Assisting in the Reuse of Existing Materials to Build Adaptative Hypermedia

Nadjet Zemirline



ASSISTING IN THE REUSE OF EXISTING MATERIALS TO BUILD ADAPTIVE HYPERMEDIA

PH.D. IN COMPUTER SCIENCE

by

Nadjet ZEMIRLINE

Intended to be defended on July 12^{th}, 2011

Prof. Chantal REYNAUD	University of Paris-Sud-XI, INRIA-Saclay	Thesis director
Prof. Yolaine BOURDA	Engineering School Supelec	Co-advisor
Prof. Serge GARLATTI	Engineering School Telecom Bretagne	Rapporter
Prof. Philippe TRIGANO	University of Technology of Compiegne	Rapporter
Prof. Bruno DEFUDE	Engineering School Telecom SudParis	Examiner
Prof. Jean-Paul SANSONNET	Computer Sciences Laboratory for Mechanics & Engineering Sciences	Examiner

Contents

List of Tables

List of Figures

Part I

Introduction

CHAPTER 1

Introduction

1 Context and research questions

Since the explosion of Internet in 1990s, there has been a growing demand for personalization and the *one-size-fits-all* approach has been no longer applicable. One of the answers to these requirements has been Adaptive Hypermedia Systems (AHS), which adapt their behavior to the needs of individual users. We use the following definition of AHS, which is the more widely used:

> *By adaptive hypermedia systems we mean*
> *all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user* [6].

Thus, AHS are tools to access information based upon users' profiles represented in a *user model*. They also require a *domain model* to represent the application domain information. Adaptation mechanisms are defined in an *adaptation model* relying on these models and an *adaptation engine* is needed to execute the whole system. Systems authored and executed using these tools are usually referred by Adaptive Hypermedia (AH).

Adaptive Hypermedia Systems have amply proved their utilities, particularly in education, where they support authoring Adaptive Educational Hypermedia (AEH). In AEH[1], learners get access to personalized information according to their knowledge, preferences and goals. However, until today only few AEH have been developed, because of the difficulty of their authoring process.

Indeed, authors of AH, people who design the AH, have to:

1. define a *domain model*. This means that, they have to model the set of resources to be accessed by users, which are mainly a piece of media or a combination of pieces of media built dynamically (for example, a specific exercise). They can be related to each other by particular relations or characterized by attributes such as the format. Besides, concepts may be associated to these resources. For example, a document about DBMS explaining what are the *functional dependencies* is a resource related to the concept *functional dependency*. Similarly to resources, concepts may be related to each other by relations;

2. define a *user model*. This means that, they have to specify the users' characteristics that have to be considered by the AH. The characteristics may include personal information on users such as their first name, age, information on their preferences, information on their goals or even information on their historical use of the AH;

3. define an *adaptation model*. This means that, they have to define the appropriate adaptation mechanisms for each user. This may include several steps. For each step, authors specify, on the one hand, the most appropriate resources to propose to users and also the most appropriate way to present them, and on the other hand, the actions to be performed by users, those required by the AH in order to move to the next step.

Consequently, authors must know well the architecture of AHS in order to be able to create the different parts[2]. However, not all authors have modeling skills for expressing data models or have logic or programming skills for expressing adaptation.

Furthermore, authors have to answer to two questions:

How can they integrate their own data models into data models of an AHS? most often, authors come with their own data models and their instantiations and would like to reuse them.

[1] AEH are considered as a sub-part of AH

[2] Note that, defining a *domain model* or a *user model* is done in a similar manner. Both domain and user models structure a set of data. For this reason, in the following, we refer to both of them by data models.

1. The first challenge concerns reusing of authors' data models, their instantiations and their associated meta-data in the format supported by the used AHS.

   A direct consequence of this first challenge is the possibilities of loosing data. It may happen that some data cannot be translated and thereby will not be considered in the AH. This issue must not be underestimated.

   Another direct consequence concerns update mechanisms that are associated to the authors' models. The only way to keep these mechanisms is to rewrite them.

2. The second challenge concerns the lack of standard of the data models in the used AHS. This is an issue if the used system is no longer maintained or if authors want to change. Do authors need to re-create all their resources? To avoid this, it is desirable to move away from a "*one-to-one*" adaptive hypermedia authoring paradigm to a general "*write once, use many*" [70].

Most of developed solutions [23, 48, 30, 26] propose a translation process of authors' models. This means that, they require that authors translate all their data and meta-data. This translation process may come with a change of format or vocabulary. Therefore, such solutions fail solving the first challenge, but they try to override the second one thanks to several export possibilities.

How can they express adaptation? for expressing adaptation, authors have to describe multiple adaptation strategies. An adaptation strategy, as specified in [69], *defines how the adaptation is performed. Namely adaptation rules specified in the strategy are used to adjust the presentation to the learner with a particular learning preference [2], style [68] or need [6].* Thereby, authors face numerous challenges when defining their adaptation strategies.

1. The first challenge concerns the expression of adaptation strategies. It is often done using condition-action rules, event-condition-action rules or programming languages, which is a complex and time-consuming task. Recent solutions propose graphical tools or languages using constructors to support expression adaptation.
2. The second challenge concerns the reuse of adaptation strategies from one system to another one, and the expression of adaptation strategies independently of any AHS. This reflects the second challenge raised by the first question asked by authors: "*write once, use many*" [70]. The proposed paradigm endorses expressing adaptation at a high level, independently of all AHS and then translating this adaptation into a particular AHS.
3. The third challenge concerns the granularity in writing adaptation strategies. An adaptation strategy is composed of independent parts (for example: proposing definitions before exercises, proposing only textual textual resources). Its aim is to avoid writing the common parts of adaptation strategies several times.

Multiple solutions have been proposed [49, 23, 17, 69, 65] in order to meet these challenges. In [49, 23], they propose to make the expression of adaptation easier, but they were related to a particular AHS and failed to answer the second and the third challenge. [17, 69, 65] are concerned with the expression of adaptation using constructors and generic adaptation language, independent of any adaptation engine. These works failed to answer the third challenge, because till today, an adaptation strategy has been considered as a whole block and can not be easily reused.

Up to here, we have presented two groups of challenges. In this thesis, we take up each one as it is described below.

2 Contributions

Before going further, we would like to note that in this thesis: (1) we adopt authors' point of view, and (2) the aim of our contributions is to assist authors in the design of their AH using existing AHS, but in any way to build a new AHS.

In the following, we present our contributions taking up each group of challenges.

Integrating authors' data models into existing AHS : we propose a framework addressing the two challenges at the same time. Our objectives is to enable authors to reuse existing adaptation expressed using a particular system. Therefore, we propose an integration process of authors' models and data into existing systems without any translation or loss of information, and we rely on the use of OWL[3], a W3C standard[4].

For these purposes, we propose, on the one hand, to create a support for defining mappings between elements of the model included in an existing system and elements of the author's models, and on the other hand, to help creating consistent and relevant models integrating (1) the model included in an existing system and (2) the author's models and (3) taking into account the mappings between them. Therefore,

- we assist in the specialization and merging of two models using a semi-automatic process. This process delivers mappings between elements of both models with 100% precision;
- we propose a declarative reasoning module based on a modified version of the OWL meta-model;
- we assist in the detection of inconsistency mappings between elements (classes, properties or relations) of the two merged models.

Expressing adaptation : we propose a framework addressing the three challenges at the same time. It concentrates on the ease of defining adaptation strategies at a high level, at a fine granularity, and on the facility of reusing existing adaptation strategies.

We focus here on the expression of adaptation strategies for adaptive navigation, where users are forced to navigate among proposed navigation paths. This can be done by selecting, imposing a particular order or by recommending resources [41].

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. Therefore,

- we propose the EAP framework supporting the definition of adaptation strategies, through the use and the combination of elementary adaptation patterns. The most difficult part of the combination process is done automatically;
- we define a typology of 22 elementary adaptation patterns based on exhaustive criteria for selecting resources and specifying how to propose them;
- we plug the EAP framework on several existing solutions. For this purpose, we have studied deeperly some existing solutions: the GLAM platform, a rule-based system and the LAG language, a procedural language, which is translated to several existing systems, such as ADE [61] or AHA! [14]. From that, we have proposed a unifying

[3] www.w3.org/TR/owl-features/
[4] www.w3.org/standards/

vision of the domain model and an integrated vision of basic actions. This has enable us to define conversion rules from the EAP framework to, firstly, the GLAM platform, and secondly, LAG language.

These theoretical qualitative studies have been complemented by the development of plug-ins for the Protégé tool[5]. We propose

- A MESAM plug-in to merge two OWL models using a specialization process.
- An eapTab plug-in to support the functionalities of the EAP framework. These functionalities include the instantiation of elementary adaptation patterns, the association of instantiations with user characteristics and the combination of instantiations of elementary adaptation patterns. The eapTab also includes a module able to translate adaptation strategies written by our framework to the LAG format.

Furthermore, we have conducted experimentations and evaluations in the e-learning application domain, which allowed us to validate our approaches. We have made:

- experimentations of MESAM tab on a use case;
- evaluation with volunteers of the ease of use of the EAP framework versus a rule-based language (the GLAM platform) and versus a generic adaptation language (LAG).

All these contributions are based on the use of OWL[6]), and on declarative reasoning [33].

3 Outline of the dissertation

After having introduced in this first part, the context, the research questions and having listed our contributions, we present now the plan of the thesis.

In part 2, we provide background information on Adaptive Hypermedia (AH), on which we rely on to position and understand our contributions.

In part 3, we present how authors would be assisted in integrating their data models into existing systems. This contribution is generic and can be applied to integrate any other authors' models with any other systems, even for application domains different from the AH field.

In part 4, we detail how authors would express adaptive navigation in an easier manner, at a higher level and at a finer granularity than what exist in the AH field today. Furthermore, we study the expressivity of knowledge represented by our solution versus existing systems and present a unifying vision of expressing the domain model and also a unifying vision of the adaptation. This study about expressivity has served us to define translators from our approach to some existing systems.

In part 5, we describe the implementation of our different contributions, experimentations and evaluations in the e-learning domain application.

Additionally, part 3 and part 4 have been accompanied each by a use case on which we rely to explain our motivations and our contributions. Several appendices have also been written to provide more technical details. They haven't been put in the related contribution in order to facilitate reading this thesis, but they are clearly referenced.

[5]Protégé tool is available at *protege.stanford.edu/*
[6]http://www.w3.org/TR/owl-features/

Part II

Close work related to authoring Adaptive Hypermedia

CHAPTER 2

Positioning

This chapter provides background information on Adaptive Hypermedia (AH), on which we rely on to position and understand our contributions.

Section 1 introduces models used in AH. This includes the first reference model AHAM, and others models developed on the base of AHAM. We give a particular interest to the GLAM model, which has been extensively used in this thesis as it was previously developed in Supelec. Therefore, we have a complete access to all its components. Afterward, we describe systems implementing models describing AH, in order to support the authoring and the execution of AH.

From this point, we take authors' point of view for authoring AH. For this purpose, Section 2 describes solutions having been proposed to support the authoring process of AH. These solutions are organized in two categories: aids helping authors in modeling domain models, and aids helping authors in expressing adaptation. For each category, there are solutions dedicated to one AHS or compatible with several AHS. Consequently for each described solution, we specify whether it is dedicated to one AHS or is compatible with several AHS.

Section 3 points to the actual difficulties when authoring AH and introduce our contributions.

Before going further, two elements, concept and resource, have to be defined as they are defined and used differently by works presented in this chapter.

- A concept refers in the AH field to a notion.
- A resource refers in the AH field to a piece of media or a combination of pieces of media built dynamically. This media may be a document, a video etc.

Most often, a concept is related to one or more resources. Conversely, a resource is related to one or more concepts. Each time we present a work, we specify explicitly how these two elements are used.

1 Adaptive Hypermedia

In the following, we start by presenting some models used to describe AH (cf. Section 1.1). Afterward, we describe systems implementing these models in order to support the authoring and the execution of AH (cf. Section 1.2).

1.1 Models of Adaptive Hypermedia

Since the 1990s, several AH have been developed. However, during the first decade, they had been directly developed for specific purposes and in an ad-hoc manner. It was very difficult to maintain or reuse them. Furthermore, there was no reference model defining a general structure of the developed AH until 1999 when De Bra has proposed the first model for AH, called AHAM, which is mainly inspired by the *Dexter hypertext Reference model*[1] [35].

The AHAM model defines an AH as a 4-tuple [74]:

$$< \textit{domain model}, \textit{user model}, \textit{adaptation model}, \textit{adaptation engine} >$$

The domain model. It describes the structure of the available information of a particular application domain. This structure is modeled using concepts and relationships between concepts. A concept is an abstract representation of a piece of information of the application domain. It can be either atomic or composed of multiple concepts. Each concept has an identifier and may have an arbitrary number of attributes, where each attribute is defined by a name and a value. It can be linked to other concepts using relationships. A relationship has a name and is characterized by a weight. The most used types of relationships between concepts are binary relationships but other types of relations are allowed.

The user model. It consists of a set of properties describing users' knowledge of the application domain and personal information. These properties are exploited by the adaptation process. The user model is more precisely an overlay of the domain model. This means that, at least a user attribute is defined per concept. At the first execution of the AH, user properties may be initialized using default values or values collected using a questionnaire.

The adaptation model. It describes, on the one hand, the adaptation that would be proposed to users according to their defined properties, and on the other hand, how the user properties must be updated. These functionalities are expressed using event-condition-action rules by exploiting the structure and the content of the domain and user models. Two types of rules may be defined, generic or specific rules. Generic rules are expressed on domain and user models, whilst specific rules are expressed on instances of domain and user models.

The adaptation engine. It is responsible for computing the adaptation. It takes as input the three previous models. Afterward, it builds the pages that would be proposed to users and it updates the user properties.

As described above, the specification of the adaptation model in AHAM is very general. It does not specify explicitly what types of adaptation will be proposed. In fact, in the meantime when the AHAM model had been defined, another work done by Brusilovsky, had proposed a census of adaptations [6] in the AH field. This work is an enrichment of the AHAM model for new models.

Brusilovsky had made a census of two types of adaptations: adaptive presentation and adaptive navigation. For each type of adaptation, he has defined several non disjoint methods, that may be implemented differently. We present in Figure 2.1 the Brusilovsky typology

[1] The Dexter Model had been proposed in the early years of 1990 for hypertext applications.

of adaptation defined in 2001. From the left to the right, Figure 2.1 describes the two types of adaptations, for each type its possible methods and for each method its different implementations. For example, in the adaptive navigation, the link hiding method may be implemented either by hiding, disabling or removing the links on which the adaptation is acting.

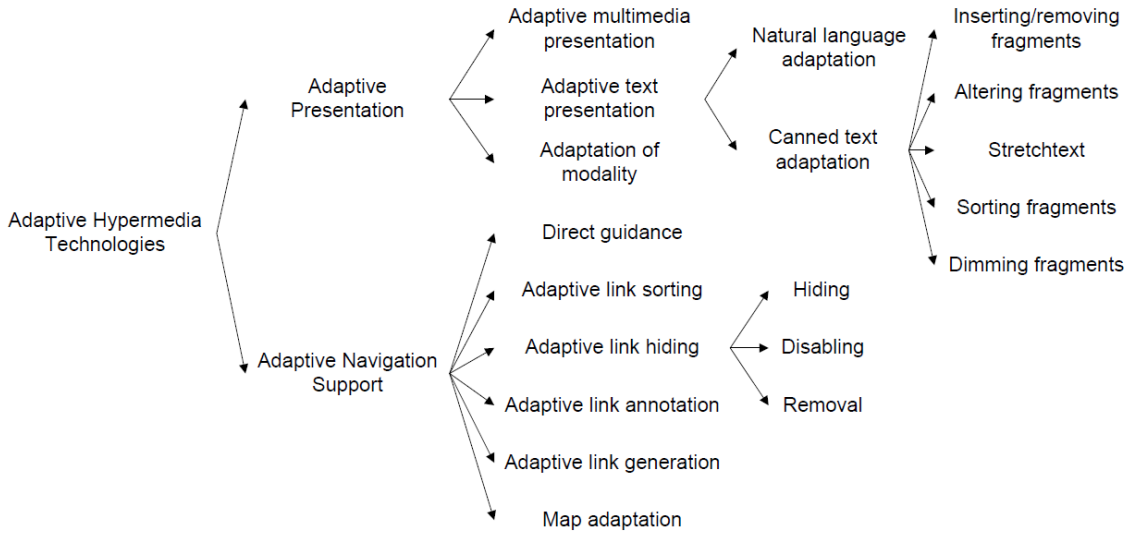

Figure 2.1 – The Brusilovsky typology [6]

Adaptive presentation: it concerns presentation aspects of pages proposed to users. For a particular user, this method specifies whether fragments has to be enlarged or not.

