructs (e.g. units, attributes). But, the description of adaptation remains difficult to specify, as authors have to write a GAL program (use of SPARQL[2]queries to select resources) in a sequential way and no aid is proposed for them. Furthermore, generated adaptation strategies by these adaptation languages are considered as a whole block and can not be easily reused.

Hypertext and adaptation patterns. Some design patterns for expressing personalization in web applications have been proposed [16], based on commonly used design structures. They are suitable for designers of adaptive systems but not for authors of authors of a

1. GAL is proposed in the context of the GRAPPLE project http://www.grapple-project.org/

2. www.w3.org/TR/rdf-sparql-query/

particular adaptive hypermedia course. Such adaptation design patterns have been proposed in the e-learning domain [12], [19]. Garzotto et al. [19] have proposed patterns corresponding to learning styles. Cristea et al. [12] have proposed a taxonomy of various AEHS (Adaptive Educational Hypermedia Systems) design patterns according to different learning styles. There is no real formalization and no support for an automatic export to a particular adaptation language. One adaptation strategy (as complex as it can be defined by authors) is expressed using only one pattern. Patterns can not be neither combined together or modified, i.e, authors have to find a pattern corresponding to their desired adaptation strategy, otherwise, they can not express it.

2.3 Open corpus adaptive systems

In the AH community, research concerning the integration of open corpus content into adaptive systems has been under scrutiny for several years - mostly in the field of education [3]. Most of the existing systems are built upon an existing AHS (e.g., [9] on top of [15]). Multiple issues are to be faced in order to develop open corpus based adaptive systems ([12], [20]), including automatic hypertext creation, indexing of open corpus resources and content preparation. None of these systems face the problem of the definition of adaptation, by an AHS author, in a simple way.

2.4 Conclusion

We have discussed here solutions helping authors to find what adaptation they can propose and how they can express it. However, till now, there are no works concerning building complex adaptation strategies, independent of any system by combining simple adaptations. In this paper, we focus on this specific point. Adaptation strategies must be defined at a fine granularity. Our aim is thus to help authors defining their own adaptations, independently of any adaptation engine, at a higher level and in an easy manner. In the next section, we introduce a use case giving the intuition of our contribution. This scenario is subsequently used in the paper.

3 Motivation, use case

Assume that Jane who is a lecturer in computer science wants to build an adaptive course from her materials, i.e., Jane is going to author an AEHS. She has first to define a domain model, then to describe the characteristics of her students in a user model, and finally to express the desired adaptation.

Jane proposes a domain model in UML (cf. Figure 1), in-which she considers the addressed notions as instances of the class *Concept*[3]. The concepts must be learnt in a particular order, that is defined through the relation

3. In this paper, names of classes have the first letter in upper-case and are in italic, and name of instances have the same name as the class for which they belong in lower-case

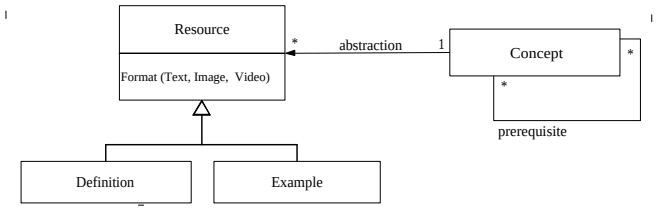

Fig. 1. Jane's domain model

pre-requisite. Each concept may be trained using definitions or examples. *Definition* and *Example* are subclasses of the class *Resource*, i.e. each of their instances has a content, which can be proposed to students. Furthermore, each resource may be in different *formats*: *text*, *image* or *video*.

Jane considers the following student characteristics

- learning mode: *in-depth* learning mode means that each subject must be known in-depth before going to a related subject. *In-breadth* learning mode means that a student has to know a variety of subjects before going in-depth.
- reasoning mode: an *inductive* reasoning mode means that the student has access to examples before the related definitions are presented to him. In a *deductive* reasoning mode definitions precede examples.
- presentation form: a *verbal* presentation form is for students preferring textual resources and an *audio* presentation form is for those preferring audio resources.

Among the adaptation strategies Jane wants to propose, we are going to focus on the adaptation strategy $S1$. It concerns students whose learning mode is *in-depth*, with an *inductive* reasoning mode and preferring *audio* resources. $S1$ proposes resources that are examples before those which are definitions. They will be in an *audio* format if that one is available otherwise in a *textual* format. They will be related to concepts ordered according to a depth-first navigational path using the relation *pre-requisite*.

Jane can express $S1$ using solutions supported by her AH system. However, they are not easy to implement and require good backgrounds. See as an illustration, the implementation of $S1$ using GLAM in figure 2 (for GLAM syntax see section 7.2.1), and using LAG in figure 3. This implies that Jane has already her domain and user models in the format understood by the used AH system.

Naturally, Jane expressed $S1$ in three parts: 1) $S1$ concerns students whose learning mode is in-depth, 2) $S1$ concerns students with an inductive reasoning mode, 3) $S1$ concerns students preferring audio resources. These parts can be considered independently of one another and may compose the definition of other adaptation strategies, for example $S2$, an adaptation strategy for students whose learning mode is *in-depth*, with an *inductive* reasoning mode and preferring textual resources. $S2$ differs from $S1$ only in proposing resources in a

S1 is defined in three steps:
Step1: defining GLAM rules

R1 *type (r, Example) ∧ format (r, audio) ∧ abstraction(r, Concept1) ∧ abstraction(currentR, Concept2) ∧ pre-requisite (Concept2, Concept1) ∧ pre-requisite (Concept1, goal) → Read(r, degree)*
R2 *type (r, Example) ∧ format (r, text) ∧ abstraction(r, Concept1) ∧ abstraction(currentR, Concept2) ∧ pre-requisite (Concept2, Concept1) ∧ pre-requisite (Concept1, goal) → Read(r, degree)*
R3 *type (r, Definition) ∧ format (r, audio) ∧ abstraction(r, Concept1) ∧ abstraction(currentR, Concept2) ∧ pre-requisite (Concept2, Concept1) ∧ pre-requisite (Concept1, goal) → Read(r, degree)*
R4 *type (r, Definition) ∧ format (r, text) ∧ abstraction(r, Concept1) ∧ abstraction(currentR, Concept2) ∧ pre-requisite (Concept2, Concept1) ∧ pre-requisite (Concept1, goal) → Read(r, degree)*
R5 *type (r, Example) ∧ format (r, audio) ∧ abstraction(r, Concept1) ∧ pre-requisite (Concept1, goal) → Read(r, degree)*
R6 *type (r, Example) ∧ format (r, text) ∧ abstraction(r, Concept1) ∧ pre-requisite (Concept1, goal) → Read(r, degree)*
R7 *type (r, Definition) ∧ format (r, audio) ∧ abstraction(r, Concept1) ∧ pre-requisite (Concept1, goal) → Read(r, degree)*
R8 *type (r, Definition) ∧ format (r, text) ∧ abstraction(r, Concept1) ∧ pre-requisite (Concept1, goal) → Read(r, degree)*

R1 proposes audio examples according to a depth-first navigational path on their concepts using the relation *pre-requisite*), and these concepts enable to reach the goal[a].
R5 proposes audio examples that are linked to concepts which enable to reach the goal.
Step2: defining associations between rules and user characteristics: γ(audio) is associated to R1, R3, R5, R7, γ(text) is associated to R2, R4, R6, R8, γ(Definition) is associated to R3, R4, R7, R8, γ(Example) is associated to R1, R2, R5, R6, and γ(depth-first) is associated to R1, R2, R3, R4, R5, R6, R7, R8.
Step3: defining GLAM meta-rules

MR1 *γ(audio) is preferred on γ(text)*
MR2 *γ(Example) before γ(Definition)*
MR3 R1 before R5
MR4 R2 before R6
MR5 R3 before R7
MR6 R4 before R8

(MR1 means that the rules associated to the characteristic presentation form audio are executed, when they return no results, the rules associated to the characteristic presentation form text are executed).

a. In this paper, the goal to be reached by users (learners for this example) is modelled as a property

Fig. 2. Jane's $S1$ in the GLAM format

```
initialization (
      while true (
            PM.GM.Concept.show = false
            UM.Concept.defAudio = false )
      while ( enough(GM.Concept.type == Example
        GM.Concept.label == audio, 2))
            do (PM.GM.Concept.show = true )
)
implementation (
      if enough(PM.GM.Concept.access == true
        GM.Concept.type == Definition , 2)
            then (PM.GM.Concept.show = true
            UM.Concept.defAudio = true)
      if enough (PM.GM.Concept.Parent.access == true
        UM.Concept.defAudio == true
        GM.Concept.type == Example , 3)
            then (PM.DM.Concept.show = true)
)
```

Fig. 3. Jane's $S1$ in the LAG format

textual format if that one is available otherwise in an *audio* format.

To enable Jane easily define her strategies, i.e. the most natural way as possible, we offer the possibility to specify each part of a strategy by defining the set of resources to propose and the order in which they have to be proposed. According to this approach, $S1$ will be built from the following parts:

S1-1 presents resources linked to the domain concepts ordered according to a depth-first navigational path using the *pre-requisite* relation.
S1-2 presents only *audio* resources if they are available otherwise presents textual resources.
S1-3 presents examples before definitions.

The adaptation strategy $S1$ is intended to students with specific characteristics. Therefore, each part of the strategy has to be labelled by a student characteristic, i.e. *S1-1*, for example, will be defined for in-depth learning mode students. Thereby, to define $S2$, Jane can reuse the parts *S1-1* and *S1-3*, she has only to define the part *S2-2* for the *textual* presentation form.

We presented here the intuition of our contribution according to Jane's needs, in the following, we describe our approach in a more general way.

4 Main Aspects of Our Framework

We propose the EAP framework in which authors have a clear separation between what kind of adaptation strategies they want to provide to users and the technicalities involved in writing it. The idea is to help authors in selecting the adaptation strategy and then generated it in a semi-automatic way. Defined adaptation strategies are described at a high level and independently of any adaptation engine.

The EAP framework focuses only on the expression of adaptation strategies. So, it assumes authors have already created their domain and user models. Furthermore, our framework is based on design patterns [18]. Design patterns describe recurrent solutions to common problems in software design. The solutions are generic and cannot be directly translated to code. For e.g, the Object-oriented design patterns describe relationships and interactions between classes or objects, without specifying the final application classes or objects that are involved. In practice, design patterns can speed up the development process by providing tested, proven development paradigms. We argue that an adaptation strategy is a kind of conception, where authors have to write several times the same parts of an adaptation strategy, sometimes on different elements. Consequently, the proposed framework uses a set of building blocks independent from any application domain, called *elementary adaptation patterns*, which are based on design patterns. Thereby they can be used and instantiated to define specific adaptation strategies.

The main steps for authoring an adaptation strategy with the EAP framework are:

1) *Selection*. The author either selects elementary adaptation patterns (those needed to define his adaptation strategy) and instantiates them on his own model (thereby, elementary adaptations are defined), or reuses existing elementary adaptations.
2) *Specification*. The creator specifies associations between user characteristics and elementary adaptations.
3) *Computation*. The computation of the adaptation strategy resulting from step 2 is automatic.

We have defined a typology and a library of elementary adaptation patterns that can be selected for use within an adaptation strategy, and which we introduce in section 5. The instantiation process and the combination process are described in section 6.

Before going further, let us apply the EAP framework on Jane's use case. Jane needs now:

1) To define *S1-1* (resp. *S1-2*, *S1-3*) by instantiating the appropriate elementary adaptation pattern on the relation *pre-requisite* (resp. on the classes *Example, Definition*, on the property *format*).
2) To associate *S1-1* with *in-depth* learning mode, *S1-2* with *inductive* reasoning mode, *S1-3* with *audio* presentation form.
3) As $S1$ is for students with an *in-depth* learning mode, an *inductive* reasoning mode and who want *audio* resources. $S1$ is thus automatically built by combining *S1-1*, *S1-2*, *S1-3*.

Note that, that way, Jane does not need to worry about technical problems in the expression of $S1$.

5 Elementary adaptation patterns

The notion of elementary adaptation patterns that we propose, is an abstraction of Jane parts. Furthermore, we defined our elementary adaptation patterns in a manner that is independent from any application domain in order to be able to cover other authors parts. Thereby, the criteria used to define our elementary adaptation patterns are defined in a generic way (cf. Section 5.1). Elementary adaptation patterns are described in section 5.2, and their typology is defined in section 5.3.

5.1 Fundamental criteria for defining elementary adaptation patterns

As each part of the $S1$ strategy defined by Jane, an elementary adaptation pattern targets a set of resources of a particular type to be presented and also specifies the order in which they will be proposed. This section presents exhaustive criteria to select resources (cf. Section 5.1.1) and to organize the selected resources (cf. Section 5.1.2).

5.1.1 Criteria used to select resources

Criteria used to select resources are based on the domain model, where resources are structured and described. We argue that the general description of a domain model includes the following elements:

- *a set of classes*. This set must contain the class representing all the resources to be proposed to users which we have called *Resource*, and the class representing all the domain concepts, which we have called *Concept*.
- *a set of relations between classes*. Each relation defines a graph on instances of classes on which it is defined. This graph can be navigated according to two different navigational paths in order to reach the goals: depth-first or breadth first.
- *a set of properties*.

Thereby, we have differentiated between criteria selecting resources and criteria defining a navigational path on relations. Our criteria for selecting resources are: their belonging to a class, the values of some properties, or the presence of a relation that defines a navigational path through the resources or the concepts graph. Furthermore, our criteria currently considered for defining a navigational path are either depth-first, breadth-first or random.

5.1.2 Criteria used to order the selected resources

We have looked over works defining adaptation methods, by giving a particular interest for adaptive navigation, without mattering if the methods are applied on a set of links to resources or resources themselves.

We have looked over the Brusilovsky typology (cf. Section 2.1) excluding methods of the adaptive navigation support group which modify resources (e.g, hiding links belonging to content of resources). Only *direct guidance*, *adaptive ordering* and *adaptive link annotation* have been considered.

We have also looked over the classification of external actions in AHS defined by Stash and al. [26]. The classification includes *actions on items* (e.g, selection, showing items or links to items), *actions on a set of items* (e.g, ordering), *hierarchical actions* (e.g, action on parent or child) and *actions on the overall environment* (e.g, changing the layout). We only consider the actions having impact on the navigation of users, this includes: *actions on items*, *actions on a set of items* and *hierarchical actions*. Furthermore, we distinguish between actions and elements on which the actions are performed. The elements can be an item, a set of item, parents or children. So, we only consider the *selection*, *show* and *order* actions.