Adaptive navigation: we give further details on methods of this particular type of adaptation in Chapter 4, Section 1.

Note that, methods and their implementations can be applied to content, presentation and navigation adaptation. Recently, Knutov et al. [41] have decided to separate between the three types of adaptation. Therefore, they propose non-disjoint methods for content adaptation, adaptive presentation and adaptive navigation.

Often in order to express adaptive presentation and content adaptation, additional information is needed in the domain model. This is possible only when the structure of the domain model is not going to evolve, thus they are easily applied on close corpus AH[2]. Besides, adaptive navigation consists only of proposing navigational paths to users among a set of concepts and resources [7], for example: ordering a set of resources. This does not need any additional information on concepts and resources. Therefore, adaptive navigation can be applied on both close and open corpus.

The AHAM model is considered as the reference model in the AH field. Consequently, based on AHAM, several models have been proposed [62, 39, 13] taking its advantages and trying to propose extensions, particularly for the e-learning domain. They have also exploited the work done by Brusilovsky in defining adaptation. In [41], a presentation and a deep analysis of the most known and used models in the AH field has been done.

[2]Note that, two types of AH have been defined according to whether they act on a close or an open corpus of documents [9].

In the following, we present a description of a few models compared to the state of the art described in [41], and also some other ones not cited in [41]. As we will describe below, we have chosen these particular models for their specificities. In Section 1.1.1, we present models allowing to define AH whatever the application domain is (e-learning, on-line help, ...). In Section 1.1.2, we describe models defined specifically for AEH and we conclude in Section 1.1.3.

1.1.1 Models for Adaptive Hypermedia whatever the application domain

Several models supporting the authoring of AH for any application domain have been proposed. We present in this section two of them: the SEM-HP model which is based on semantic web technologies, and the GLAM model which is more generic than existing models and has been extensively used in this thesis for illustration purposes.

SEM-HP

The *SEMantic, Systemic and Evolutionary Model for the development of the adaptive Hypermedia system* [48] is a semantic model for authoring and maintaining AH. It is based on the AHAM model. Therefore, it proposes the same separation of expressing domain, user and adaptation models. SEM-HP is different from AHAM in several points. One of them is the fact that SEM-HP includes two abstraction levels.

The first level is dedicated for authoring the AH through the composition of four subsystems, presented below.

1. *Conceptual subsystem*: this subsystem includes the definition of the domain model using a semantic net [63]. The semantic net can include two types of nodes: concepts or items.
   - A concept is similar to the notion of concept used in the AH field (cf. Section 2). It is defined by a name, an identifier and is possibly linked by semantic relations to other concepts.
   - An item is a piece of information that can be proposed to users. It corresponds to a resource in the AH field. An item is defined by a name, a content and is linked by an association to a concept. The association has a name, a domain, a range and a weight.
2. *Presentation subsystem*: it concerns the specification of the part of the semantic net which is going to be reused in the AH and adapted to users.
3. *Navigation subsystem*: in this subsystem, the navigation of the semantic net is described using rules based on temporal logic. The rules are deduced by exploiting the semantic relations defined between concepts. More details can be found in [47].
4. *Learning subsystem*: it concerns the definition of user characteristics, user updates and the adaptation.

   Note that here, the user goals are considered as user characteristics. The definition of user characteristics includes user knowledge, interest, preferences and goals. These characteristics are updated according to the interaction of the user with the AH. Therefore, the user updates are triggered after each user action.

   The adaptation concerns the definition of constraints needed to access items. It is specified using condition-action rules. Each rule is composed of constraints about specific concepts or specific user characteristics.

The second level is dedicated for maintaining the consistency of the AH after the execution of user actions. Each user action is evaluated before being executed. This includes verifications on the definition and the execution of the action itself, like: verification of each of its preconditions

and postconditions, simulation of the execution of an action etc. Other verifications have been added, like checking the consistency of the semantic net (ensuring that each semantic association is defined between two concepts, each functional association must have at least a name, etc.). These last verifications are checked at the first execution of the AH and checked each time the user action modifies the domain model. Once all verifications are satisfied, the action is executed and changes are propagated on the concerned AH.

Another difference between the AHAM and SEM-HP models is that, SEM-HP proposes to users to navigate inside a conceptual structure (a semantic net) where in the AHAM model users navigate through a set of hierarchical links (a sort of composition links).

A system, JSEM-HP, has been implemented based on the SEM-HP model (cf. Section 2).

GLAM

The Generic Layered Adaptation Model (GLAM) [38] is a platform allowing to define AH particular to any application domain. It has been developed previously in the department of computer science at Supelec in 2006 [39]. The platform is based on the AHAM model. It is made up of a generic adaptation model relying on generic user and domain models, and of an adaptation engine. Below, we detail further each of the models composing GLAM.

- *The GLAM domain model*: it proposes a clear separation between modeling concepts and resources. The GLAM domain model does not impose any particular constraints on modeling the available resources. The authors are free in their modeling whatever their vision is. Following this purpose, GLAM domain model aims to be a generic model, more generic than others. C. Jacquiot et al. in [38] have shown that AHAM or Munich domain models are specializations of GLAM domain models.

- *The GLAM user model*: similarly to the modeling of the GLAM domain model, the GLAM user model does not impose any particular constraints. A user may be described through its personal information, knowledge about a specific concept or a resource, and its goals. Also, the GLAM user model aims to be more generic than other existing user models. C. Jacquiot et al. in [38] have shown that AHAM or Munich user models are specializations of GLAM user models.

- *The GLAM adaptation model*: on the contrary to the existing models which propose to consider at the same time the user and domain models for expressing adaptation, GLAM brings a new vision for expressing adaptation (for an example of GLAM adaptation cf. Chapter 4, Section 2). It proposes to express adaptation at two distinct 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 on proposed resources. It includes two elements:

  – *Action*: an action describes the proposed activity for the proposed resource (resource_i in the rule above).

- *Degree*: a 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. 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. Therefore, an association between each user characteristic and the set of rules applicable for this user characteristic is defined. As a user has multiple characteristics, meta-rules have been proposed to select the appropriate rules to be executed.

Let R_1, R_2 be two sets of rules, 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 R_1 is preferred 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 must be executed before R_2. It defines a strict order between the elements on which they are expressed, noted $R_1 \prec R_2$. Note that, in GLAM, two types of orders can be proposed. A partial order is expressed using the degrees of desirability and an imposed order is expressed using the order meta-rule.

A set of verifications have been integrated to the adaptation model, such as looking for dead rules that are no more useful, or checking if the adaptation is deterministic and is calculated in a polynomial time.

Note that, similarly to the SEM-HP model, the GLAM model proposes to users to navigate through a set of concepts, on the contrary of the AHAM model where users navigate through a set of links.

1.1.2 Models for Adaptive Educational Hypermedia

In this section, we present one of the latest models for authoring AEH, called CAM. It is based at the same time on the AHAM and LAOS models. For this purpose, we start by presenting the LAOS model. Afterward, we describe the CAM model.

LAOS

The *Layered AHS Authoring Model and Operators* is specific for defining adaptive hypermedia in educational field [13]. It is an extension of the AHAM model. In fact in addition to proposing a domain, a user and an adaptation model, the LAOS model proposes modeling of two other models: the goal and constraints model and the presentation model. We describe in the following each of the five models of the LAOS model[3].

- *The domain model*: it is defined through a set of concepts, where each concept has a name and a set of attributes. Each attribute has a name and a content. The attribute corresponds to what is referred in the AH field by a resource.

[3] A simplified modeling of the domain model and goal and constraints model using UML is described in Chapter 5, Section 1.1.

Concepts can be linked by two types of relations. A hierarchical relation (a sort of composition relation) with a multiplicity of 0..* and a relatedness relation with a multiplicity of 1. The domain model is very similar to the one of the AHAM model except that it does not allow modeling specific meta-data. The modeling of meta-data is done in the goal and constraints model.

- *The goal and constraints model*: it filters and restructures the domain model, with respect to pedagogical goals. This means that, it is based on some (or all) the concepts of the domain model. The concepts included in the goal and constraints model may only be
  - restructured. By restructuring, we mean that, authors may modify how the concepts are linked by the hierarchical relation. For example: assume in the domain model the author defines C1, C2, and C3 as concepts, such as C1 is the parent of C1 and C3. In the goal and constraints model, the author may restructure these concepts like: C2 is a parent of C3 and C1.
  - enriched with meta-data. Two types of meta-data are allowed for each concept map: a string named *label* and an integer named *weight*.

  Note that, the content of concept maps cannot be modified in the goal and constraints model.

- *The user model*: it defines two types of user knowledge: knowledge about the domain model (for example: the attribute *access* indicating whether a concept has been visited or not by a user) and personal information (for example: the attribute *login* used for identifying the user). It is very similar to the one of the AHAM model.

- *The presentation model*: it takes into consideration the physical properties (for example: the attribute *widget* indicating the type of device used by the user for accessing the AH) and the environment of the presentation (for example: the attribute *scale* indicating the width of texts used when building pages that would be proposed to users).

- *The adaptation model*: it is also called the LAG model. It is still under development in order to take into account all types of adaptations of the Brusilovsky typology. It is defined in three levels [12]. The low level concerns the adaptation event-condition-action rules. The medium level concerns the expression of adaptation using a set of constructors (a sort of a higher level language[4]), an automatic translator has been developed to generate adaptation at a low level. The high level concerns the description of adaptation strategies and their annotation in order to make their reuse easier. No translation to the other levels is made.

  Authors often express adaptation strategies at the medium level using constructors. In fact, constructors make it easier the expression of adaptation strategies (for examples of constructors see LAG cf. Section 2.2, and for an example of a LAG adaptation strategy see Chapter 4, Section 2). The constructors may be used at a generic level by exploiting the structure of the domain (and goal) model, or at a specific level by exploiting directly the content of the domain (and goal) model.

Several implementations enabling to define AH at a high level and compatible with several adaptation engines have been defined (cf. Section 2.1.2, Section 2.2.2).

CAM

The *Conceptual Adaptation Model* has been developed in the context of the GRAPPLE project. The *Generic Responsive Adaptive Personalized Learning Environment* project is a EU FP7 STREP Project[5]. It ran from February 2008 until February 2011.

[4]Several implementations have been done allowing to express adaptation at this level such as LAG or LAG-XLS (cf. Section 2.2).

[5]Information related to the GRAPPLE project including its deliverables are available at *http://www.grapple-project.org/*

It has been proposed by the same people than those who developed the AHAM model, and in collaboration with those who defined the LAOS model. The CAM model aims to be more general and more flexible than AHAM or LAOS models. In fact, the CAM model allows defining AH with an extensible number of layers, but with at least three layers. The three layers must include one layer about the domain model, another layer about the user model and at least one layer for adaptation aspects. Each layer may interact with all the other layers. We describe below each of these layers.

- *The domain model*: it defines the conceptual structure of the available resources of a learning application. For this purpose, it includes the description of concepts (as defined in the AH field), their associated properties, binary relations between them[6] and the resources (as defined in the AH field) describing these concepts.

  Note that, here, only the semantics of concepts, properties and relations are defined, no behavior is associated to them. The domain model is very similar to the one defined in the LAOS model. However, one major distinction with LAOS, is that CAM proposes a graph organization of resources, rather than a hierarchical tree of resources.

- *The user model*: it defines the user characteristics that will be considered in the AH. The user model is very similar to the one defined in the AHAM and LAOS models.

- *The adaptation model*: it defines the adaptive behaviors. One adaptive behavior is defined per layer. It is expressed using constraints on the elements (concepts, attributes or relations) of the domain model. For example, we can define *a goal layer*: similar to the goal and constraints model in LAOS. This layer identifies the sequence of concepts that should be learned in order to reach a particular goal.

All the layers are gathered together in order to compose an adaptive hypermedia.

1.1.3 Summary

As it has been presented in this section, several models based on the AHAM model, have been proposed. Each of them introduces new terms, new interfaces and new adaptation implementations. A recent work [41] has highlighted main similarities and differences between most known and used models in the AH field. This work is considered as an important step in order to define an unifying model that will group all the specificities of other models.

Based on existing models, Adaptive Hypermedia Systems (AHS) have been implemented. They propose to authors domain, user and adaptation models and most importantly an adaptation engine. In the following, we present some existing AHS.

1.2 Adaptive Hypermedia Systems

Similarly when describing models of AH, here, we distinguish between AHS allowing to author AH for different application domains (cf. Section 1.2.1), from AEHS allowing to author AEH (cf. Section 1.2.2).

[6] Only binary relations can be modeled. This means that, when authors want to express n-ary relations, they have to decompose them in several binary relations.

1.2.1 Adaptive Hypermedia whatever the application domain

AHA!

The Adaptive Hypermedia for All (AHA!) system[7] is an implementation of the AHAM model. It has been under development since 1996. It has been regularly extended with new features until 2006 where the last version AHA! 3.0 [22, 21] used in this thesis has been developed. It is an open source system, this means that, its sources are downloadable and modifiable if desired.

The AHA! system proposes an adaptation engine to execute adaptive hypermedias composed of the following models:

- *A user model*. It is considered as an overlay of the domain model and it implements two types of users:
  - users knowledge (relating to the position of users in the domain model). Each user knowledge is necessarily associated to at least one concept. It can be persistent or not, and its type is of three different kinds: integer, boolean or string;
  - personal data (such as the name of a user). As the user model is an overlay of the domain model, personal data cannot be modeled apart. Consequently, a particular concept is added that will include all personal data about users.

    Note that, AHA! does not consider the goal of users. If authors want to consider it, they have to explicitly add a particular attribute and write the corresponding adaptation.
- *A domain and an adaptation model*. Even if AHA! is based on the AHAM model, one of the major differences is that AHA! does not separate domain and adaptation models. Therefore, there is only one model describing available concepts with embedded adaptation.

  AHA! considers three types of concepts: abstract concepts, pages and fragments. Each of them is described further below:
  - abstract concepts are equivalent to concepts in the AH field;
  - pages and fragments are equivalent to resources in the AH field.

  Each concept is characterized by a name, a description, a type (abstract concepts, pages and fragments) and an unlimited number of attributes. Each attribute can be either persistent or not. It can be either a meta-data about a concept (for example: the attribute *format* indicating the type of media: text, video, or image) or a user's knowledge (for example: the attribute access indicating whether the concept has been visited by the user or not). Furthermore, concepts can be linked between them by relations, necessarily using a *hierarchical* relation (a sort of composition relation) and eventually using other types of relations. The AHA! includes verifications allowing to detect cycles when defining relations and thus minimizing the termination problem that may happen.

  The embedded adaptation includes at the same time adaptation rules and update rules. We detail each type of rules further below.
  - The adaptation rules must be expressed using event-condition-action rules. We are not going to present the structure of these rules, more details can be found in [21]. Note that, AHA! supports two types of adaptation in regard to the Brusilovsky typology: adaptive presentation and adaptive navigation.
    1. As for the adaptive presentation, AHA! supports almost all types of adaptive presentation of the Brusilovsky typology, except the sorting of fragments and natural language adaptation. However, AHA! allows to define adaptation in a

[7] http://aha.win.tue.nl/

static manner. That means that, when a type of adaptation is applied for a user, it is no more editable. Authors may specify how each page should look using the conditional inclusion of fragments.

2. As for the adaptive navigation, AHA! supports the hiding, disabling and removing links, and the adaptive link annotation. However, it does not support the adaptive link generation and map adaptation of the Brusilovsky typology.
   A page presented to users includes a set of links and a content. The AHA! adaptation engine calculates whether the embedded links are going to be proposed or not and if yes it calculates the appropriate color (blue for unvisited, purple for visited)[8].