On the other hand, we have looked over AHS implementing adaptive navigation like AHA! [15], WHURLE [24], GLAM [22] etc. We found that GLAM implements a kind of adaptation not mentioned elsewhere. This adaptation proposes alternative resources if the desired resources are not available. We find it interesting and have retained it in our own typology.

From this study, we conclude that there are four basic modes to select resources in a setting of adaptive navigation support. These modes are the following:

- *Selection only*: provides a set of resources, which are all proposed to the user, i.e, only the selected resources are proposed to users, the other resources are not proposed. This selection mode is equivalent to the combination of the *selection and show actions* described by Stash et al. In fact, Stash et al. propose to select and show selected resources in two separate processes, while in our approach implicitly all selected resources are shown. There is no equivalent in the Brusilovsky typology.
- *Recommended selection*: provides multiple sets of resources (at least two) that include knowledge to specify which set should be recommended rather than the other (sets of) resources. for example, we can recommend definitions rather than examples. The user can access both types of resource, but a typographic indication enables the user to identify which resources are recommended. It is equivalent to the *adaptive link annotation* described by Brusilovsky, but there is no equivalent in the actions described by Stash et al.
- *Ordered selection*: provides multiple sets of resources (at least two), accompanied with knowledge to specify the order in which they must be presented. Only one set of resources is proposed at a time, and the resources of a particular set are not proposed until all the resources of all sets of higher priority have been viewed by the user. E.g, in e-learning, concepts can be selected and ordered using the pre-requisite relation defined between concepts. It is equivalent to the *adaptive ordering* described by Brusilovsky and to the *direct guidance* described by Brusilovsky when the returned result includes only one resource in each set. It is also equivalent to the combination of the *order and show* actions described by Stash et al..
- *Alternate selection*: provides multiple sets of resources (at least two), accompanied with data that specifies the order in which they must be presented, knowing that only one set is presented to the user. E.g, we propose textual resources when they are available, and audio resources in the absence of textual resources. Neither Brusilosvky or Stash et al. has considered this selection mode.

5.2 Description of elementary adaptation patterns

We propose the following definition for elementary adaptation patterns, based on the definition of design patterns [18].

Definition 1: An elementary adaptation pattern describes a generic solution for a generic elementary adaptation problem.

This solution is independent from any language, and it exploits the characteristics of the domain model.

Definition 2: A generic elementary adaptation problem describes a criterion to select resources to be proposed and a criterion to define in which order the selected resources are going to be proposed.

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.
Solution: the solution includes two elements:
Expressions: denote a set of resources to be proposed to the user, and the conditions which have 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. The formal description of expressions may be accompanied by an informal description.
Meta-expressions: a binary relation between two expressions. Indeed, when using multiple expressions, we specify the way they have to be considered by using meta-expressions. The formal description of meta-expressions may be accompanied by an informal description.
Constituents: describe the elements of the domain model used in the expressions described in the solution pattern.

Fig. 4. Description of elementary adaptation patterns

We define in the figure 4 the characteristics retained from [18] and used to describe elementary adaptation patterns.

The solution part is the most formal part of the elementary adaptation patterns. We have defined a grammar using the Extended Backus-Naur Form (EBNF) [29]. The grammar is described in the figure 5. It includes a set of non-terminal elements expressed between brackets, and a set of terminal elements expressed between coats. For people not familiar with EBNF syntax, we give examples of the solution part respecting the proposed grammar (cf. Figure 8, 9, 10). These examples are also accompanied by an informal description.

We give an informal description of the semantic of the language defined by the grammar and some associated constraints. In order to do so, we consider a domain model *DM*, composed of:

- *Cls* = {c/ c is a class}
- *Rel* = {rel/ rel is a relation}
- *Prop* = {p/ p is a property}
- Val_p = {v/ v is a value of the property p}
- *Res* = {r/ r is a resource}

We defined general elements, which we describe in figure 6. Furthermore, we defined predicates to facilitate the selection of either resources or concepts. These predicate are:

- *instanceOf*: *instanceOf(r, c)* is true, for all resources *r* that are instances of the class *c*.
- *characteristicOf*: *characteristicOf(r, p, op, v)* is true, for all resources *r* having the property *p* and satisfying the comparison test using the operator *op* and the value *v*.
- *linked*: *linked(i1, i2, rel)* is true, for all instances *i1* that are linked directly to the instance *i2* by the relation *rel*.
- *linked-transitive*: *linked-transitive(i1, i2, rel)* is true, for all instances *i1* that are linked directly or indirectly to the instance *i2* by the relation *rel*.

⟨*Solution*⟩ ::= ⟨*Expressions*⟩ ⟨*Meta-Expressions*⟩.

⟨*Expressions*⟩ ::= (⟨*Expression*$_{rel}$⟩)*| (⟨*Expression*$_{prop}$⟩)*| (⟨*Expression*$_{cls}$⟩)*.
⟨*Meta-expressions*⟩ ::= (⟨*Id*⟩ "$\prec$" ⟨*Id*⟩)* | (⟨*Id*⟩ "$\uplus$" ⟨*Id*⟩)* | (⟨*Id*⟩ "|" ⟨*Id*⟩)*.

⟨*Expression*$_{rel}$⟩ ::= ⟨*Id*⟩ ":"⟨*Exp*$_{rel}$⟩ ("$\wedge$" ⟨*Exp*$_{rel}$⟩)*.
⟨*Expression*$_{prop}$⟩ ::= ⟨*Id*⟩ ":"⟨*Exp*$_{prop}$⟩.
⟨*Expression*$_{cls}$⟩ ::= ⟨*Id*⟩ ":"⟨*Exp*$_{cls}$⟩.

⟨*Exp*$_{rel}$⟩ ::= linked "("⟨*Inst*⟩","⟨*Inst*⟩ ","⟨*Rel*⟩ ")" | linked-transitive "("⟨*Inst*⟩","⟨*Inst*⟩ ","⟨*Rel*⟩ ")" | distance "("⟨*Inst*⟩","⟨*Inst*⟩ ","⟨*Rel*⟩ ","⟨*Number*⟩")".

⟨*Exp*$_{prop}$⟩ ::= characteristicOf "("⟨*Res*⟩","⟨*Prop*⟩ ","⟨*Operator*⟩ ","⟨*Val*⟩")" .
⟨*Operator*⟩ ::= "=" |"$\neq$" | "$\leq$" | "$\geq$".

⟨*Exp*$_{cls}$⟩ ::= instanceOf "(" ⟨*Res*⟩","⟨*Cls*⟩ ")" .

⟨*Id*⟩ ::= ⟨*String*⟩.

⟨*Cls*⟩ ::= "c"⟨*Number*⟩ .
⟨*Inst*⟩ ::= "concept"⟨*Number*⟩ | ⟨*Res*⟩.

⟨*Res*⟩ ::= "resource"⟨*Number*⟩ .

⟨*Rel*⟩ ::= "r"⟨*Number*⟩ .
⟨*Prop*⟩ ::= "p"⟨*Number*⟩ .
⟨*Val*⟩ ::= (⟨*String*⟩ — ⟨*Number*⟩)+.
⟨*String*⟩ ::= ["a"-"z"] ⟨*String*⟩ * .
⟨*Number*⟩ ::= ["0"-"9"] ⟨*Number*⟩ * .

Fig. 5. Syntax of the characteristic *Solution*

Elements	Variable referring to
<*Number*>	any integer number
<*String*>	any string
<*Id*>	identifiers. Identifiers belonging to the same solution part have to be different
<*Res*>	a resource
<*Inst*>	either a concept or a resource
<*Cls*>	a class of *DM*
<*Rel*>	a relation of *DM*
<*Prop*>	a property of *DM*
<*Val*>	a value among the allowed values for the used property

Fig. 6. Description of general elements

- *distance*: *distance(i1, i2, rel, n)* is true, for all instances *i1* that are distant from the instance *i2* by *n* instances using the relation *rel*.

These predicates compose 3 types of expressions:

- $<Exp_{cls}>$ for expressions on classes.
- $<Exp_{prop}>$ for expressions on properties. Expressions belonging to the same solution part have to be expressed on the same property.
- $<Exp_{rel}>$ for expressions on relations. When the expression includes multiple selections, the variables indicating the selected resources have to be the same.

When more than one expression is defined in a solution, meta-expressions must be defined between all expressions of the solution. This is done using the expression identifiers. Each identifier used in the definition of a meta-expression must correspond to an expression identifier. Three types of meta-expressions are proposed. They are:

- <*Id1*> $\prec$ <*Id2*> means that the set of resources selected with the expression identified by *Id1* is proposed before the set of resources selected with the expression identified by *Id2*.
- <*Id1*> $\uplus$ <*Id2*> means that the set of resources selected with the expression identified by *Id1* is recommended rather than the set of resources selected with the expression identified by *Id2*. A typographic indication can be used to differentiate between the set of resources recommended from those they are not.
- <*Id1*> | <*Id2*> means that the set of resources selected with the expression identified by *Id2* is an alternative of the set of resources selected with the expression identified by *Id1*.

5.3 Typology of elementary adaptation patterns

We have defined a library of 22 elementary adaptation patterns using the criteria defined in section 5.1. An elementary adaptation pattern is based simultaneously on (1) one of the 4 selection modes of resources to be proposed, (2) one of the 3 elements of the domain model involved in the selection process and when the element is a relation, we consider also (3) one of the 2 types of navigation through the resources or the concepts graph. The two navigation modes are applied for all the selection modes except for the *selection only* mode, which proposes a set of resources according to a particular criterion.

In order to be able to look easily over the defined elementary adaptation patterns, we have organized them in a tree where each leaf is an elementary adaptation pattern (cf. Figure 7). The tree represents our typology.

Let us now use this typology to help Jane to define $S1$. We note that each part of $S1$ can be defined thanks to a pattern. The pattern *P2.1.1.1* (cf. Figure 8) is used to define *S1-1* (*S1-1* consists of ordering concepts according to a depth-first navigational path using the relation prerequisite, and presents resources linked to these concepts), *P3.3* (cf. Figure 9) is used to define *S1-2* (*S1-2* consists of presenting only audio resources if they are available otherwise presents textual resources), and *P2.2* (cf. Figure 10) is used to define *S3-3* (*S3-3* consists of presenting examples before definitions).

After having described the typology and some elementary adaptation patterns, lets come back to the process of defining adaptation strategies.

6 Defining Adaptation Strategies

This sections focuses on the steps 1 and 3 of the authoring steps of adaptation strategies (cf. Section 4).

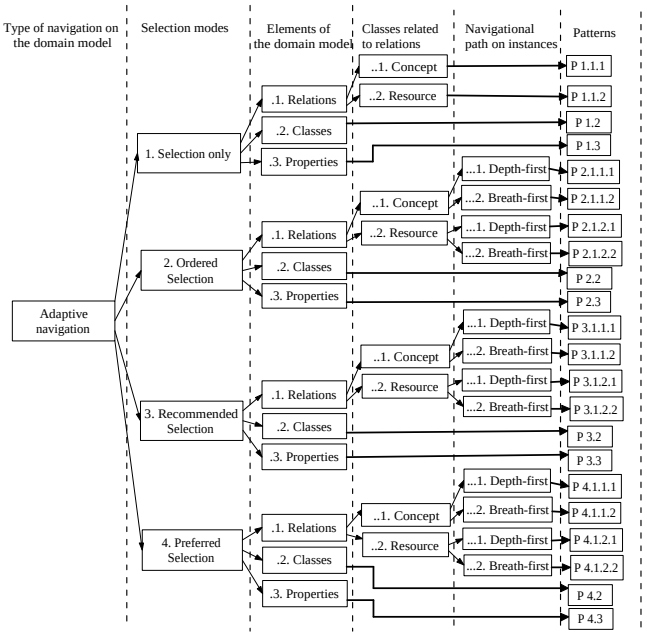

Fig. 7. Typology of elementary adaptation patterns

Name: Ordered Selection - Depth first- Relation - Concept
Intent: This pattern proposes resources according to a depth first navigational path on concepts.
Solution:
Expression
E_1: *linked(rCurrent, concept', abstraction)* $\wedge$ *linked-transitive(concept, goal,* $relation_i$*)* $\wedge$ *linked(r, concept, abstraction)* $\wedge$ *linked(concept, concept',* $relation_i$*)*
E_2: *linked-transitive(concept, goal,* $relation_i$*)* $\wedge$ *linked(r, concept, abstraction)*
According to E_1: selected resources are linked to concepts using $abstraction$. these concepts can reach the $goal$ using $relation_i$ and are directly linked to the current concept. According to E_2: selected resources are linked to concepts using $abstraction$, these concepts can reach the $goal$ using $relation_i$.
Meta-expressions
$E_1 \prec E_2$
According to this meta-expression, the set of resources selected by E_1 is proposed before the set of resources selected by E_2.
Constituents:
concept: a variable describing an instance of the class *Concept*.
rCurrent: a variable describing the current instance proposed to users of the class *Resource* or of one of its specializations.
goal: a variable describing the goal to reach, which is an instance of the class *Concept*.
r: a variable describing an instance of the class *Resource* or of one of its specializations.
$relation_i$: a variable describing a relation defined between instances of the class *Concept*.
abstraction: a variable describing a relation defined between an instance of the class *Concept* and one or more instances of the class *Resource* or of one of its specializations.

Fig. 8. OrderedSelection-DepthFirst-Relation-Concept

Name: Ordered Selection - Classes
Intent: This pattern proposes ordered resources belonging only to subclasses of the class Resource.
Solution:
Expressions
E_1: *instanceOf (r,* $Class_1$*)*
...
E_n: *instanceOf (r,* $Class_n$*)*
According to Ei: selected resources are instances of the class $Class_i$
Meta-expressions
$E_i \prec E_j$, i < j, i = 1..n and j = 1..n.
According to this meta-expression, the set of resources selected by E_i is proposed before the set of resources selected by E_j (i < j).
Constituents:
r: a variable describing an instance of the class *Resource* or of one of its specializations.
Class$_i$: a variable describing a subclass of the class *Resource*.

Fig. 9. Ordered Selection-Classes

Name: Alternate Selection - Properties
Intent: This pattern proposes resources that satisfy particular values of the property $property_i$, if no resources are available, other resources that satisfy other values of the property $property_i$ will be proposed.
Solution:
Expressions
E_1: *characteristicOf(r,* $property_i$*, op,* val_1*)*
....
E_n: *characteristicOf(r,* $property_i$ *, op,* val_n*)*
According to Ei: selected resources have the property $property_i$ and their value must satisfy the comparison test.
Meta-expressions
$E_i \mid E_j$, i < j, i = 1..n and j = 1..n.
According to this meta-expression, the set of resources selected by E_j is an alternative of the set of resources selected by E_i (i < j).
Constituents:
r: a variable describing an instance of the class *Resource* or of one of its specializations.
property$_i$: a variable describing a property of the class *Resource*.
val: a variable describing a possible value for the property $property_i$.