– The update rules act on the user model (i.e., updates user knowledge) and are expressed using if-then-else rules. We are not going to describe these rules but more details can be found in [4].

The AHA! adaptation engine includes a component for building HTML pages that would be proposed to users[9]. The execution algorithm is iterative and is executed in a sequential way. It includes three steps.

1. First phase concerns the initialization of user knowledge. This includes the personal data and all attributes of concepts. After what, rules to be executed are selected.
2. Second phase concerns the updates of user characteristics before performing adaptation. After what, the new page is built and is presented to the user.
3. Third phase concerns the final updates of user characteristics.

Note that, the phases are executed in that order, i.e. rules belonging to the first phase have a higher priority than those belonging to the second phase.

1.2.2 Adaptive Educational Hypermedia Systems

APeLS

The Adaptive Personalized e-Service [10, 19] delivers personalized courses to learners, and these personalized courses are modeled in three distinct models.

- *The learner model*. It includes the description of learners that have to be considered to perform personalization, including their learning goals.

  This model can be seen as a sub-part of the user model of the AHAM model as the learner model is dedicated for e-learning domain. Two interesting features have been integrated to the user model in order to update the learner model: a Rebuild Scope questionnaire for learner characteristics and a Rebuilt Style for Learning styles. They are proposed as questionnaires, and learners have to answer them for each update.

- *The content model*. This model can be seen as the domain model of the AHAM model. It specifies the learning resources[10] that are going to compose the personalized courses. The learning resources are organized using the *Candidate Content Groups*[11] (CCG). Learning resources are grouped in the same CCG when they express the same goal but are implemented differently. Note that, the goal of each CCG is described in its meta-data.

[8] These colors are those defined by default in the AHA! system but they can be modified by the authors.

[9] For simplicity here, we say that the built page is presented to the user, but in reality the AHA! acts as a server and the user is a client. It follows more a client-server architecture.

[10] ltsc.ieee.org/wg12/20020612-Final-LOM-Draft.html

[11] We did not find an official description but we used in this thesis the one found at http://www.w3.org/2004/06/DI-MCA-WS/presentations/final/121450/text18.html and followed by APeLS

- *The narrative model*. This model can be seen as the adaptation model of the AHAM model. It supports two types of adaptation: the adaptive presentation and adaptive navigation. There is no clear description what is precisely supported and not supported among the two types of adaptation.

  - For the adaptive presentation, it is supported through the principle of candidacy (for more details see [19]).
  - For the adaptive navigation, authors may build narrative courses by defining an order between the CCG. Afterward, the adaptation engine will choose whenever the most appropriate learning resources to present from each CCG according to the learning model.

Therefore, the APeLS implements in its own way the three models of the AHAM model.

1.2.3 Summary

So far, we have introduced some existing AHS that support the authoring process and execution of AH. We have particularly focused on two types of AHS: those allowing to author AH for any application domain and those allowing to author AEH.

Now, let us see how authors can use these AHS to create and execute their own AH.

2 Authoring Adaptive Hypermedia

As described before, each AHS proposes a domain, a user and an adaptation model. It also includes an adaptation engine allowing to execute the adaptation model on domain and user models. When authors use one of existing systems, they encounter several problems.

The first problem is the difficulty of authoring AH using directly these systems. In fact, using a AHS requires that authors have multiple skills. When using the domain and user models of the used AHS, authors must have modeling skills, whilst when using the adaptation model, they must have logical and programming skills. However, most often authors do not have these required skills. For example, when a teacher want to propose a AEH, he hasn't necessarily the logic and programming skills to author the adaptation model. This is why, solutions have been proposed in order to help authors during the development process of their AH.

The second problem is the reuse degree of an AH. In fact, when an AH is developed using a particular AHS, it is dependent of this used system, which is an issue if the system is no later maintained, if authors want to change the current system or if authors want to express more than what is supported by the current system. For this reason, a new paradigm has been proposed '*write once, use many*' [70], where the domain model and the adaptation are expressed at a high level, and are independent from existing adaptation engines. Therefore, translations to existing adaptation engines have been encoded in order to execute them using existing adaptation engines.

Note that, most of the proposed solutions do not consider the user model separately. Some of them propose to consider the user model at the same time with the domain model like *Concept Editor* [23], or at the same time with the adaptation model like *MOT adapt* [12]. Therefore, we don't define a separate section for modeling the user model, but we describe the modeling of the user model according to specificities of each solution.

In the following, we present solutions to define domain models in Section 2.1, and to express adaptation in Section 2.2.

2.1 Specifying the domain model

Here, we distinguish between solutions proposed at the top of a particular AHS (cf. Section 2.1.1), from those allowing to express domain model at a high level and therefore compatible with numerous AHS (cf. Section 2.1.2).

2.1.1 Solutions dedicated to a particular Adaptive Hypermedia System

Concept editor

The Concept editor [23] is a Java applet integrated to AHA! for defining the domain model and the adaptation rules.

- *Definition of a domain model*. The Concept editor proposes a domain model, which has to be followed by authors when defining their own domain model. So, authors have only to define instances of the proposed domain model.

  Authors are allowed to define new instances of concept. For each new instance, they specify the following information: name, title, content, type, whether it is an element of a hierarchy (a sort of composition relation) and its parent, siblings and children. An instance may have several attributes, persistent or non persistent[12].

  The Concept editor includes also the definition of the user model, which is considered as an overlay of the domain model. Consequently, for each instance of concept, authors have to specify attributes on user characteristics to be considered in the adaptation rules later.

- *Definition of adaptation rules*. Similarly to the user model, the adaptation rules are also considered as an overlay of the domain model. We detail this process in Section 2.2.

Graph author tool

Similarly to the Concept editor, the Graph author tool [23] is a Java applet integrated to AHA!. However, this later is closer to the AHAM model in its implementation. That means that, there is a clearer separation between specifying available instances of concepts and adaptation rules than what is done in the Concept editor. More precisely, authors specify instances of concepts and according to that, the tool is able to generate automatically adaptation rules. These two distinct processes are described below.

- *Definition of a domain model*. Similarly to the Concept editor, the graph author tool proposes a domain model that authors have to use when defining their own domain model. That means that, they have only to define instances of classes in the domain model of the Graph author tool.

  Similarly to Concept editor, the user model is considered as an overlay of the domain model.

  An interesting distinction between both tools is that, using the Graph author tool authors can specify more relations between instances of concepts. A set of predefined relations are proposed, furthermore authors may add new ones. However, adding new relations is advised only for technical authors as it required to extend the generated adaptation manually (cf. Section 2.2).

- *Definition of adaptation rules*. Adaptation is generated automatically based on defined templates exploiting the domain model of the Graph author tool. We detail this process in Section 2.2.

[12] Non persistent means that, the attributes are saved only during the current session. They will be reinitialized in the next session.

JSEM-HP

Java-based SEM-HP [48] is a tool implementing the SEM-HP model (cf. Section 1.1.1). It proposes a set of graphical tools supporting the authoring of different subsystems of the SEM-HP. Here, we detail only the aids concerning the conceptual and presentation subsystems, whilst in the section 2.2 we present the implementation of the learning and navigation subsystem.

- A graphical tool has been associated to the conceptual subsystem. It enables authors to define easily their semantic nets in a consistent way. More precisely, the tool enables authors to define concepts, semantic relations between concepts, items[13] and an association between a concept and items. Each association must have a name and eventually a weight. The weight represents the importance of the item associated to a concept.

- Similarly, a graphical tool has been associated to the presentation system. It enables authors to reuse existing semantic nets, to restructure them and eventually extend them. The tool ensures that the final semantic net is consistent. For this purpose, it integrates automatic verifications described previously in the SEM-HP model, like the fact that there is no cycle.

2.1.2 Solutions compatible with numerous Adaptive Hypermedia Systems

MOT

My Online Teacher (*MOT*) [30] has been developed over the last 7 years and several versions of *MOT* have been proposed. We detail here the version 3.1. *MOT* is an implementation of the specification of the domain model and of the goal and constraints model in LAOS [13]. Therefore, MOT[14] is a tool specific to e-learning applications. It enables an author to express his domain and goal models at a high level independently of any adaptive hypermedia system. The author may export his models in CAF or RDF(S)[15].

MOT makes a clear separation between modeling the domain model and the goal and constraints model.

- The domain model includes domain maps (concepts and attributes of concepts[16]). The concepts are structured using a *hierarchical* structure (a sort of composition relation), using attributes to describe their content and possibly using relatedness relations computed automatically between concept maps (for more details see [30]). It can be built from scratch using the visual interface or by importing existing materials such as power point presentation[17], a wikipedia web page[18], IMS-QTI[19] or SCORM content[20], etc.

- The goal and constraints model is based on a part (or parts) of a particular domain model. Here, the content of the domain maps can't be modified, but it can be:

  - restructured by modifying the hierarchical structure or defining attributes of a concept as attributes of another concept;

  - enriched with meta-data (In the version 3, it supports only two meta-data: a *label* which is string and a *weight* which is an integer)[21].

[13] An item corresponds to a resource in the AH field.
[14] MOT is available on-line at *http://mot.dcs.warwick.ac.uk/*.
[15] *http://www.w3.org/TR/rdf-schema/*
[16] Note that, *attributes of concepts* are called *resources* in the AH field.
[17] *http://office.microsoft.com/en-us/powerpoint/*
[18] *http://www.wikipedia.org/*
[19] *http://www.imsglobal.org/*
[20] *http://scorm.com/*
[21] Note that, the meta-data used in MOT are specific to educational context.

Note that, the goal and constraints model is designed specifically for a particular lesson. For this reason, MOT people speak about lessons and sub-lessons. In this section, we have chosen to keep the terms of concept maps and attributes not to disturb the reader.

One of the main advantages of MOT is the separation between roles of authors. An author working with MOT is assumed to be a content author who has background on modeling resources and not necessarily on expressing adaptation. Thereby, several authors may be involved in the design of an AH, even for designing the domain model and the goal and constraints model.

Domain Model tool

The Domain Model tool [26], often abbreviated by DM tool, has been developed in the context of the GRAPPLE Project. More precisely, it is one of the tools integrated in the Graphical Authoring Tool (abbreviated in GAT). In order to explain the role of the DM tool, let us present first a global overview of GAT.

GAT aims at allowing to define an AH only using visual tools, at a high level and system-independently. It includes three components.

1. *Domain Model (DM) authoring tool*: it allows defining concepts, properties of concepts and relations between concepts. With this tool only the semantics of the relations are specified, not their behavior which is specified later using the CAM tool.

2. *Concept Relationship Type (CRT) authoring tool*: a CRT file describes the adaptive behavior of elements (concepts, resources, properties, or relations) of the domain model using constraints. It defines also the updates that have to be performed on user knowledge.

3. *Conceptual Adaptation Model (CAM) tool*: it proposes to instantiate the CRT on concrete elements (concepts, resources, properties or relations) of the domain model. Furthermore, the tool aims at gathering all the adaptive behaviors together. Therefore, it proposes a global vision of adaptation at a high level and using graphical aspects.

All the components are needed. In fact, a concept not defined using the DM tool can't be exploited in the CRT or the CAM tool. Also a concept defined using the DM tool will not be proposed to users if it is not selected using the CRT tool.

Note that, materials modeled using these components are expressed in XML[22] in order to facilitate interoperability between the three components. They are stored in a database using an internal format. For this purpose, translators as web services have been implemented in order to facilitate storage and access to these materials.

Similarly to the MOT tool, the DM tool describes the structure of the available concepts and resources. It enables authors to specify the following information.

- General information about the domain model, like, its name, its creation date, the name of its authors, etc.

- Available concepts, by defining for each concept: a name, an identifier, a description, a set of keywords, a set of resources and binary relations with other concepts.

  Each resource must be identified by a name, an usage type, a location and a set of meta-data can be specified using the LOM standard[23].

  Each binary relation is characterized by its name, its domain and range. A set of relations with a predefined semantic is proposed: is-composed-by, is-a, belong-to.

[22] www.w3schools.com/xml/default.asp
[23] ltsc.ieee.org/wg12/20020612-Final-LOM-Draft.html

2.1.3 Summary

So far, we have presented solutions enabling authors to define a domain model usable by existing AHS. But as it has been shown all these existing solutions propose a predefined domain model that has to be used by authors. That means that, authors must use these predefined models, which is an issue if they have their own models and instances. Consequently, they become obliged to translate their instances of their domain model and their associated meta-data in the format accepted by the used solutions.

2.2 Expressing adaptation

Here, we distinguish also between solutions related to a particular AHS (cf. Section 2.2.1), from those allowing to express adaptation at a high level and therefore compatible with numerous AHS (cf. Section 2.2.2).

2.2.1 Solutions dedicated to a particular Adaptive Hypermedia System

Concept editor

As described before in Section 2.1, the Concept editor [23] helps to instantiate the domain model included in the Concept editor, to specify user characteristics and also adaptation rules. Here, we focus only on the definition of the adaptation rules.

As the adaptation rules are considered as an overlay of the domain model, authors have to specify for each concept, what is going to happen when a specific attribute has a particular value using If-Then-Else rules. They have also to describe how user knowledge must be updated.

There is no support for expressing adaptation rules. Authors have to write the adaptation rules by themselves using the defined syntax. They have also to specify whether a rule triggers other rules or not. The AHA! adaptation engine includes automatic verifications in order to prevent termination and confluence problems.

Consequently, the Concept editor is considered as a low-level authoring tool because authors have to write the adaptation rules.

The written adaptation is saved in XML or in the AHA! format that is transmitted to the AHA! adaptation engine as input in addition to the domain model.

Graph Author Tool

As described before in Section 2.1, the Graph author tool [23] is similar to the Concept editor [23]. It helps to instantiate the domain model included in the Concept editor, to specify user characteristics and generates automatically adaptation rules. Here, we focus on the parts related to the generation of adaptation.

The graph author tool includes a set of predefined templates that are used to automatically generate adaptation rules. These templates exploit relations included in the Graph author tool. Consequently, adding new types of relations between concepts require to write manually the associated adaptation. This is why, this process is only let for technical persons.

Note that, AHA! does not consider the goal of a user. This means that, existing templates used to generate adaptation do not include such processing. Consequently, authors have to write manually their adaptation for considering the goal of a user.

The adaptation is generated in XML or in the AHA! format and will only be executed by the AHA! adaptation engine.

JSEM-HP

In Section 2.1, we have detailed the implementation of the conceptual and presentation subsystems. In this section, we focus on the aids proposed by the navigation and the learning subsystems in JSEM-HP [48].

The navigation subsystem automatically calculates a set of navigational paths based on the semantic net. The authors have at their disposal a graphical tool in order to modify the generated rules or eventually to add new ones. They do not have to write rules but they can modify the condition or conclusion part of a rule in a visual manner.

The learning subsystem implements three distinct parts.

- The user model. A graphical interface is added to the tool in order to define user characteristics. Remember that, user characteristics include data, like: knowledge on items, goals, personal information and preferences. A dialogue interface is proposed for specifying user knowledge. This allows to specify the user experience in navigation/ subject using one of the four allowed values. This is viewed as a limitation if authors want to consider more values.
- The rules for updating user knowledge. An editor is proposed with a set of constructors to define updates. An update can be triggered only after visiting an item.
- The adaptation behaviors. It is not clear what type of adaptation can be expressed. There is no comparison with the Brusilovsky typology.

2.2.2 Solutions compatible with numerous Adaptive Hypermedia Systems

LAG - MOT adapt (its future the PEAL tool)

LAG is considered as the first generic adaptation language. With the help of this language, which has a context-free grammar [17], an adaptation engineer can construct LAG programs[24]. A LAG program contains two parts:

- An initialization part: the aim of this part is to enable authors to specify explicitly the types of resources to be proposed to users at their first access. Therefore, the initialization part is executed only once for each user, and all its instructions are processed in a sequential way. User characteristics are described in a user model. There are a set of attributes, either free variables or overlay variables, the latter dependent on the domain model. For simplification, the user model specification is part of the LAG program. Thus, authors can also include the initialization of free variables or overlay variables, the latter dependent on the domain model that will be used. From a different viewpoint, we can compare the initialization part to a constructor in an object oriented programming language.
- An implementation part: the aim of this part is to enable authors to specify what type of resources should be proposed to users, after each of their interactions with the adaptive hypermedia. Therefore, the program code contained in is evaluated after each user action, on all the resources available at runtime in the adaptive system. Using again the object oriented paradigm, the implementation part could be implemented as a *while true do <set-constructors>* loop which is executed after each user action, where <set-constructors> is the program code of the implementation part.

The grammar is based on a set of constructors[25], in order to facilitate the expression and the reuse of adaptation. The constructors allow to manipulate elements of the domain, the user

[24] Note that, a LAG program may include at the sime time one or more adaptation strategies.
[25] The syntax and the semantic of LAG constructors are described at *http://www.dcs.warwick.ac.uk/ acristea/mot.html*.

and the presentation models. Most of them can't be used alone, and they have to be combined together. Among the proposed constructors we mention:

- `while`: allows defining a loop on a set of constructors. It has the following syntax: `while <condition> (<set-constructors> ).` The <set-constructors> are performed as many times as the <condition> part is evaluated to true.
- `DM`: gives access to information modeled in the domain model. It has to be associated with a constructor giving access to the modeled information. For instance, checking if a concept attribute is of type example can be written as:
  `if (DM.Concept.attribute.type == example)`
- `Concept`: gives access to resources. For example when associated with the constructor DM, such as in `DM.Concept`, it gives access to resources modeled in the domain model.
- `GM`: gives access to information modeled in the goal model, and thus to pedagogical information. It has to be associated with a constructor specifying the desired modeled information. For instance, checking if a specific concept in the goal model is labeled as visual, can be done as follows:
  `if (GM.Concept.label == visual)`
- `PM`: gives access to information modeled in the presentation model. The presentation model describes the interaction of the user with the display screen: the resources currently to be shown to the learner, as well as the way they are to be shown. For simplification, the presentation model is represented as part of the LAG strategy. Thus, the presentation information has to be associated with a constructor giving access to the modeled information. In other words, a concept can be shown by a statement such as:
  `PM.GM.Concept.show = true`
- `show`: allows to propose or not a resource to the user. It has to be combined with at least another constructor. For example, `PM.GM.Concept.show` allows proposing resources of the goal model.
- `UM`: gives access to knowledge modeled in the user model. It has to be associated with at least one constructor giving access to the modeled knowledge. For example, it can be combined with the DM constructor: `UM.DM` gives access to user knowledge dependent of the domain model.

Note that, the adaptation and user models are merged together in LAG. Consequently, the author has also to specify the user characteristics in the same time when specifying the adaptation.

When several LAG programs are defined, they can be combined together [61]. Currently, only priority between LAG programs is supported, LAG programs can't be merged together unless they are rewritten. This is due to the fact that, several problems may occur when combining LAG programs, including:

- *Execution order*: in which order the LAG programs have to be considered. The results of execution may be different if LAG programs are reversed, in particular for the initialization part as it is executed in a sequential way.
- *Variable clashes*: this mainly happens when the same variable is used in more than one LAG program. The variable may be updated several times (once in each LAG program).
- *Type conflicts*: this concerns the fact that one variable may be defined differently in numerous LAG programs.

The adaptation written in LAG is expressed at a high level, and currently it can be executed by ADE [61], AHA! [14], WHURLE [49] and blackboard [54] adaptation engines.

LAG-XLS

LAG-XLS [67] is also an instantiation of the LAG model (cf. Section 1.1.2) and a generic adaptation language. However, it is limited to be used with learning styles.

LAG-XLS implements the external actions which can be defined in an adaptation strategy (cf. Chapter 4, Section 1.1) using constructors. There are constructors to select, to show, to order resources or to exploit navigational paths. The use of these constructors is described in a DTD. We are not going to present this DTD here, instead of that, we present a figure reviewing its main elements and attributes (cf. Figure 2.2) extracted from the thesis of Stash [66].

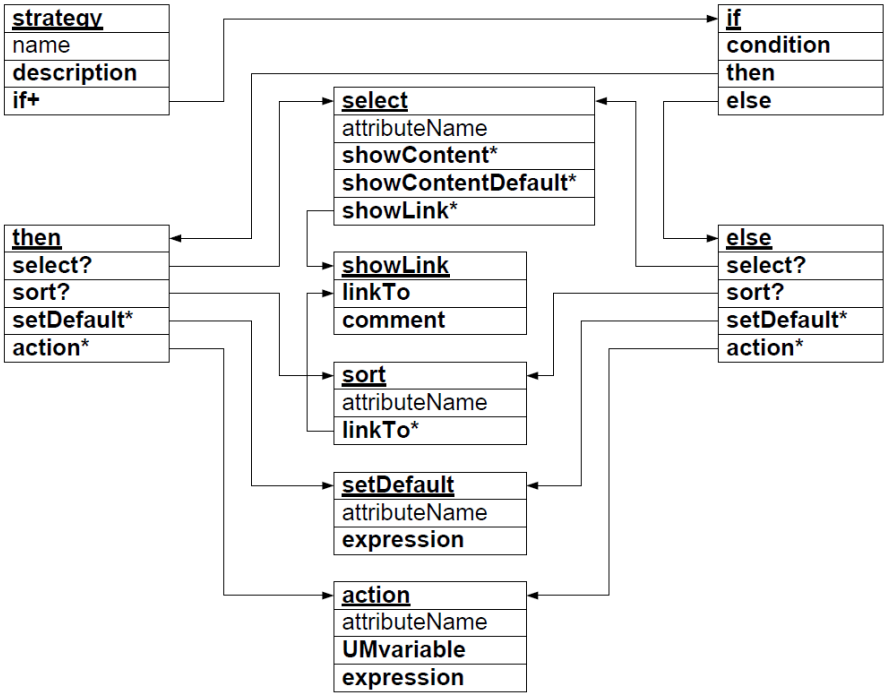

Figure 2.2 – Structure of the elements and attributes of the DTD defining LAG-XLS [66]

Note that, the name of the elements and attributes used in the DTD are very intuitive. They allow authors to easily express their actions. The element `strategy` is the root element. It refers to one adaptation strategy in LAG-XLS. It has an attribute `name` referring to the name of the adaptation strategy, an attribute `description` referring to its textual description in natural language and at least one element `if`. The element `if` implements the usual if in an imperative programming language. Therefore, it is composed of an attribute `condition`, elements `then` and `else`. The elements `then` and `else` have the same structure. They allow to select resources using the element `select`, to order resources using the element `order`, to set defaults value using the element `setDefault` and to express actions on the learner characteristics using the element `action`.

LAG-XLS proposes also meta-strategies allowing to infer learner characteristics according to learner interactions with the system.

Consequently, as the LAG language, LAG-XLS does not separate adaptation and user models. However, some differences subsist between both languages. Note that, the two languages have never been evaluated together. Each language has been evaluated separately with students. The evaluation and resulted discussions about LAG-XLS can be found in [68]. On the other hand,

the LAG language has been evaluated several times as it evolves, some of its evaluations and discussions can be found in [11], or in [30]. We resume the differences in the following.

- Initially, one of the advantages of using LAG-XLS instead of the LAG language in education was the fact that LAG-XLS expresses adaptation in XML but not LAG. However, today even LAG allows to write adaptation in XML.
- Another advantage of LAG-XLS concerns its syntax, which is assumed to be easier than the syntax of the LAG language. It doesn't include elements such as `WHILE`, `FOR` which are more for technical authors.
- LAG-XLS has been proposed in order to be completely compatible with the AHA! adaptation engine, no conversion to other adaptation engines has been proposed so far. One the other hand, the LAG language has conversion to AHA!, WHURLE and blackboard, etc.
- Since 2007, LAG-XLS has not been extended, whilst the LAG language is regularly enriched by new constructors.

GAL - GRAPPLE Adaptation Language

The GRAPPLE Adaptation Language, often abbreviated in GAL [65], argues to gather, on the one hand, all the functionalities of existing adaptation engines, and on the other hand, to be an intermediate language between existing authoring environments and adaptation engines. For this purpose, GAL plans to include translators from existing authoring environments to GAL and from GAL to existing adaptation engines.

However, GAL supports only the adaptive navigation. By adaptive navigation, GAL considers the structure of web pages and relations between them. Therefore, the author using this language has a vision of buiding a set of linked web pages.

The navigational structure of a web application is defined using abstract constructors, organized in a hierarchy of GAL constructors. We are not going to cite all the possible constructors [64]. We only cite some of them in the following:

- `unit`: it allows to group together the elements that will be shown together. Concretely, they can be seen as the definition of a web page.
- `subunit`: it allows to embed units into another unit.
- `attribute`: it allows to define characteristics of a unit. For example, the name of a particular unit.
- `query`: it allows to exploit information included in the domain or the user model. It is expressed using the SPARQL[26] language.
- `link`: it allows to define hierarchical links between units (a sort of composition relation).
- `order`: it allows to define a partial order between non related units. The order is defined using a variable of integer type. The unit with the lowest order is considered first. When more than one unit have the same order, they are randomly arranged.
- `updateQuery`: it allows to update information included in the user model, or the temporarily variables specified inside the units.

[26] www.w3.org/TR/rdf-sparql-query/

These GAL constructors can be composed together in order to define the structure of web pages and links between them. They can also be used to specify the following types of adaptive navigation of the Brusilovsky typology:

- *Adaptive multimedia presentation*: GAL allows to select several media according to different constraints. The constraint satisfied by the user determines the suitable media to be proposed to the user.
- *Adaptive text presentation*: it is similar to the adaptive multimedia presentation but only applied on textual resources. This means that, GAL allows to select alternative textual resources according to different constraints. The constraint satisfied by the user determines the suitable media to be proposed to the user.
- *Direct guidance*: GAL allows to express direct guidance by embedding different GAL units.
- *Link sorting*: this is directly done using the constructor *order*.
- *Link hiding*: this is done by defining constraints on the links to be shown. The none selected links are not hidden but presented in a light gray.
- *Map adaptation*: it is used to generate menu structures and is defined using the hierarchical links between units.

Note that, GAL does not allow to express annotation of links, because the authors of GAL place annotation of links as presentation issues and they do not want to consider such issues.

Consequently, GAL proposes a new vision of building adaptation, particularly of the adaptive navigation. Comparing with LAG, we cite the following differences:

- GAL proposes a web page vision (note that an adaptation strategy may result in a composition of web pages), whilst, LAG proposes to write a program for each adaptation strategy.
- GAL proposes the separation between the specification of the user model and the adaptation model, which is not the case when using LAG.

Besides, GAL has some similarities with LAG. In fact, similarly to LAG, the generated adaptation strategies using GAL are considered as a whole block and can not be easily reused as they are embedded in one or several web pages.

This later point is considered as a drawback when thinking about adaptation strategy granularity. GAL does not propose an ideal granularity for facilitating reuse of existing GAL adaptation strategies. Another drawback that we can cite using GAL is the fact that authors have to write a GAL program (use of SPARQL[27] queries to select resources) in a sequential way and no aid is proposed for them yet.

CAM - Conceptual Adaptation Model tool

The CAM tool [37] aims to be a tool demonstrating the possibilities of the CAM model (cf. Section 1.1.2). It has been designed in order to be a system-independent tool and to allow a high level modeling. It focuses on graphical possibilities for authoring AH in order to be as intuitive as possible.

One or more CRT files (as described before in Section 2.1.2) can be added in the CAM tool. For each CRT file, authors have to instantiate its elements on instances of the domain model.

[27] www.w3.org/TR/rdf-sparql-query/

Initially, the CAM tool has to integrate a translator to GAL, in order to allow adaptation specified using CAM be executed by all the adaptation engines accessible from GAL. However, lastly, this idea was abandoned and a direct translator was implemented from CAM to GALE (GRAPPLE Adaptation Learning Environment)[28]. Therefore, the adaptive hypermedia defined using GAT (the Graphical authoring tool to express the domain model. See Section 2.1.2) can be directly executed by GALE, which is a generic (for e-learning) and an extensible adaptation engine. GALE is assumed to support all existing types of adaptation[29].

2.2.3 Summary

Up to here, we have presented solutions enabling authors to express adaptation. However, as it has been shown, some of these solutions remain specific for computer science authors (like, LAG, GAL), whilst, others try to be more visual and therefore propose more guidance. However, they lack in providing information on what type of adaptations authors can express. Except expressing the link sorting/ordering of the Brusilovsky typology, no indication is given for the other types of adaptations. Consequently, till today, expressing adaptation is not possible for all authors.

3 Relative position in comparison with existing work

After having presented models used to describe AH in Section 1, we have noticed that an AH is composed of several parts. It necessarily includes domain, user and adaptation models. These models have been implemented in AHS in order to support authoring and execution of AH.

However, as shown in Section 1.2. authoring AH using directly an AHS requires from authors several backgrounds which often they haven't. In fact, it requires modeling skills to express domain and user models and logic or programming skills to express adaptation.

Recently, solutions assisting authors in the design process of their AH have been proposed. They concern modeling domain model or expressing adaptation.

Concerning the modeling of domain and user models. Most often authors of AH have already their domain and user models with their instantiations, and they would like to reuse them in their AH. For example, in the case of AEH, authors have already their courses and their learners' characteristics, and they would like to propose an AEH that exploits their materials.

For this purpose, authors use existing AHS with which they define and execute their AH. However, each AHS includes domain and user models that authors have to reuse when defining their AH. One question arises *how could an author integrates his domain and user models (specially instances of his models) into the domain and user models of the used AHS?*

Existing solutions presented in Section 2.1 propose to authors an instantiation process. This means that, their models and their instances would be defined as instances of the models of the used AHS. The instantiation process is performed twice: once on domain models and once on the user models. However, these solutions require to transform authors' models and their instantiations, which is an issue if some of authors' models cannot be transformed. Furthermore, the added instances are only a copy in the domain and user models of the used AHS, which is an issue if the original instances are updated.

We propose in Part III a new solution considering both the author's domain and user models and also their instances. The instances are neither transformed nor modified. Our solution allows easily and quickly to plug into existing AHS using a specialization process. Thereby, adaptation defined on the domain and user models of the used AHS can be directly executed on

[28] GALE is based on the AHA! system.

[29] Note that, we did not find a reference proving this, for this reason, we said *it is assumed to*.

a specialization of these models (on classes, properties or relations). Note that, defining a model as a specialization of another model is also useful in other applications, like to exploit SPARQL[30] queries expressed on an existing ontology. Indeed, defining an ontology as a specialization of the existing one allows to reuse the SPARQL queries.

By this process, our objectives are twofold: on the one hand, to create a support for defining mappings between elements of the model included in an existing system and elements of the author model, and on the other hand, to help creating consistent and relevant models integrating (1) the model included in an existing system and (2) the author's model and (3) taking into account the mappings between them.

Concerning the expression of adaptation. Most often adaptation AHS is expressed using condition-action rules or event-condition-action rules. It is considered as the less intuitive part to be authored in AH by non technical persons (like: teachers who have not the required logic or programming skills).

Existing solutions presented in Section 2.2 propose graphical tools or languages using constructors in order to support the expression of adaptation. The graphical tools integrate predefined templates to generate adaptation. However, when authors add new information that are not considered in predefined templates (as the goal of users), they have to write manually adaptation. Furthermore, authors have no control on the types of adaptation to propose to their users. On the other hand, languages using constructors remain difficult to use as they require from authors to have skills in programming language.

Among the existing solutions, some of them are expressed at a high level, independently of any adaptation engines. Therefore, they integrate translators to existing adaptation engines. However, any of the existing solutions has faced the problem of expressing adaptation strategies at a fine granularity, and therefore make easier their reuse.

We propose in Part IV a new framework addressing three challenges. It concentrates (1) on the ease of defining adaptation strategies at a high level and at a fine granularity, (2) on combining them and (3) on the facility of reusing existing adaptation strategies.