Fig. 10. Alternate Selection-Properties

The step2 is simple and we do not give further details. We start first by describing the step1 related to the instantiation process of elementary adaptation patterns (cf. Section 6.1). Then, we detail the step3 related to the combination process (cf. Section 6.2), and we end by using the EAP framework to define Jane's adaptation strategy $S1$ (cf. Section 6.3).

6.1 Defining elementary adaptations

In order to propose a generic solution, elementary adaptation patterns are defined on a generic domain model. Consequently, when authors select an elementary adaptation pattern, they have to instantiate its constituents on their personal domain model, in order to obtain the elementary adaptation that meets their needs. We define elementary adaptations as follows:

Definition 3: An elementary adaptation is obtained after an instantiation of an elementary adaptation pattern on a

Name: Ordered Selection-Example-Definition
Intent: This elementary adaptation proposes ordered resources belonging only to *Example* and *Definition* in this order.
Solution:
Expressions
E_1: *instanceOf (r, Example)*
E_2: *instanceOf (r, Definition)*
According to E_1, selected resources are instances of the class *Example*, and according to E_2, selected resources are instances of the class *Definition*.
Meta-expressions
$E_1 \prec E_2$
According to this meta-expression, all examples are proposed before all the definitions.
Constituents:
r: a variable which represents an instance of the class Resource or of one of its specializations.
Example: a variable which represents the class *Example*, a subclass of the class *Resource*.
Definition: a variable which represents the class *Definition*, a subclass of the class *Resource*.

Fig. 11. The elementary adaptation *S1-3*

particular domain model.

Elementary adaptations has therefore the same structure as elementary adaptation patterns. The generation of an elementary adaptation is done in a semi-automatic way: the characteristics *Name*, *Intent* are generated in a semi-automatic way and the characteristics *Solution* and *Constituents* are automatically generated.

For example, when Jane wants to express the *S1-3* part, she selects the pattern *P2.2* and instantiates it with the classes *Example* and *Definition* (for informal description, see section 3, for formal description see figure 11).

Following this principle, authors can define several elementary adaptations, each of them being associated to one user characteristic (step2 in section 4). When users have a profile composed of several characteristics, complex adaptation strategies have to be defined. They are obtained by combining elementary adaptations, each one being associated with a component of the user profile. This combination process is detailed in the next section.

6.2 Combining elementary adaptations

Combining elementary adaptations together defines a combined adaptation.

Definition 4: A combined adaptation defines a set of resources that satisfy simultaneously all constraints imposed by multiple elementary adaptations.

A combined adaptation has the same characteristics and is structurally identical to an elementary adaptation. Concretely, the combination process of a set of elementary adaptations consists in combining their characteristics together. A manual process is used to combine the characteristics *Name* and *Intent* as it needs natural language processing (not detailed in this paper). We propose an automatic process to combine the characteristics *Solution* and *Constituents* which is explained further below.

The combination of the characteristic *Constituents* is simple. Constituents coming from the different adaptations are gathered together into a set of constituents. However, the combination of the characteristic *Solution* is more complex and we have defined the following process.

We have chosen to base the process on criteria concerning the selection of resources, as our final aim is to propose a set of resources. Thereby, we have criteria based on: classes to which a resource belongs, properties satisfied by a resource, and relations in which a resource participate. We have exploited these criteria in the combination process of the characteristic Solution. We express this process in two sequential steps:

1) Build different sets of identifiers of expressions, one set for each different criterion (cf. Section 6.2.1).
2) Build one adaptation from the sets built in step 1 (cf. Section 6.2.2).

6.2.1 Step one of the combination

Let $Sol_1, Sol_2, ..., Sol_n$ be the solution part of the elementary adaptations to combine, where each Sol_i is composed of:

- n_i expressions noted E_i, each expression having an identifier Id_i.
- m_i meta-expressions noted ME_i.

We group the identifiers whose expressions are expressed on one given criteria in different sets.

- the identifiers whose expressions exploit classes are put in the same set Set_{cls} = {Id_i/ Id_i is an identifier that denotes an expression exploiting classes}.
- the identifiers whose expressions exploit relations are grouped into sets, one set per relation. Set_{rel} = {Id_j/ Id_j is an identifier that denotes an expression exploiting the relation *rel*}.
- the identifiers whose expressions exploit properties are grouped into sets, one set per property. Set_{prop} = {Id_j/ Id_j is an identifier that denotes an expression exploiting the property *prop*}.

where each $Id_i \in \text{Sol}_i$ belongs only to one set, either to the Set_{cls}, to a set of {Set_{rel}}, or to a set of {Set_{prop}}.

6.2.2 Step two of the combination

Let $Set_1, Set_2, ..., Set_p$ be the sets of identifiers obtained after the first step, let Sol_c be the solution resulting from the second step of the combination process composed of:

- n_j expressions noted CE_c.
- m_j meta-expressions noted CME_c.

Let Set_c be the set of p tuples built as follows:

$$Set_c = Set_1 X Set_2 X ... X Set_p$$

For each tuple, a distinct identifier is defined and is associated to an expression CE_c:

$$CE_c = E_1 \wedge E_2 ... \wedge E_p$$

where

- CE_c is the expression belonging to Sol_c.
- E_i is the expression whose identifier is Id_i, and $Id_i \in Sol_i$, *i = 1...p*.

Identifiers are also used to associate knowledge to expressions. This results in defining meta-expressions. Defining meta-expressions on the expressions E_c of the solution Sol_c is done as follows.

Let CE_i and CE_j be two expressions belonging to the solution Sol_c, where CE_i (resp. CE_j) contains E_1 (resp. E_2), E_1 and E_2 belong to the same solution, and are linked by the meta-expression Id_1 M_h Id_2 (Id_1 (resp. Id_2) is the identifier of E_1 (resp. E_2)). In that case, we deduce the meta-expression Id_i M_h Id_j (where Id_i (resp. Id_j) is the identifier of CE_i (resp. CE_j)).

However, as a meta-expression is an anti-symmetric binary relation between two expressions, two types of conflict can be encountered. They are processed automatically (by deleting all meta-expressions in conflict except one). The process uses a default solution that can be changed by the author.

- Conflict 1: The generation of the same relation between CE_i and CE_j and between CE_j and CE_i (e.g. $CE_1 \prec CE_2$ and $CE_2 \prec CE_1$). We propose to order sets of adaptations obtained after the first step according to (1) sets based on the navigational path of the graph, (2) sets exploiting the type of the resources, (3) sets exploiting the characteristics of the resources.
- Conflict 2: The generation of two meta-expressions between two identical expressions (e.g. $CE_1 \prec CE_2$ and $CE_1 \uplus CE_2$). We give a different priority to meta-expressions according to the defined relation: (1) Priority, (2) Recommendation, (3) Alternate.

We have implemented the following deduction process of CME_c. The *p* sets of identifiers coming from the first step are first ordered according to the proposed order in the resolution of conflict 1. In a second time, each meta-expression defined using these identifiers allows us to deduce multiple meta-expressions of CME_c. Each time a meta-expression is deduced, we check if it does not generate a conflict with the already generated meta-expressions. If a conflict of the first type is generated, the current meta-expression is not considered and the deduction process will continue. If a conflict of the second type is generated, we retain only one meta-expression according to the order defined in the solution of the second conflict.