Afterward, we have conducted a study of the expressivity of our framework versus some existing AHS. We have studied the expressivity of their modeling of adaptation and also the expressivity of their modeling of the elements on which the adaptation is expressed. From these studies, we have come up with an integrated vision of the basic actions that can be used in defining adaptation, and also with an unifying vision of modeling the domain model. These studies have served us to identify their similarities and thus to propose translators from our framework to existing AHS. We have also highlighted their main modeling differences.

[30] www.w3.org/TR/rdf-sparql-query/

Part III

Assisting in the reuse of author's domain and user models

CHAPTER 3

Integrating author's models into models of Adaptive Hypermedia Systems

In this part, we propose a new solution allowing easily and quickly to plug author's models and also their instances into existing systems using a specialization process.

By this process, our objectives are twofold: on the one hand, to create a support for defining mappings between elements of the model included in an existing system and elements of the author model, and on the other hand, to help creating consistent and relevant models integrating (1) the model included in an existing system and (2) the author's model and (3) taking into account the mappings between them. The process is seen as a specialization process generating specialization and equivalence mappings. It is inspired of schema/ontology mapping in knowledge engineering. Consequently, we present some close works. This merging/specialization process is generic, and can be applied on any specific and generic models. For this purpose, it relies on a meta-model, and reasoning is expressed in a declarative way[1] [33] using SWRL rules[2].

We implemented this process as a Protégé plug-in[3] (cf. Chapter 7, Section 1) and have accompanied it with experimentations in the e-learning application domain (cf. Chapter 8, Section 1).

This has been presented at three international conferences: a conference specific for AHS in *AH'2008* [79], a conference on knowledge engineering in *EKAW'2008* [80], and a conference on the Protégé tool in *Protégé'2009* [78].

The reminder of the chapter is the following. In Section 1, we describe succinct close work in order to position and understand our contribution. Afterward, in Section 2, we present the main aspects of our merging/specialization process including an illustration on a use case. Then, we describe successively the different steps of the merging/specialization process in Section 4, 5 and 6. Each of these sections, includes an illustration on a use case. Finally, in Section 7, we summarize the advantages of our process.

Details on the implementation and experimentations of the presented process can be found successively in Chapter 7 and Chapter 8.

[1] According to *John mcCarthy*, who is a pioneer in Artificial intelligence, there are several advantages to declarative systems (1) simplicity of use for humans, (2) overall computational efficiency, (3) support for collaboration in heterogeneous systems, etc.

[2] http://www.w3.org/Submission/SWRL/

[3] The plug-in is available at http://protegewiki.stanford.edu/wiki/MESAM/

1 Related work in the knowledge engineering field

In the knowledge engineering field, research on the integration of heterogeneous information systems has been put in perspective since several years. There are several approaches for performing a semantic integration depending on the degree of integration usually referred as ontology [34] mapping, merging or enrichment.

Ontology mapping aims to define correspondences between different entities of the ontologies (at least two) involved in the ontology mapping process. Generally, the ontology mapping process consists of three phases (cf. Figure 3.1) [55]

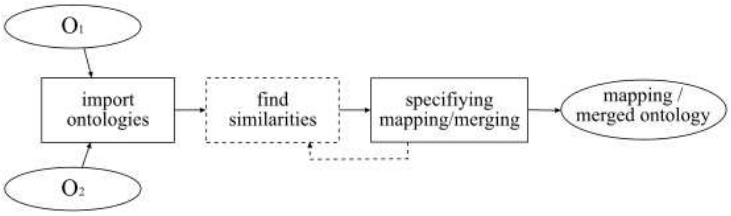

Figure 3.1 – The Ontology Mapping Process [55]

1. *Import of ontologies*. It enables users to import ontologies expressed in different languages. The ontologies are converted to a common format in order to perform the next phases.
2. *Calculate similarities*. Most of existing approaches try to make this phase as automatic as possible and several of them use a Match operator to calculate similarities between ontologies [56].
3. *Specify mapping or merging*. The similarities calculated in the previous phase are used to define mappings between entities of the ontologies involved in the process. Mostly, it is an interactive process. Let us take as illustration the I-PROMPT system. I-PROMPT automaically generates a set of mappings. Then the humain decides whether to retain each mapping or not. After each choice, a new set of mappings is generated again and has to be validated again.

Schema or ontology mapping is often an intermediate step in a more global process. So several approaches [46, 71, 24, 52, 44, 28] propose then to merge the ontologies involved in the mapping process. The merged ontology must unify the ontologies involved in this mapping process and has to be coherent.

The mapping ontology can also be a prerequisite for the enrichment of ontologies. In that case, one ontology is preferred and new elements comming from another one are added.

In the following, we get interest for approaches proposing both mapping and merging processes, and the solutions to deal with inconsistencies. Note that, we do not consider neither scalability nor redundancy. We are not interested in these problems in our work.

Surveys through the last years have been provided [45, 55, 57]. Existing approaches are either based on instances of the two given ontologies that are to be mapped (*bottom-up* approach) or on concepts (*top-down* approach). We describe below each approach.

1.1 Approaches based on merging models according to a bottom-up approach

Here, we present only the FCA-MERGE [71] framework which presents the main principles of approaches based on merging models according to a bottom-up approach.

The FCA-MERGE [71] framework proposes a semi-automatic process to build a merged ontology. It considers as input (1) two ontologies (O_1, O_2 cf. Figure 3.2) assumed to be related to the same domain and (2) a set of natural language documents (D cf. Figure 3.2) pertinent for both ontologies. The process consists of three steps (cf. Figure 3.2).

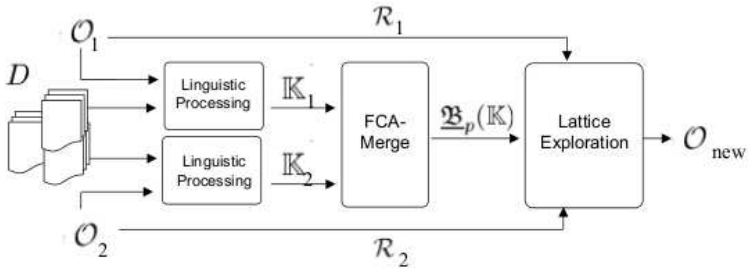

Figure 3.2 – Ontology Merging method [71]

1. *Instances extraction and computation of two contexts (K_1, K_2 cf. Figure 3.2).* This step is automatic. Instances are extracted from the set of documents and a context is built per ontology. Each context contains associations between instances of concepts in the ontology and the documents to which they belong.

2. *Deduction of one context from the two contexts and construction of a concept lattice.* This step is automatic thanks to the FCA-Merge core algorithm, which allows to build a context from two other ones. After that, a concept lattice is constructed using FCA techniques (more details can be found at [72]. The main idea is that, FCA techniques first calculate concepts to be included in the concept lattice. A concept gathers all instances of K_1, K_2 having the same number of associations and identical associations. Thus, they define as many concepts as instances with different associations. Afterward, FCA techniques generate the concept lattice. This includes (1) organizing concepts in different levels depending on the number of associations of their instances, i.e., concepts having instances with the same number of associations belong to the same level. It also includes (2) defining links between concepts of different levels. Concepts of the same level are not linked. Concepts of a level are linked to concepts of the next level if they include instances of the same context.

3. *Construction of the merged ontology from the concept lattice.* This step is semi-automatic. The expert accesses the concept lattice and the sets of relations (R_1, R_2 cf. Figure 3.2), in order to create concepts and relations in the merged ontology (O_{new} cf. Figure 3.2). The concepts emerge from the content of the concepts in the concept lattice for which the user is asked to propose a name.

This framework has been integrated to the OntoEdit environment[4].

1.2 Approaches based on merging models according to a top-down approach

There are several approaches supporting merging models based on a top-down approach [59, 46, 24, 42, 51, 28]. They can be divided in two groups: approaches proposing to merge ontologies by creating new classes as in I-PROMPT [51] and approaches making the union of the classes of ontologies involved in the merging process as OntoMerge [28]. In the following, we present one system belonging to each approach.

[4] www.semtalk.com/semnet_files/POntoEdit.htm

PROMPT [51] suite includes a set of tools supporting ontology mapping (Anchor-PROMPT), merging (I-PROMPT), versioning (PROMPT-Diff) and factoring out semantically complete sub-ontologies (PROMPT-Factor). These tools have been developed such as they can be used together or independently. Each of them is proposed as a plug-in of the Protégé tool[5]. For our needs, we focus only on I-PROMPT[6], which is an interactive tool for merging two ontologies (cf. Figure 3.3). On Figure 3.3 the gray boxes are performed by the I-PROMPT and the white boxes are performed by the user.

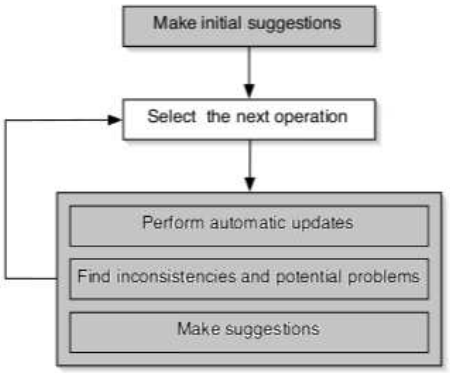

Figure 3.3 – The flow of I-PROMPT algorithm [51]

Given two ontologies, the merging process is the following.

1. *Make initial suggestions*. I-PROMPT generates a set of potential merge operations based on the two ontologies and on linguistic similarity measures[7] based on the class names. Suggestions concern merging similar class names of the two ontologies by creating a new class or copying a class from one of the two ontologies.

2. *Select the next operation*. The user is invited to select one potential merge operation or to add a new operation. The allowed operations are *a deep or a shallow copy of a class*, *merge classes*, *merge attributes*, *merge relations* and *merge instances* [52]. The user may select an operation and classes (or relations or attributes) on which the operation will be performed.

3. This third step includes three different tasks:

   (a) *Perform automatic updates*. I-PROMPT performs the selected operation, for example the *merge classes* operation into a new class, which is followed by the following actions:
   - specialization (and generalization) links in which the class participates. If these links establish relations with classes that don't belong to the merged model, these classes are added before adding specialization (and equivalence) links.
   - all relations and attributes whose domain is a class in the merged ontology are added.

   (b) *Find inconsistencies and potential problems*. I-PROMPT checks for inconsistency problems from a list of possible inconsistencies (cf. Table 3.1). For each possible inconsistency, the tool has a predefined solution.

   (c) *Make suggestions*. I-PROMPT exploits the merged ontology in order to make new suggestions of classes, attributes, relations or instances.

[5]Protégé tool is available at *protege.stanford.edu/*

[6]PROMPT is available at http://protege.stanford.edu/plugins/prompt/prompt.html

[7]It uses simple lexical-distance measures to find classes with similar names. Other term-comparison algorithms can be added.

After that, the process comes back to the step: *Select the next operation*.

Inconsistencies	Meaning in the merged ontology	Solution in I-PROMPT
Name conflicts	Elements[8] having the same name but being different.	The user is invited to rename elements in conflict.
Dangling references	An element refers to another which doesn't belong to the merged ontology.	The user is invited to add the missing reference.
Redundancy in the class hierarchy	There is more than one path from a class to a superclass	The user is invited to remove one of the offending parents.
Instances violating restrictions	Instances don't follow range or cardinality constraints	Instances are deleted from the merged ontology.

Table 3.1 – List of inconsistency problems and their resolution in I-PROMPT

OntoMerge [28] proposes a solution to merge two ontologies by considering the union of both ontologies. It uses namespaces in order to avoid conflicts between names of elements of both ontologies. Furthermore, it defines bridge axioms between related elements of the two ontologies involved in the merging process. Therefore, the merged ontology includes the two ontologies involved in the merging process and also bridge axioms.

OntoMerge uses an internal representation for performing the merging process. It is based on a typed first-order logic language, called Web-PDDL (Web-Planning Domain Definition Language), which allows to capture several ontology languages. It proposes import and export functionalities of DAML + oil and OWL ontologies.

OntoMerge does not support matching. That means that, bridge axioms have to be written by users (domain expert with first-order logic skills) or calculated using matching algorithms.

Despite this latter disadvantage, OntoMerge aims to be used in tasks like: dataset translation, ontology extension generation or querying different ontologies.

1.3 Summary

In the engineering field, existing systems are either based on instances or on concepts. They are either automatic or interactive tools. However, despite they have been used with many different ontologies and in many different domains, according to me they have not been used to merge abstract models with specialized ones and have not been applied to the AH field.

In our work, we focus on these specific points. The models to be merged are relatively small and don't raise scalability problem. The merging process is performed once at design time. Generic models are composed of abstract classes which have no instances. The author of the system knows the models to be integrated in the system very well and can then easily provide simple correspondences between their elements. Given these initial correspondences, the software implementing our approach is expected to reason on the models and to generate additional correspondences between classes, attributes or relations. The hypothesis underlying this work is that it is easy for authors to specify simple correspondences between classes and then evaluate mappings returned by the system. However, the consistency of the merged model has to be automatically checked.

So we propose a semi-automatic merging and top-down approach, that has been motivated by the context of the AH field. Nevertheless the approach is generic and can be applied to other applications. Our aim wasn't to propose ad'hoc solutions.

[8] An element may be a class, a relation or an attribute.

2 Main aspects of our merging/specialization process

We propose a generic approach that can be applied on any two models. We define a model as follows:

Definition 1 *a model is composed of a set of classes, of properties and of relations between classes.*

We describe in Section 2.1 the characteristics of the two models used by our process, and in Section 2.2 the characteristics of the model built by the process. Afterward, we present in Section 2.3 the steps that have to be performed by our merging/specialization process. Finally, we illustrate in Section 2.4 this process on a use case.

2.1 Characteristics of the two models used by the process

The process is performed on two models.

A generic model. It is any model that can be specialized by another model. The *generic model* has no instances.

A specific model. It has to be of the same application domain as the *generic model*. The *specific model* has instances. Furthermore, the author has a good understanding of the *specific model*.

Our process proves it utility in the case of systems including reasonings expressed on models, which authors would like to reuse but by making them executed on their own models. Our process enables authors to integrate their own models into these systems. For this purpose, it assumes that authors' models include equivalent or more restrictive elements than those included in the models on which reasonings are expressed. Once the process is performed, all elements of models on which reasonings are expressed are related to authors' models. Consequently, existing reasonings can directly be executed on authors' models and their instances.

A concrete application of this process is in the AH field, where currently there are numerous adaptations expressed on domain and user models. Furthermore, authors have most often their own domain and user models and they would like to reuse defined adaptations on their own models. Our process enables authors to integrate their domain model into the domain model on which adaptations are defined and similarly for the authors' user model.

For example, in the AH field, the well known AHAM model (cf. Chapter 2, Section 1.1) proposes a domain model and assumes also a set of predefined adaptations exploiting this domain model. On the other hand, the domain model proposed by *Graph Author Tool* (cf. Chapter 2, Section 2.1.1) is an implementation of the AHAM domain model. A way to reuse predefined adaptations of the AHAM model can be to perform our merging/specialization process. The AHAM domain model could be considered as a generic model, while the domain model of the *Graph Author Tool* could be considered as a specific model. That way, predefined adaptations defined in AHAM could be performed on instances of the domain model of the *Graph Author Tool*.

2.2 Characteristics of the built model

The result of the merging/specialization process is a merged model (cf. Figure 3.4) that integrates at the same time: the generic model, the specific model, and mappings between elements of both models.

By a *mapping*, we mean a relation linking two elements of the two different models. The relation can only be defined between two elements of the same type, which are either classes,

properties or relations. Furthermore, as our aim is to define a model as a specialization of another, we only consider two types of mappings: equivalence and specialization mappings[9].

The mappings are validated at the semantic and structural level. For this purpose, our process relies on the author who has a very good understanding of his models and of the application domain of his models. He will be responsible of the semantic validation while all the structural verifications will be done automatically.

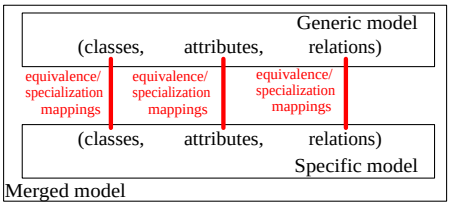

Figure 3.4 – General overview of the merged model

2.3 The merging/specialization process

The main steps of the merging/specialization process are the following (cf. Figure 3.5):

1. Specification, by the author, of equivalence and specialization mappings between classes of the generic and the specific models, merging the generic model and the mapped classes of the specific model (together with the associated mappings) in order to obtain a new model (cf. (1) Figure 3.5).
2. Automatic computation of additional mappings between classes, the mappings and the linked classes being added in the being built model (cf. (2) Figure 3.5).
3. Structural verification. It automatically computes mappings between elements different from classes. This includes mappings between properties and between relations (cf. (3) Figure 3.5).
4. Validation by the author of the deductions made by the system in step 3 (cf. (4) Figure 3.5).

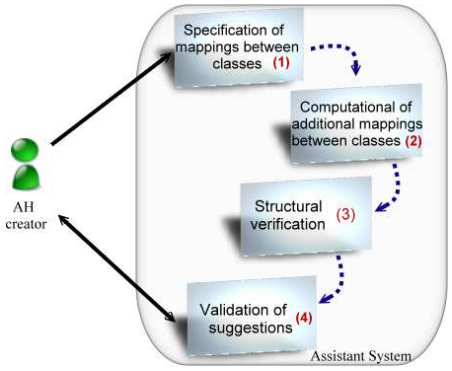

Figure 3.5 – The architecture of the proposed merging/specialization process

The process uses computation in the steps 2 and 3 to make deductions. That means that, generic and specific models have be described in a language supporting such computations. The process supports OWL, more particularly OWL-Lite or OWL-DL models. We do not consider

[9]We present further in this chapter definitions associated to each type of mappings.

OWL-Full models whose reasoning is not decidable. This choice is not restrictive as today OWL-DL is widely used to express models. Furthermore, it is a W3C recommendation[10] since 2004.

The four steps are described successively in Sections 3, 4, 5 and 6. Furthermore, we adopt the following notations:

- $C_{m,i}$ to represent the class i from the model m.
- $R_{m,d,j}$ to represent the relation j with domain d in the model m.

2.4 Applying the merging/specialization process on John's use case

Here, we apply our process on an example related to the AH field. We assume that John, who is a lecturer in computer science, has his domain model and resources belonging to that model. Furthermore, he would like to use the GLAM platform and the defined adaptations without transforming his resources.

These adaptations are expressed on domain and user models respecting the GLAM syntax (cf. Table 3.2). They concern students[11] whose learning mode is *in-depth* and prefer *audio* resources. It proposes resources 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*.

The adaptation is defined in three steps (for GLAM syntax see Chapter 2, Section 1.1.1):
Step 1: GLAM rules • R1: *type (r, Resource)* $\wedge$ *format (r, audio)* $\wedge$ *abstraction(r, Concept1)* $\wedge$ *abstraction(currentR, Concept2)* $\wedge$ *pre-requisite (Concept2, Concept1)* $\wedge$ *pre-requisite* (Concept1, goal)* $\rightarrow$ *Read(r, degree)* • R2: *type (r, Resource)* $\wedge$ *format (r, text)* $\wedge$ *abstraction(r, Concept1)* $\wedge$ *abstraction(currentR, Concept2)* $\wedge$ *pre-requisite (Concept2, Concept1)* $\wedge$ *pre-requisite* (Concept1, goal)* $\rightarrow$ *Read(r, degree)* • R3: *type (r, Resource)* $\wedge$ *format (r, audio)* $\wedge$ *abstraction(r, Concept1)* $\wedge$ *pre-requisite* (Concept1, goal)* $\rightarrow$ *Read(r, degree)* • R4: *type (r, Resource)* $\wedge$ *format (r, text)* $\wedge$ *abstraction(r, Concept1)* $\wedge$ *pre-requisite* (Concept1, goal)* $\rightarrow$ *Read(r, degree)* R1 proposes audio resources according to a depth-first navigational path on their concepts using the relation *pre-requisite*. These concepts can reach the goal[12]. R3 proposes audio resources linked to concepts. These concepts can reach the goal.
Step 2: associations between rules and user characteristics: • The value *audio* of the user characteristic *presentation form* is associated to R1, R3 • The value *textual* of the user characteristic *presentation form* is associated to R2, R4 • The value *depth-first* of the user characteristic *learning mode* is associated to R1, R2, R3, R4
Step 3: GLAM meta-rules • MR1: The value *audio* of the user characteristic *presentation form* $>$ The value *textual* of the user characteristic *presentation form* • MR2: R1 $\supset$ R3 • MR3: R2 $\supset$ R4 MR1 means that all rules associated to the characteristic presentation form audio are executed. When no results are returned, rules associated to the characteristic presentation form text are executed.

Table 3.2 – Example of adaptation in the GLAM format

[10] www.w3.org/TR/owl-guide/
[11] cf. Table 3.3
[12] The goal to be reached by users (students here) is modeled as a property.

In order to reuse these adaptations, both domain and user models on which adaptations are expressed have to be integrated with John's models. In the following, we focus only on the part relative to the domain model. First, we describe the domain model on which adaptations are defined and which is considered as generic by our process. Then, we present John's model, which will be considered as specific by our process. Finally, we propose an overview of the merged model that should be obtained after performing our process.

learning mode: *in-depth* learning mode means that each subject must be known in-depth before going to a related subject. **presentation form:** a *verbal* presentation form is for learners preferring textual resources, a *visual* presentation form is for learners preferring image resources, and an *audio* presentation form is for those preferring audio ones.

Table 3.3 – Students characteristics

2.4.1 Description of the generic model

We present in Figure 3.6 the modeling of the generic model (i.e., the domain model on which GLAM adaptations are expressed). The generic model includes only two classes:

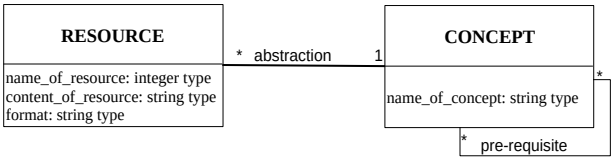

Figure 3.6 – Description of the domain model exploited by the adaptation in Table 3.2

- A *CONCEPT* class models abstract notions that must be proposed to users. Concepts can be pre-requisite of other concepts[13]. Each concept has a name (modeled in the *name_of_concept* property) and may be related to one or more resources by the relation *abstraction*.
- A *RESOURCE* class models documents that will be proposed to users. A resource is characterized by a name (modeled in the *name_of_resource* property), a content (modeled in the *content_of_resource* property) and a format in which it is available (modeled in the *format_of_resource* property). Furthermore, each resource is only related to one concept using the relation *abstraction*.

2.4.2 Description of the specific model

Figure 3.7 describes John's domain model, and Table 3.4 indicates some instances of John's domain model.

The domain model includes several classes:

- A *NOTION* class models notions about AHS that must be learned by users. Notions can be successors of other notions and they can also be part of other notions. Each notion has a title describing the notion to be learned and may be related using the relation *abstraction* to one or more documents which are either a definition, or descriptions or illustrations.

[13]Here, names of classes are in upper-case and in italic, and names of instances have the same name as the class for which they belong in lower-case.

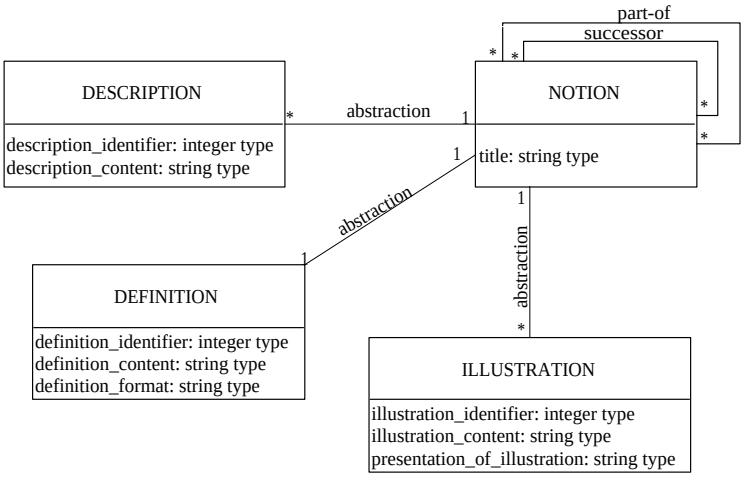

Figure 3.7 – Description of John's domain model

- A *DEFINITION* class models documents about definitions. Each document is characterized by an identifier (modeled in the *definition_identifier* property), a content (modeled in the *definition_content* property) and a format in which the sample is available (modeled in the *definition_format* property). A definition document is related to at most one notion using the relation *abstraction*.
- An *ILLUSTRATION* class models documents about samples. Each document is characterized by an identifier (modeled in the *illustration_identifier* property), a content (modeled in the *illustration_content* property) and a format in which the sample is available (modeled in the *presentation_of_illustration* property). An illustration document can be at most related to a notion using the relation *abstraction*.
- A *DESCRIPTION* class models documents about descriptions. Each document is characterized by an identifier (modeled in the *description_identifier* property), and a content (modeled in the *description_content* property). A descripion document is related to at most one notion using the relation *abstraction*.

Instance number	Class of John's domain model	Instance	Related to instances
1	NOTION	*title*: adaptive hypermedia system	2 (*via successor*), 4(*via successor*), 5(*via successor*), 7(*via successor*)
2	NOTION	*title*: architecture of AHS	3(*via successor*), 6(*via successor*)
3	NOTION	*title*: GLAM platform	7(*via successor*)
4	DEFINITION	*definition_identifier*: definition_1 *definition_content*: ... *definition_format*: ...	
5	DEFINITION	*definition_identifier*: definition_2 *definition_content*: ... *definition_format*: ...	
6	DEFINITION	*definition_identifier*: definition_3 *definition_content*: ... *definition_format*: ...	
7	ILLUSTRATION	*illustration_identifier*: illustration_1 *illustration_content*: ... *presentation_of_illustration*: text	

Table 3.4 – Some of the instances in John's domain model

2.4.3 Description of the built model

The model built by the process includes the generic model, the specific model, and the mappings between the elements of the generic and specific models. Therefore, it includes a set of classes, of properties and of relations and also mappings between elements (that are either classes, properties, or relations). We describe in Table 3.5 the classes included in the built model and the mappings defined between them by John, and in Table 3.6 the properties and relations included in the built model and the mapping deduced automatically between them.

Class of the generic model	Class of the specific model	Mapping between the two classes
CONCEPT	NOTION	equivalent class
RESOURCE	DEFINITION	equivalent class
RESOURCE	ILLUSTRATION	sub class

Table 3.5 – Mappings defined between two classes of the merged model

Generic model	Specific model	Mapping
pre-requisite domain: CONCEPT, range: CONCEPT	successor domain: NOTION, range: NOTION	equivalent property
name_of_concept domain: CONCEPT, range: string	title domain: NOTION, range: string	equivalent property
abstraction domain: CONCEPT, range: RESOURCE	abstraction domain: NOTION, range: DEFINITION	equivalent property
abstraction domain: RESOURCE, range: CONCEPT	abstraction domain: DEFINITION, range: NOTION	equivalent property
name_of_resource domain: RESOURCE, range: integer	definition_identifier domain: DEFINITION, range: integer	equivalent property
content_of_resource domain: RESOURCE, range: string	definition_content domain: DEFINITION, range: string	equivalent property
format domain: RESOURCE, range: string	definition_format domain: DEFINITION,range: string	equivalent property
abstraction domain: CONCEPT,range: RESOURCE	abstraction domain: NOTION, range: ILLUSTRATION	equivalent property
abstraction domain: RESOURCE, range: CONCEPT	abstraction domain: ILLUSTRATION, range: NOTION	equivalent property
name_of_resource domain: RESOURCE, range: integer	definition_identifier domain: ILLUSTRATION, range: integer	sub property
content_of_resource domain: RESOURCE, range: string	definition_content domain: ILLUSTRATION, range: string	equivalent property
format domain: RESOURCE, range: string	definition_format domain: ILLUSTRATION, range: string	equivalent property

Table 3.6 – Mappings defined between two properties (or two relations) of the built model

We explain in the following sections how each of these mappings can been obtained.

3 Step 1/4: specification of mappings between classes

The process has two main goals. A first goal is to build a new model that includes the generic and specific models, instances of the specific model and mappings between elements of the two models. The mappings must be sure at 100% in order to ensure a correct reuse of reasoning (reuse of adaptation, in case of the AH field) expressed on elements of the generic model. A second goal is to propose a process as automatic as possible.

Existing solutions are based on efficient algorithms to deduce mappings between two models. However, they most often need authors to validate the deduced results. Given the size of potential models involved in the process, we have decided to propose a different process. It is based on authors' knowledge on his model and on the used application domain. We argue that this will allow to avoid non pertinent results. More precisely, the process considers in Step 1 mappings between classes that have to be specified by authors, given the number of classes which is smaller than the number of relations and attributes of the generic and specific models.

3.1 Mapping between classes

In order to specify mappings between classes of both models, authors can refer to the *comments* scope of each class in which they find a description of the class. This step is manual and is supported by a tool implementing the process.

3.2 Applying the first step on John's use case

John relies on the description of each class of each model, in order to understand the role of each class. We assume that, John defines:

- an equivalence mapping between the *CONCEPT* and *NOTION* classes;
- a specialization mapping between the *RESOURCE* and *DEFINITION* classes;
- a specialization mapping between the *RESOURCE* and *ILLUSTRATION* classes;

4 Step 2/4: deduction of additional mappings between classes

Starting from the mappings between classes specified by the author, other mappings between classes can be automatically deduced. In fact, the author defines in step 1 mappings between two classes if he estimates that a class of the *specific model* is equivalent to or is a subclass of a class of the *generic model*, but some mappings are implicit for him.

For example: assume that the class *Staff* belongs to the *specific model*, and the classes *Employee* and *Person* belong to the *generic model* such as *Employee* is a subclass of *Person*. The author defines that *Staff* is a subclass of *Employee* but he may not define *Staff* as a subclass of *Person* because it is implicit for him. In these cases, the process has to make the implicit mappings explicit in order to be able to deduce all mappings between the other elements of the two models.

We propose to adopt a pattern-based process to achieve this deduction. Pattern-based processes for mapping identification across models assume that structural regularities always characterize the same kind of relations.

4.1 Pattern-based process for deducing additional mappings between classes

We have defined 8 patterns which are characterizations of structural contexts composed of 3 classes, either two classes of the generic model and a class of the specific model or two classes

of the specific model and a class of the generic model (2 categories). The idea is to deduce the nature of the relation R (equivalence or specialization) between $C_{s,1}$ a class of the specific model and $C_{g,1}$ a class of the generic model, when a third class belonging to one of the two models, $C_{m,2}$, is linked to $C_{s,1}$ by a relation R_1 and to $C_{g,1}$ by a relation R_2, R_1 and R_2 being either equivalence or specialization relations. We identified four patterns per structural context category to represent all possible cases, that is to say 8 patterns all in all.

Given R_{equiv} an equivalence relation and $R_{subClass}$ a specialization relation, the deduction of supplementary mappings is based on the composition (noted o) properties of these two kinds of relations described below:

- $R_{equiv} \circ R_{subClass} = R_{subClass}$
- $R_{subClass} \circ R_{equiv} = R_{subClass}$
- $R_{subClass} \circ R_{subClass} = R_{subClass}$
- $R_{equiv} \circ R_{equiv} = R_{equiv}$

The eight patterns we have defined are generic and usable only to identify additional mappings between classes. In order to describe them further, we propose in Figure 3.8 the following notation.