6.3 Jane's adaptation strategy

We apply here our framework in order to define Jane's adaptation strategy $S1$. We consider that the elementary adaptation *S1-1*, *S1-2* and *S1-3* (cf. Figure 12) have been defined. Then, Jane has established correspondences between each elementary adaptation and a user characteristic *S1-1* with *in-depth* learning mode, *S1-2* with *inductive* reasoning mode, *S1-3* with *audio* presentation form. We focus now on the way *S1-1*, *S1-2* and *S1-3* are combined in order to produce $S1$.

	Expressions	Meta-expressions
S1-1	E_{1-1} = linked-transitive(r, goal, prerequisite) $\wedge$ linked(rCurrent, r, prerequisite) E_{1-2} = linked-transitive(concept, goal, pre-requisite) $\wedge$ linked(r, concept, *abstraction*)	$E_{1-1} \prec E_{1-2}$
S1-2	E_{2-1} = characteristicOf(r, format, =, audio) E_{2-2} = characteristicOf(r, format, =, text)	$E_{2-1} \mid E_{2-2}$
S1-3	E_{3-1} = instanceOf(r, Example) E_{3-2} = instanceOf(r, Definition)	$E_{3-1} \prec E_{3-2}$

Fig. 12. Description of *S1-1*, *S1-2*, *S1-3*

Next, Jane associates *S1-1* with *in-depth* learning mode, *S1-2* with *audio* display mode, *S1-3* with *inductive* reasoning mode.

S1-1, *S1-2* and *S1-3* are combined automatically to define $S1$. The combination process of their characteristic *Solution* is performed as follows. It has as input 3 elementary adaptations expressed on 3 different elements of the domain model. After the step 1 of the combination process, 3 sets are built, one adaptation per set. After the step 2, one combined adaptation is built, which is composed of 8 expressions and 44 meta-expressions. Among the deduced expressions, we have:

- $E_{c,1} = E_{1-1} \wedge E_{2-1} \wedge E_{3-1}$ = linked-transitive(r, goal, prerequisite) $\wedge$ linked(rCurrent, r, prerequisite) $\wedge$ characteristicOf(r, format, =, audio) $\wedge$ instanceOf(r, Example)
- $E_{c,2} = E_{1-1} \wedge E_{2-1} \wedge E_{3-2}$ = linked-transitive(r, goal, prerequisite) $\wedge$ linked(rCurrent, r, prerequisite) $\wedge$ characteristicOf(r, format, =, audio) $\wedge$ instanceOf(r, Definition)
- $E_{c,3} = E_{1-2} \wedge E_{2-1} \wedge E_{3-1}$ = linked-transitive(r, goal, prerequisite) $\wedge$ characteristicOf(r, format, =, audio) $\wedge$ instanceOf(r, Example)
- $E_{c,4} = E_{1-2} \wedge E_{2-2} \wedge E_{3-1}$ = linked-transitive(r, goal, prerequisite) $\wedge$ characteristicOf(r, format, =, text) $\wedge$ instanceOf(r, Example)
- $E_{c,5} = E_{1-2} \wedge E_{2-2} \wedge E_{3-2}$ = linked-transitive(r, goal, prerequisite) $\wedge$ characteristicOf(r, format, =, text) $\wedge$ instanceOf(r, Definition)

Among the deduced meta-expressions retained, we have: $E_{c,1} \prec E_{c,2}$, $E_{c,2} \prec E_{c,3}$, $E_{c,2} \prec E_{c,4}$, $E_{c,2} \prec E_{c,5}$ and $E_{c,3} \mid E_{c,4}$.

Among the deduced meta-expressions not retained, we have: $E_{c,3} \prec E_{c,2}$, $E_{c,4} \prec E_{c,2}$, $E_{c,1} \mid E_{c,4}$, $E_{c,2} \mid E_{c,5}$ and $E_{c,2} \mid E_{c,4}$.

7 Validation

In this paper, we promote two main ideas behind the EAP framework: *enabling authors to specify their adaptation strategies at a high level*, and *easiness of defining authors' adaptation strategies*. Here, we prove these ideas by presenting the implementation of our framework (cf.

Section 7.1), by discussing the execution of generated adaptation strategies using an existing adaptation engine (cf. Section 7.2) and by evaluating the expression of adaptation using the EAP framework versus a rule-based language (cf. Section 7.3).

7.1 Implementation of the EAP framework

The framework has been implemented as a plug-in of the Protégé tool[4], called EAP. Currently, EAP plug-in is under test. Its architecture is presented in section 7.1.1 and its main functionalities are described in section 7.1.2.

7.1.1 Architecture of the EAP plug-in

As described in Figure 13, the plug-in includes two parts.

First, a knowledge part gathers the library of elementary adaptation patterns and combination rules. The library is modelled in OWL[5], where each elementary adaptation pattern is an OWL class and is defined as a specialization of a class called *ElementaryAdaptationPattern*. On the other hand, the combination rules implement the combination process (cf. Section 6.2) in a declarative way using SWRL rules[6] and the swrlx built-ins[7].

Second, the process part is made of components performing interaction with an inference engine (in our case Jess) and the OWL Protégé editor. We have used the OWL Protégé API to manipulate the creator's domain and user models, the library of elementary adaptation patterns and their instantiations. We have also used the SWRL Jess Bridge to execute SWRL rules using Jess.

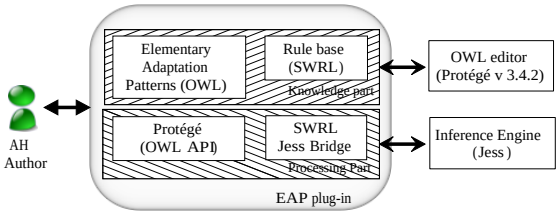

Fig. 13. Architecture of the EAP plug-in

7.1.2 Interaction with the EAP plug-in

The plug-in proposes multiple facilities. The author starts by loading his user and domain models. He can then define elementary adaptations by selecting an elementary adaptation pattern, and the constituents of the elementary adaptation. The solution part will then be generated automatically. The author can later define associations between an elementary adaptation and a user characteristic, while the combination process of multiple elementary adaptations is done automatically. Finally, EAP helps authors to export their adaptation strategies automatically in the GLAM format. This conversion is done automatically and is described in the following section. We plan to implement additional extensions, for example, to be able to generate LAG adaptation.

7.2 Execution of generated adaptation strategies

Adaptation strategies generated using the EAP framework are expressed at a high level, and are independent of any adaptation engine. Therefore, translators to existing adaptation engines are needed to execute these adaptation strategies. In this paper, we present our work to plug our framework on the GLAM platform, in order to be able to execute generated adaptation strategies by the GLAM adaptation engine. Before presenting this process, we first describe the GLAM platform.

7.2.1 GLAM platform

GLAM (Generic Layered Adaptation Model) is a platform 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. An adaptation strategy in GLAM is described in 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. Rules are expressed using a condition-conclusion format as:

$$\text{predicate}_1 \wedge ... \wedge \text{predicate}_n \rightarrow \text{Action (resource}_i\text{, degree)}$$

The condition part describes the conditions having to be satisfied by resources proposed to users. Usually, this part is related to the existence of a relation defining a particular navigational path in the domain model, eventually to a type of resources or to restrictions concerning the resource format expressed using attributes of the *Concept* or *Resource* classes.

The conclusion part describes the activity proposed to users for proposed resources. It includes two elements:

- *Action*: describes the proposed activity for the proposed resource (resource$_i$ in the rule above).
- *Degree*: can be used in different treatments. In GLAM, it is used to describe the relevance of a resource against the others. It allows several resources to be proposed to the user, the degree of relevance being represented with a code (color for example). The degree of relevance has five values (very high, high, medium, low, and very low), each value is associated to a particular color.

A level based on user-related knowledge and user characteristics. It is exploited using meta-rules. Meta-rules describe mechanisms that govern selection, scheduling, and excluding rules for a given user according to his profile. Let R_1, R_2 be two sets of rules, where

4. http://protege.stanford.edu/
5. www.w3.org/TR/owl-guide/
6. www.w3.org/Submission/SWRL/
7. The swrlx built-ins augment swrl rules with additional functionalities, e.g, creating new instances

four types of meta-rules are proposed. Each meta-rule is a binary relationship between rules:

- A preference meta-rule between R_1 and R_2 means that we prefer to execute R_1 rather than R_2, noted $R_1 > R_2$.
- A requirement meta-rule between R_1 and R_2 means that the execution of R_1 requires the execution of R_2, noted $R_1 \supset R_2$.
- An exclusion meta-rule between R_1 and R_2 means that either R_1 or R_2 is executed, noted $\overline{R_1 R_2}$.
- An order meta-rule between R_1 and R_2 means that R_1 is executed before R_2. The order meta-rules define a strict order between the elements on which they are expressed, noted $R_1 \prec R_2$.

7.2.2 Plugging EAP framework on GLAM platform

In order to be able to plug the EAP framework on the GLAM platform, we have first considered domain and user models used by the EAP framework in GLAM, then have translated adaptation strategies expressed using the EAP framework to adaptation in the GLAM format.

Concerning domain and user models. The essential elements of domain model used by our framework are found in GLAM: the modelling of resources, concepts, and properties and the relations between concepts and resources. Furthermore, GLAM adaptation engine execute resources in RDF [8]format, i.e., all OWL resources can be exported in RDF.

Concerning adaptation strategies. Adaptation strategies expressed using the EAP framework are composed of the characteristics *Name*, *Intent*, *Solution* and *Constituents*. However, only the characteristic *Solution* has to be translated to GLAM, this later includes a set of expressions and of meta-expressions. We have hence proposed two translations.

Translation of expressions to the GLAM format. 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 based on E_i.
- The conclusion part of R_i is generated with a desirability degree set to "medium".

Translation of meta-expressions to the GLAM format. For each meta-expression M_k belonging to the set of all the meta-expressions which compose an adaptation strategy, we perform on the following steps:

- If the kind of M_k is "$E_i \prec E_j$", and if R_i (resp. R_j) is the rule obtained from E_i (resp. E_j), we generate the meta-rule $R_i \prec R_j$.
- If the kind of M_k is "$E_i \mid E_j$", and if R_i (resp. R_j) is the rule obtained from E_i (resp. E_j), we generate the meta-rules $R_i \prec R_j$ and $\overline{R_i R_j}$.
- If the kind of M_k is "$E_i \uplus E_j$", 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 fixed. When two rules have a very high desirability degree, they can't have a higher degree than the existing one. The same principle is applied to a very bad desirability degree.

8. www.w3.org/RDF/

7.3 Evaluation of the EAP framework versus rule-based system

We have carried out an experiment with real users to see if it is easier and faster to express adaptation strategies using elementary adaptation patterns rather than using GLAM. For that, we have asked assistant professors from Supelec[9] and INRIA[10] to define adaptation strategies according to Jane use case (cf. Section 3).

The seven volunteers were asked to fill a questionnaire before starting the evaluation. The questionnaire was about their skills on AHS, on personalization, on rule-based languages and teaching experience. Most of volunteers have no skills on AHS, intermediate skills on personalization, only few of them have an intermediate background in rule-based languages. They have also between one to seven years experience in higher education (cf. Figure 14).

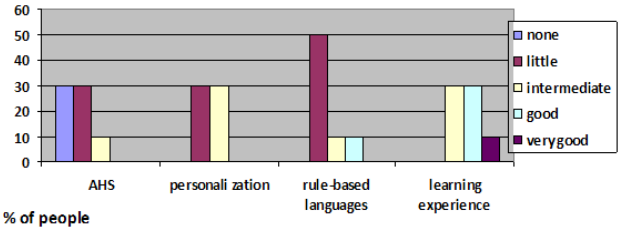

Fig. 14. Skills of our volunteers

Consequently, we start first by explaining them general knowledge on AHS. We let then volunteers making examples to get friendly with both solutions. Finally, they had to define at least $S1$ (cf. Section 3), first according to the EAP framework using EapTab, secondly according to GLAM rules and meta-rules using bloc notes. The evaluation was based on two criteria:

1) the difficulty of expressing adaptation strategies using each solution.
2) the time spent to express adaptation strategies using each approach.

In terms of difficulty (cf. Figure 15), most of the volunteers found GLAM very complex to use. It seemed difficult to express navigational paths and to define the correct meta-rules. On the other hand, they found the use of eapTab easier. The approach of breaking down

9. www.supelec.fr/
10. www.inria.fr/saclay/recherche/

an adaptation into multiple elementary ones seemed to be pleasant and intuitive for them. Only ergonomic requirements have been given in order to improve the usability of eapTab.

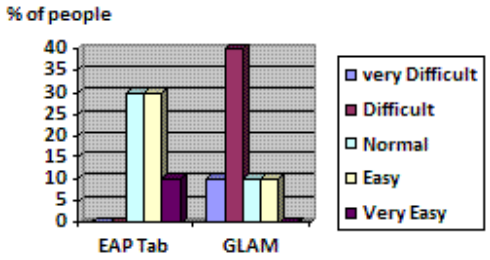

Fig. 15. Estimation of difficulty to express $S1$

Concerning the time estimation (cf. Figure 16), most volunteers were able to create similar adaptation strategies in eapTab within approximately half the time that they spent using GLAM.

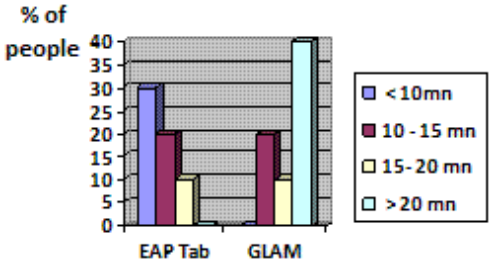

Fig. 16. Estimation of time spend to express $S1$

We have noticed that volunteers defined only $S1$ when they used GLAM, while they did not hesitate to define different strategies using eapTab. We explain these results by the fact that when using GLAM format, authors have to manually find all the conditions that must be satisfied by the proposed resources and manually compose the different conditions. The conditions must be written as rules in GLAM. Then, they have to define which rules is be applied to which user by writing meta-rules.

The adaptation strategy $S1$ is defined using eight rules and six meta-rules in GLAM (cf. Figure 2), where GLAM rules includes repetitive parts, e.g, the selection of definitions is present in four rules. While using EAP framework, it requires only three elementary adaptations (cf. Section 6.3).

8 Conclusion and Future Work

This paper proposes the EAP framework, in which adaptation strategies are described at a finer granularity than what is proposed by existing languages. This is obtained by the definition of 22 elementary adaptation patterns to express the adaptive navigation, and which are organized in a typology. The elementary adaptation patterns are based, on one hand on a criterion to select resources, on the other hand on a criterion to organize the selected resources. Thereby, we have defined a set of criteria to select resources and to organize selected resources.