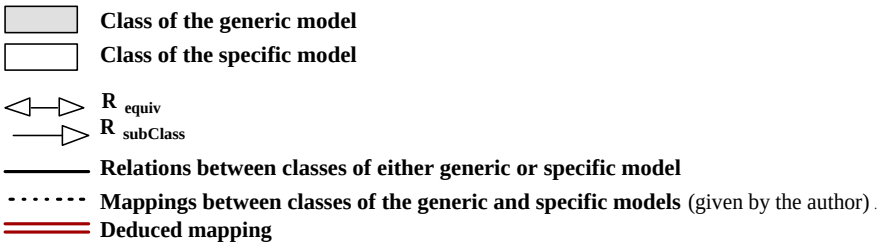

Figure 3.8 – Graphical notations

The formula $\mathbf{R}_{equiv} \circ \mathbf{R}_{subClass} = \mathbf{R}_{subClass}$ describes two patterns (cf. Figure 3.9). The pattern in the left side (respectively the right side) of Figure 3.9 covers the case where there are two classes $C_{g,1}$, $C_{g,2}$ of the generic model (respectively $C_{s,1}$, $C_{s,2}$ of the specific model) linked by a R_{equiv}. When the author defines a $R_{subClass}$ between $C_{s,1}$ and $C_{g,1}$, the pattern in the left side deduces a $R_{subClass}$ between $C_{s,1}$ and $C_{g,2}$ (respectively the pattern in the right side deduces a $R_{subClass}$ between $C_{s,2}$ and $C_{g,1}$).

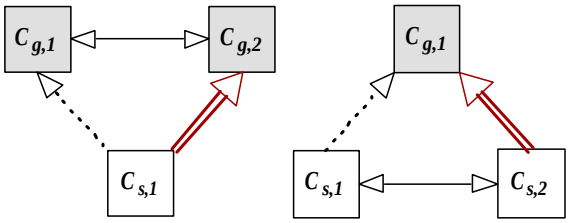

Figure 3.9 – Patterns corresponding to the formula $R_{equiv} \circ R_{subClass} = R_{subClass}$

For example, assume three classes *Woman*, *Person* and *Human*.

- In the left side of Figure 3.10, *Woman* belongs to the *specific model*, and *Person* and *Human* belong to the *generic model*. *Person* is defined as equivalent to *Human*. If the author defines

Woman as a subclass of *Person*, the formula $\mathbf{R}_{equiv} \circ \mathbf{R}_{subClass} = \mathbf{R}_{subClass}$ deduces that *Woman* is also a subclass of *Human*.

- In the right side of Figure 3.10, *Woman* and *Female* belong to the *specific model*. *Woman* is defined as equivalent to *Female*. *Human* belongs to the *generic model*. If the author defines *Woman* as a subclass of *Human*, the formula $\mathbf{R}_{equiv} \circ \mathbf{R}_{subClass} = \mathbf{R}_{subClass}$ deduces that *Female* is also a subclass of *Human*.

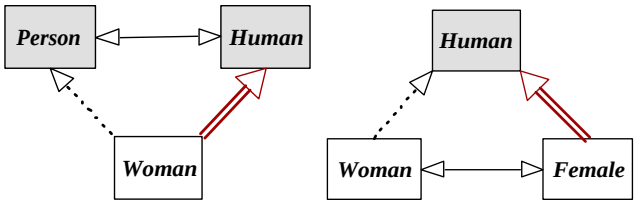

Figure 3.10 – Instantiated patterns ($R_{equiv} \circ R_{subClass} = R_{subClass}$)

The formula $\mathbf{R}_{subClass} \circ \mathbf{R}_{equiv} = \mathbf{R}_{subClass}$ describes two patterns (cf. Figure 3.11). The pattern in the left side (respectively the right side) of Figure 3.11 covers the case where there are two classes $C_{g,1}$, $C_{g,2}$ of the generic model (respectively $C_{s,1}$, $C_{s,2}$ of the specific model) linked by a $R_{subClass}$. When the author defines a R_{equiv} between $C_{s,1}$ and $C_{g,1}$ (respectively between $C_{s,2}$ and $C_{g,1}$), the pattern in the left side deduces a $R_{subClass}$ between $C_{s,1}$ and $C_{g,2}$ (respectively the pattern in the right side deduces a $R_{subClass}$ between $C_{s,1}$ and $C_{g,1}$).

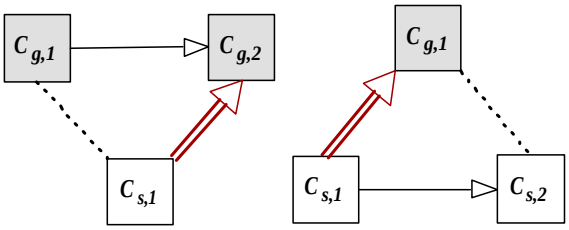

Figure 3.11 – Patterns corresponding to the formula $R_{subclass} \circ R_{equiv} = R_{subclass}$

For example, assume three classes *Woman*, *Female* and *Human*.

- In the left side of Figure 3.12, *Woman* belongs to the *specific model*. *Female* and *Human* belong to the *generic model*. *Female* is defined as a subclass of *Human*. If the author defines *Woman* as equivalent to *Female*, the formula $\mathbf{R}_{subClass} \circ \mathbf{R}_{equiv} = \mathbf{R}_{subClass}$ deduces that *Woman* is a subclass of *Human*.

- In the right side of Figure 3.12, *Woman* and *Human* belong to the *specific model*. *Woman* is defined as a subclass of *Human*. *Person* belongs to the *generic model*. If the author defines *Human* as equivalent to *Person*. the formula $\mathbf{R}_{subClass} \circ \mathbf{R}_{equiv} = \mathbf{R}_{subClass}$ deduces that *Woman* is a subclass of *Person*.

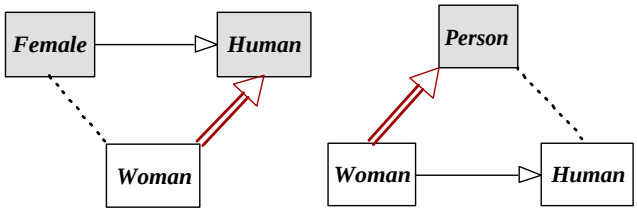

Figure 3.12 – Instantiated patterns ($R_{subclass} \circ R_{equiv} = R_{subclass}$)

The formula $\mathbf{R}_{subClass} \circ \mathbf{R}_{subClass} = \mathbf{R}_{subClass}$ describes two patterns (cf. Figure 3.13). The pattern in the left side (respectively the right side) of Figure 3.13 covers the case where there are two classes $C_{g,1}$, $C_{g,2}$ of the generic model (respectively $C_{s,1}$, $C_{s,2}$ of the specific model) linked by a $R_{subClass}$. When the author defines a $R_{subClass}$ between $C_{s,1}$ and $C_{g,1}$ (respectively between $C_{s,2}$ and $C_{g,1}$), the pattern in the left side deduces a $R_{subClass}$ between $C_{s,1}$ and $C_{g,2}$ (respectively the pattern in the right side deduces a $R_{subClass}$ between $C_{s,2}$ and $C_{g,1}$).

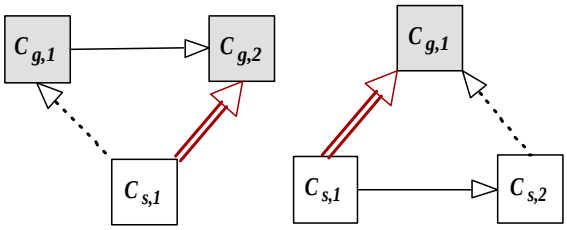

Figure 3.13 – Patterns corresponding to the formula $R_{subclass} \circ R_{subclass} = R_{subclass}$

For example, assume three classes *Student-Assistant*, *Student* and *Person*.

- In the left side of Figure 3.14, *Student-Assistant* belongs to the *specific model*. *Student* and *Person* belong to the *generic model*. *Student* is defined as a subclass of *Person*. If the author defines *Student-Assistant* as a subclass of *Student*, the formula $\mathbf{R}_{subClass} \circ \mathbf{R}_{subClass} = \mathbf{R}_{subClass}$ deduces that *Student-Assistant* is also a subclass of *Person*.
- In the right side of Figure 3.14, *Student-Assistant* and *Student* belong to the *specific model*. *Student-Assistant* is defined as a subclass of *Student*. The class *Person* belongs to the *generic model*. If the author defines *Student* is as subclass of *Person*, the formula $\mathbf{R}_{subClass} \circ \mathbf{R}_{subClass} = \mathbf{R}_{subClass}$ deduces that *Student-Assistant* is also a subclass of *Person*.

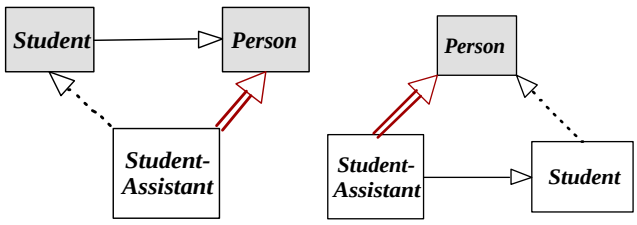

Figure 3.14 – Instantiated patterns ($R_{subclass} \circ R_{subclass} = R_{subclass}$)

The formula $\mathbf{R}_{equiv} \circ \mathbf{R}_{equiv} = \mathbf{R}_{equiv}$ describes two patterns (cf. Figure 3.15). The pattern in the left side (respectively the right side) of Figure 3.15 covers the case where there are two classes $C_{g,1}$, $C_{g,2}$ of the generic model (respectively $C_{s,1}$, $C_{s,2}$ of the specific model) linked by a R_{equiv}. When the author defines a R_{equiv} between $C_{s,1}$ and $C_{g,1}$, the pattern in the left side deduces a

R_{equiv} between $C_{s,1}$ and $C_{g,2}$ (respectively the pattern in the right side deduces a R_{equiv} between $C_{s,2}$ and $C_{g,1}$).

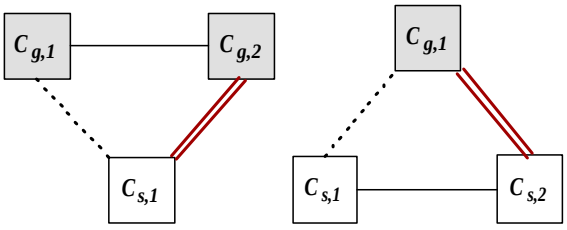

Figure 3.15 – Patterns corresponding to the formula R $_{equiv}$ ∘ R $_{equiv}$ = R $_{equiv}$

For example, assume three classes *Human*, *Individual* and *Person*.

- In the left side of Figure 3.16, *Person* belongs to the *specific model*. *Human* and *Individual* belong to the *generic model*. *Human* is defined as an equivalent to *Individual*. If the author defines *Person* as an equivalent to *Human*, the formula $\mathbf{R}_{equiv} \circ \mathbf{R}_{equiv} = \mathbf{R}_{equiv}$ deduces that *Person* is also equivalent to *Individual*.
- In the right side of Figure 3.16, *Human* and *Individual* belong to the *specific model*. *Human* is equivalent to *Individual*. The class *Person* belongs to the *generic model*. If the author defines only that *Human* is equivalent to *Person*, the formula $\mathbf{R}_{equiv} \circ \mathbf{R}_{equiv} = \mathbf{R}_{equiv}$ deduces that *Individual* is also equivalent to *Person*.

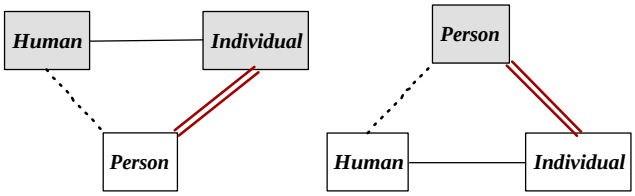

Figure 3.16 – Instantiated patterns (R $_{equiv}$ ∘ R $_{equiv}$ = R $_{equiv}$)

So far, we have described how an author can define mappings between classes of the generic and specific models, and how additional mappings between classes of the two models can be deduced. The third step deduces mappings between properties and between relations. These deductions are based on structural knowledge. We present them in the following.

4.2 Applying the second step on John's use case

No additional mappings can be deduced between classes of the two models manipulated in John's use case, as none of the specific (or generic) model includes two equivalent classes or a class being a specialization of another class.

5 Step 3/4: deduction of mappings between relations and between attributes

In this section our objective is twofold. Our first goal is to automatically deduce mappings between relations and between attributes of classes of the generic and specific models. Our second goal is to check the consistency of the new model created by the merging process.

To do so, our system uses structural knowledge, which we have modeled at a high level in a OWL meta-model in order to apply them to whatever the generic and specific models are. In the following, we present in section 5.1 the structural knowledge we have used in our system. Then, we detail in section 5.2 the modeling of this structural knowledge in a OWL meta-model. Finally, we describe how we reason on the defined OWL meta-model in order to, on one hand, deduce new mappings between relations and between attributes (cf. Section 5.3), and on the other hand, check the consistency of the new created model (cf. Section 5.4).

5.1 Structural knowledge

First of all, let us remember that we only consider OWL models (cf. Section 2.3). In OWL, a model includes a set of classes and a set of properties. A property is a binary relation. It is either:

- a relation between an individual and a datatype (representing an attribute);
- a relation between two individuals (representing a relation between two instances).

In this work, we refer to OWL properties by relations as relations (in its usual meaning) and attributes are both represented by relations in OWL. Furthermore, among the characteristics of a relation, we consider the characteristics functional and inverse functional. We also consider the cardinality constraints (OWL:maxCardinality, OWL:minCardinality and OWL:Cardinality). Other characteristics having no impact on multiplicities, like inverse of, transitive and symmetric, are processed before starting the process.

In order to deduce mappings between two relations of two different models[14], we consider information related to the domain, range, characteristics (functional, inverse functional) and cardinalities of the mapped relations. This information is used to establish the compatibility of the mapped relations. A mapping between two relations is possible only when the two relations are compatible[15]. Furthermore, we consider the information characterizing the compatibility of the mapped relations at different degrees. The order in which they are presented below is from the less constrained to the most constrained.

The less reliable mapping that can be deduced between two relations is called *potential link*. It exploits the (equivalence or specialization) mappings defined (or deduced) between the domain and range of the mapped relations, and it is described by the following definition.

Definition 2 *Two relations $R_{s,i,j}$ and $R_{g,k,l}$ are linked by a potential link if a mapping is defined between their domain and between their range.*

We exploit after that the information characterizing the compatibility of each two relations linked by a *potential link*. Therefore, more reliable mappings are deduced. We have considered three characteristics of the mapped relations: functionality property, inverse functionality property and cardinalities. The comparison of each of the three characteristics may allow to define a new mapping. We describe these mappings in the definitions below.

Definition 3 *Two relations $R_{s,i,j}$ and $R_{g,k,l}$ are linked by a compatible functional property link if those relations are linked by a potential link and if:*
$R_{s,i,j}$ and $R_{g,k,l}$ are both functional or not or $R_{s,i,j}$ is functional and $R_{g,k,l}$ is not.

[14]By misuse of language, we write in the following only *a mapping between two relations* instead of writing *a mapping between two relations where one relation belongs to the generic model and the other relation belongs to the specific model*. The two relations must be of the same type: either object or datatype properties.

[15]By compatible, we mean that, the considered information of the relation of the specific model is equivalent to (respectively more restrictive than) the same information of the relation of the generic model. We give later in this section a definition for each information characterizing the compatibility of the mapped relations.

Definition 4 *Two relations $R_{s,i,j}$ and $R_{g,k,l}$ are linked by a compatible inverse functional property link if those relations are linked by a potential link and if:*
$R_{s,i,j}$ and $R_{g,k,l}$ are both inverse functional or not or $R_{s,i,j}$ is inverse functional and $R_{g,k,l}$ is not.

Definition 5 *Two relations $R_{s,i,j}$ and $R_{g,k,l}$ are linked by a compatible cardinality property link if those relations are linked by a potential link and if:*

- *(Cardinalitymax($R_{s,i,j}$) $\leq$ Cardinalitymax($R_{g,k,l}$) and Cardinalitymin($R_{s,i,j}$) $\geq$ Cardinalitymin($R_{g,k,l}$)).*
- *Cardinalityvalue ($R_{s,i,j}$) $\leq$ Cardinalityvalue ($R_{g,k,l}$).*

The three mappings *compatible functional property link*, *compatible inverse functional property link* and *compatible cardinality property link* have the same reliability, even if each of them satisfies distinct conditions, one distinct condition at a time: compatible information either about functional property, either about inverse functional property, or about cardinality property. No condition is more reliable than the other.