Furthermore, the elementary adaptation patterns are independent from the domain model. They can be instantiated on a specific application domain in order to define elementary adaptations. The defined elementary adaptations are associated to user characteristics and combined to make whole adaptation strategies. One of the major benefits of the framework that we propose is the automation of a large and complex part of the process of generating complex adaptation strategies.

The generated adaptation strategies are expressed at a high level and independent of any adaptation engine. In order to be executed, we have described a first possible way using the GLAM adaptation engine. Furthermore, we have conducted experiments that showed the simplicity of our EAP framework rather than a rule-based language (in our case GLAM).

Our future work will be devoted to integrating our framework on other AHSs. We will study how to automatically translate generated adaptation strategies (using the EAP framework) to other generic adaptation languages like LAG, and GAL. It would also be interesting to do a scale-up test in order to detect the combinations most frequently used by creators. These adaptations could be directly recommended to them. Furthermore, it would be useful to extend our elementary adaptation patterns in order to be able to express adaptive content and adaptive presentation. Finally, our solution exploits user and domain models, which may evolve over time. Consequently, as a future work, we plan to consider the evolution of models, including how to evolve adaptation strategies defined using elementary adaptation patterns when domain or user models change.

References

[1] "http://www.blackboard.org/," March 2005, blackboard Inc, Blackboard.

[2] P. Brusilovsky and N. Henze, "Open corpus adaptive educational hypermedia," in *The Adaptive Web: Methods and Strategies of Web Personalization*, ser. LNCS, P. Brusilovsky, A. Kobsa, and W. Neidl, Eds. Springer, 2007, vol. 4321, pp. 672–696.

[3] P. Brusilovsky, "Adaptive navigation support," in *The Adaptive Web: Methods and Strategies of Web Personalization*, ser. LNCS, P. Brusilovsky, A. Kobsa, and W. Neidl, Eds. Springer, Berlin Heidelberg, USA, New York, 2007, vol. 4321, pp. 263–290.

[4] ——, "Adaptive hypermedia," *User Modeling and User-Adapted Interaction*, vol. 11, pp. 87–110, March 2001.

[5] ——, "Adaptive hypermedia," *User Modeling and User-Adapted Interaction*, vol. 11, pp. 87–110, March 2001.

[6] P. Brusilovsky, J. Eklund, and E. Schwarz, "Web-based education for all: a tool for development adaptive courseware," *Comput. Netw. ISDN Syst.*, vol. 30, pp. 291–300, April 1998.

[7] L. Pesin and P. Brusilovsky, "Isis-tutor: An adaptive hypertext learning environment," pp. 83–87, 1994.

[8] O. Conlan and V. Wade, "Evaluation of apels (2004). 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*, ser. LNCS, P. D. Bra and W. Nejdl, Eds., no. 3137. Springer, 2004, pp. 291–295.

[9] A. Cristea and L. Calvi, "The three layers of adaptation granularity," in *Proceedings of the 9th international conference on User modeling*, ser. UM'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 4–14.

[10] A. I. Cristea and A. de Mooij, "Laos: Layered www ahs authoring model with algebraic operators," in *WWW (Alternate Paper Tracks)*, 2003.

[11] ——, "Adaptive course authoring: My online teacher," in *International Conference on Telecommunications*, ser. 0-7803-7662-5. IEEE, 2003, pp. 1762–1769.

[12] E. J. Brown, A. I. Cristea, C. D. Stewart, and T. J. Brailsford, "Patterns in authoring of adaptive educational hypermedia: A taxonomy of learning styles," *Educational Technology & Society*, vol. 8, no. 3, pp. 77–90, 2005.

[13] D. Dagger, V. Wade, and O. Conlan, "Developing active learning experiences for adaptive personalised elearning," in *AH*, 2004, pp. 55–64.

[14] P. De Bra, G.-J. Houben, and H. Wu, "Aham: a dexter-based reference model for adaptive hypermedia," in *Proceedings of the tenth ACM Conference on Hypertext and hypermedia : returning to our diverse roots: returning to our diverse roots*, ser. HYPERTEXT '99. New York, NY, USA: ACM, 1999, pp. 147–156.

[15] P. De Bras, D. Smits, and N. Stash, "Creating and delivering adaptive courses with aha!" in *1st Eur. Conference on Technology Enhanced Learning*, ser. 0302-9743, LNCS, Ed. Springer, 2006, pp. 21–33.

[16] J. Danculovic, G. Rossi, D. Schwabe, and L. Miaton, "Patterns for personalized web applications," in *Eur. Conf. Pattern Languages of Program-EuroPLoP*, 2001, pp. 34–43.

[17] P. Dolog, M. Kravcik, A. I. Cristea, D. Burgos, P. D. Bra, S. Ceri, V. Devedzic, G.-J. Houben, P. Libbrecht, M. Matera, E. Melis, W. Nejdl, M. Specht, C. Stewart, D. Smits, N. Stash, and C. Tattersall, "Specification, authoring and prototyping of personalised workplace learning solutions," *IJLT*, vol. 3, no. 3, pp. 286–308, 2007.

[18] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[19] F. Garzotto, S. Retalis, A. Papasalouros, and K. Siassiakos, "Patterns for designing adaptive / adaptable educational hypermedia," *Advanced Technology for Learning*, vol. 1, no. 4, pp. 23–38, 2004.

[20] P. Brusilovsky and N. Henze, "Open corpus adaptive educational hypermedia," in *The Adaptive Web*, 2007, pp. 671–696.

[21] E. Knutov, P. De Bra, and M. Pechenizkiy, "Ah 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques," *New Rev. Hypermedia Multimedia*, vol. 15, pp. 5–38, April 2009.

[22] C. Jacquiot, Y. Bourda, F. Popineau, A. Delteil, and C. Reynaud, "Glam: A generic layered adaptation model for adaptive hypermedia systems," in *AH*, 2006, pp. 131–140.

[23] K. Levacher, E. Hynes, S. Lawless, A. O'Connor, and V. Wade, "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*, 2009, pp. 13–24.

[24] A. Moore, T. J. Brailsford, and C. D. Stewart, "Personally tailored teaching in whurle using conditional transclusion," in *Proceedings of the 12th ACM conference on Hypertext and Hypermedia*, ser. HYPERTEXT '01. New York, NY, USA: ACM, 2001, pp. 163–164.

[25] K. Sluijs, J. Hidders, E. Leonardi, and G. Houben, "Gal: A generic adaptation language for describing adaptive hypermedia," in *1st International Workshop on Dynamic and Adaptive Hypertext: Generic Frameworks, Approaches and Techniques*, 2009, pp. 13–24.

[26] N. Stash, A. Cristea, and P. De Bra, "Adaptation languages as vehicles of explicit intelligence in adaptive hypermedia," *Int. J. Cont. Engineering Education and Life-Long Learning*, vol. 17, no. 4/5, pp. 319–336, 2007.

[27] B. Steichen, S. Lawless, A. O'Connor, and V. Wade, "Dynamic hypertext generation for reusing open corpus content," in *HT '09: Proceedings of the 20th ACM conference on Hypertext and hypermedia*. New York, NY, USA: ACM, 2009, pp. 119–128.

[28] C. Stewart, A. Cristea, T. Brailsford, and H. Ashman, "Authoring once, delivering many: Creating reusable adaptive courseware," in *WBE 2005 Conference*, February 2005.

[29] I. O. for Standardization, "Information technology — syntactic metalanguage — extended bnf," ISO/IEC 14977, August 2001.