A more reliable mapping can be deduced between two relations if they satisfy at the same time the three previous conditions. That means that, this mapping satisfies all the compatibility information that characterize two mapped relations. Consequently, it will be the most reliable link that can be deduced between two relations. We have called this mapping *probable mapping*, and we propose the following definition:

Definition 6 *Two relations $R_{s,i,j}$ and $R_{g,k,l}$ are linked by a probable link if they are linked by a potential link and if their cardinalities, functionality properties and inverse functionality properties are compatible.*

Probable links can be either equivalence or specialization links according to (1) the nature of mappings defined between the classes corresponding to the range and (2) the characteristics and cardinalities associated to the mapped relations. Note that, we do not test domains of relations to deduce the kind of probable link because domains are used only to define the belonging of relations and no other constraintes on relations.

Definition 7 *A probable link between $R_{s,i,j}$ and $R_{g,k,l}$ is an equivalence probable link if the two ranges are linked by an equivalence relation and if they have the same cardinalities.*

Definition 8 *A probable link between $R_{s,i,j}$ and $R_{g,k,l}$ is a specialization probable link if a mapping is defined between their range but the cardinalities on $R_{s,i,j}$ are stronger than those on $R_{g,k,l}$ or if they have the same cardinalities but the $R_{s,i,j}$ range is a subcategory of the $R_{g,k,l}$ range.*

Note that, only links satisfying the most constraints are proposed to authors. This only concerns *(equivalent or specialization) probable links*. However, they are based only on structural knowledge. As their name indicates it, they are only probable and not sure (semantically). Furthermore, for each relation of the generic model, several relations of the specific model may have compatible information. That means that, several mappings involving a relation of the generic model and different relations of the specific model can be deduced (inversely, for each relation of the specific model). Also, inconsistencies can be deduced between two relations. This is why, we need authors to choose the correct mappings between all the deduced mappings or eventually restart the process (further details in section 6).

So far, we have described the structural knowledge considered in our process. In order to propose a general process that can be applied on whatever generic and specific models can be, we have modeled the structural knowledge at a higher level than the processed models. More precisely, we have modeled the structural knowledge in a meta-model. This implies that the generic and specific models are going to be considered as instances of the proposed meta-model.

In the following, we detail further our modeling of structural knowledge in a meta-model. After that, we give mappings deduction rules, followed by inconsistencies deduction rules. Finally, we apply this step on John's use case.

5.2 Modeling structural knowledge using a meta-model

As the models to be merged are represented in OWL, we propose to represent structural knowledge in a meta-model based on the OWL meta-model. The OWL meta-model was defined by ODM (Ontology Definition Meta Model) of OMG as a MOF2 compliant meta-model[16]. It is composed of several UML class diagrams, a class diagram per element of an OWL model. Our system does not need all the diagrams of the OWL meta-model. Furthermore, it requires specific structural knowledge that are not yet modeled in the OWL meta-model. Therefore, we have first decided which parts of the diagrams of the OWL meta-model were needed to be reused (cf. Section 5.2.1). Then, we have enriched this sub-part in order to represent the needed structural knowledge (cf. Section 5.2.2).

5.2.1 Parts taken back from the OWL meta-model

As structural knowledge used by the process is relative to classes, properties and cardinalities according to the OWL terminology, we have reused the `Class`, `Property` and `Restriction` class diagrams included in the OWL meta-model, which are described in Appendice A.

The three class diagrams propose a complete modeling of classes, properties and cardinalities, but in our work we do not need all the modeled knowledge. This is why, we have reused only the parts of the three class diagrams that are reliable in our work. More precisely, we have reused the following parts of each class diagram (cf. Figure 3.17).

- In the `Class` diagram (modeled in a UML class diagram), the `Class` class and the `equivalentClass` and `subclass` relations are needed and have been reused.
- In the `Property` diagram (modeled in a UML class diagram), the `Property`, `FunctionalProperty` and `InverseFunctionalProperty` classes are needed and have been reused. Also, the relation `equivalentProperty` defined on the `Property` class has been reused.
- In the `Restriction` diagram (modeled in a UML class diagram), only four classes are needed and have been reused. These classes are: `Cardinality Restriction`, `Max Cardinality Restriction`, `Min Cardinality Restriction` and `Restriction`.

5.2.2 Modification and enrichment of the reused parts of the OWL meta-model

Before enriching the reused OWL meta-model in order to support structural knowledge, we brought some modifications on the reused parts of the OWL meta-model. The modifications concerns mainly two things.

1. Modeling of *XML-Schema datatypes*: we have noticed that the *XML-Schema datatypes* are considered as individuals of the class *Class*. This representation is not convenient for us because some characteristics of OWL classes that we have to represent are not relevant for datatypes. Consequently, we have decided to separate the modeling of OWL classes from the modeling of datatypes in two different classes, and we have proposed the creation of two different classes. Datatypes will be modeled in the class `Class`, and individuals different from datatypes will be represented in a new class called `Application Class`.

[16] http://www.omg.org/spec/ODM/1.0/

This new class has been defined as a specialization of the class `Class` because its instances share some characteristics with the datatype instances.

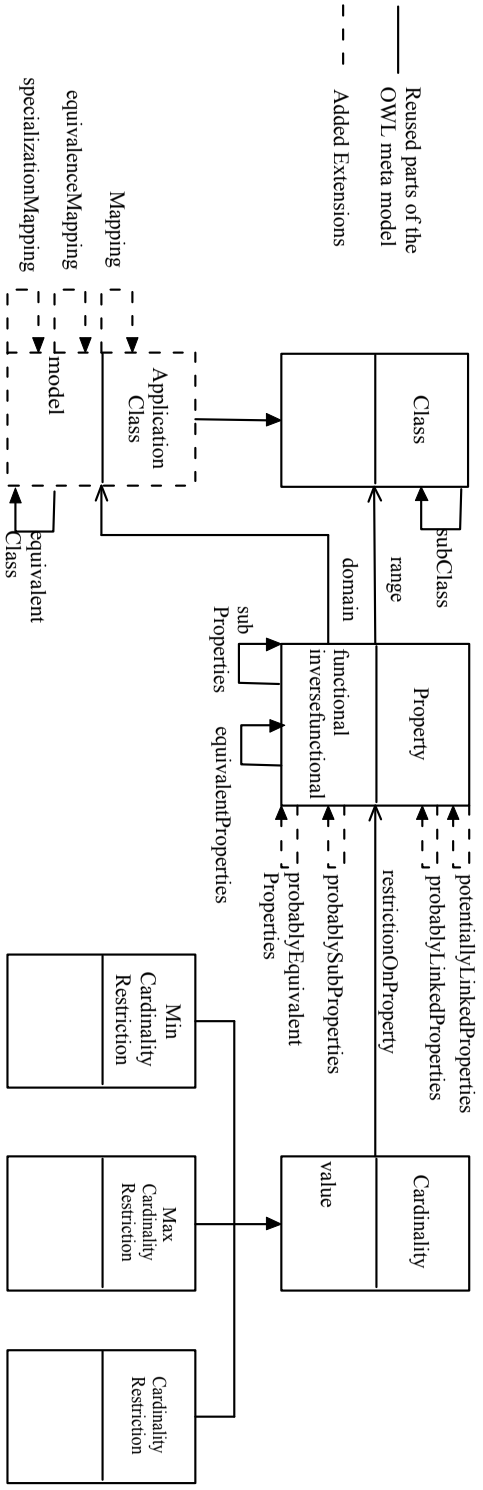

Figure 3.17 – The proposed meta-model

2. Modeling the kind of model (generir or specific) to which each instance belongs: we have added a `model` attribute only to the `Application Class` class.

After these modifications, we have enriched the extracted OWL meta-model with the structural knowledge described in the previous section. The resulting meta-model is presented in Figure 3.17.

So far, we have presented the structural knowledge used by our process in order to deduce mappings between two relations, and we have detailed their modeling on a modified version of the OWL meta-model. In the following section, we present the formalization of the deduction process of mappings between two relations in a declarative way using SWRL[17] rules.

5.3 Mapping deduction rules

In this section, we give the rules to deduce mappings between a relation of the generic model (noted P_g) and a relation of the specific model (noted P_s). The rules derive directly from the definitions given in Section 5.1 and are based on the proposed OWL meta-model described in Section 5.2. Mainly, the rules deduce four types of mappings: *potential mapping*, *compatible restriction mapping*, *probable mapping* and *inconsistency mapping*. We have grouped all the rules in Table 3.7. It includes the rule number, the condition part of the rule, and the conclusion part of the rule. Also, the table separates the rules allowing to deduce the same type of mappings. Therefore, the table is divided in four parts, each part is explained further in a section below.

Rule	Condition	Conclusion
R1	Property(?P_g) ∧ model(?P_g, "generic") ∧ domain(?P_g,?D_g) ∧ range(?P_g,?R_g) ∧ Property(?P_s) ∧ model(?P_s, "specific") ∧ domain(?P_s,?D_s) ∧ range(?P_s,?R_s) ∧ mapping(?D_g,?D_s) ∧ mapping(?R_g,?R_s)	potentiallyLinkedProperties(?P_g,?P_s)
R2	potentialLinkedProperties(?P_g,?P_s) ∧ functional(?P_g, f) ∧ functional(?P_s, f)	sameFunctionality(?P_g,?P_s) ∧ compatibleFunctionality(?P_g,?P_s)
R3	potentialLinkedProperties(?P_g,?P_s) ∧ functional(?P_g, false) ∧ functional(?P_s, true)	restrictiveFunctionality(?P_g,?P_s) ∧ compatibleFunctionality(?P_g,?P_s)
R4	potentialLinkedProperties(?P_g,?P_s) ∧ inversefunctional(?P_g, f) ∧ inversefunctional(?P_s, f)	sameInverseFunctionality(?P_g,?P_s) ∧ compatibleInverseFunctionality(?P_g,?P_s)
R5	potentialLinkedProperties(?P_g,?P_s) ∧ inversefunctional(?P_g, false) ∧ inversefunctional(?P_s, true)	restrictiveInverseFunctionality(?P_g,?P_s) ∧ compatibleInverseFunctionality(?P_g,?P_s)
R6	potentialLinkedProperties(?P_g,?P_s) ∧ MinCardinalityRestriction(?P_g, ?mg) ∧ value(?mg, m)∧ MinCardinalityRestriction(?P_s, ?ms) ∧ value(?ms, m) ∧ MaxCardinalityRestriction(?P_g, ?xg) ∧ value(?xg, x)∧ MaxCardinalityRestriction(?P_s, ?xs) ∧ value(?xs, x)	sameRestriction(?P_g,?P_s) ∧ compatibleRestriction(?P_g,?P_s)
R7	potentialLinkedProperties(?P_g,?P_s) ∧ CardinalityRestriction(?P_g, ?cg) ∧ value(?cg, m)∧ CardinalityRestriction(?P_s, ?cs) ∧ value(?cs, m)	sameRestriction(?P_g,?P_s) ∧ compatibleRestriction(?P_g,?P_s)
R8	potentialLinkedProperties(?P_g,?P_s) ∧ CardinalityRestriction(?P_g, ?cg) ∧ value(?cg, mg)∧ CardinalityRestriction(?P_s, ?cs) ∧ value(?cs, mc) ∧ swrlb:greaterThen(mg, ms)	restrictiveRestriction(?P_g,?P_s) ∧ compatibleRestriction(?P_g,?P_s)
R9	potentialLinkedProperties(?P_g,?P_s) ∧ MinCardinalityRestriction(?P_g, ?mg) ∧ value(?mg, vg)∧ MinCardinalityRestriction(?P_s, ?ms) ∧ value(?ms, vs) ∧ swrlb:greaterThen(vs, vg) ∧ MaxCardinalityRestriction(?P_g, ?xg) ∧ value(?xg, x)∧ MaxCardinalityRestriction(?P_s, ?xs) ∧ value(?xs, x)	restrictiveRestriction(?P_g,?P_s) ∧ compatibleRestriction(?P_g,?P_s)
R10	potentialLinkedProperties(?P_g,?P_s) ∧ MinCardinalityRestriction(?P_g, ?mg) ∧ value(?mg, m)∧ MinCardinalityRestriction(?P_s, ?ms) ∧ value(?ms, m) ∧ MaxCardinalityRestriction(?P_g, ?mg) ∧ value(?mg, vg)∧ MaxCardinalityRestriction(?P_s, ?ms) ∧ value(?ms, vs) ∧ swrlb:greaterThen(vg, vs)	restrictiveRestriction(?P_g,?P_s) ∧ compatibleRestriction(?P_g,?P_s)

[17] http://www.w3.org/Submission/SWRL/

R11	potentialLinkedProperties(?P_g,?P_s) ∧ MinCardinalityRestriction(?P_g, ?mg) ∧ value(?mg, m)∧ MinCardinalityRestriction(?P_s, ?ms) ∧ value(?ms, m) ∧ MaxCardinalityRestriction(?P_g, ?mg) ∧ value(?mg, vg)∧ MaxCardinalityRestriction(?P_s, ?ms) ∧ value(?ms, vs) ∧ swrlb:greaterThen(vg, vs)	restrictiveRestriction(?P_g,?P_s) ∧ compatibleRestriction(?P_g,?P_s)
R12	potentiallyLinkedProperties(?P_g,?P_s) ∧ compatibleFunctionality(?P_g,?P_s) ∧ compatibleInverseFunctionality(?P_g,?P_s) compatibleRestriction(?P_g,?P_s)	probablyLinkedProperties(?P_g,?P_s)
R13	potentiallyLinkedProperties(?P_g,?P_s) ∧ range(?P_g,?R_g) ∧ range(?P_s,?R_s) ∧ equivalentMapping(?R_g,?R_s) ∧ sameFunctionality(?P_g,?P_s) ∧ sameInverseFunctionality(?P_g,?P_s) sameRestriction(?P_g,?P_s)	probablyEquivalentProperties(?P_g,?P_s)
R14	probablyLinkedProperties(?P_s,?P_g) ∧ range(?P_g,?R_g) ∧ range(?P_s,?R_s) ∧ specializationMapping(?R_g,?R_s)	probablySubProperties(?P_g,?P_s)
R15	probablyLinkedProperties(?P_s,?P_g) ∧ restrictivefunctional(?P_g, ?P_s)	probablySubProperties(?P_g,?P_s)
R16	probablyLinkedProperties(?P_s,?P_g) ∧ restrictiveInversefunctional(?P_g, ?P_s)	probablySubProperties(?P_g,?P_s)
R17	probablyLinkedProperties(?P_s,?P_g) ∧ restrictiveCardinality(?P_g, ?P_s)	probablySubProperties(?P_g,?P_s)
R17	potentialLinkedProperties(?P_g,?P_s) ∧ functional(?P_g, true) ∧ functional(?P_s, false)	incompatibleFunctionality(?P_g,?P_s)
R19	potentialLinkedProperties(?P_g,?P_s) ∧ inversefunctional(?P_g, true) ∧ inversefunctional(?P_s, false)	incompatibleInverseFunctionality(?P_g,?P_s)
R20	potentialLinkedProperties(?P_g,?P_s) ∧ MinCardinalityRestriction(?P_g, ?mg) ∧ value(?mg, msg)∧ MinCardinalityRestriction(?P_s, ?ms) ∧ value(?ms, msr) ∧ swrlb:greaterThan(mgr, msr)	incompatibleRestriction(?P_g,?P_s)
R21	potentialLinkedProperties(?P_g,?P_s) ∧ MaxCardinalityRestriction(?P_g, ?mg) ∧ value(?mg, msg)∧ MaxCardinalityRestriction(?P_s, ?ms) ∧ value(?ms, msr) ∧ swrlb:greaterThan(msr, mgr)	incompatibleRestriction(?P_g,?P_s)
R22	potentialLinkedProperties(?P_g,?P_s) ∧ CardinalityRestriction(?P_g, ?cg) ∧ value(?cg, mg)∧ CardinalityRestriction(?P_s, ?cs) ∧ value(?cs, ms)	swrlb:notEqual(mg, ms) ∧ incompatibleRestriction(?P_g,?P_s)

Table 3.7 – SWRL Rules expressing structural knowledge

5.3.1 Deducing a potential mapping

The rule R_1 inferring a *potential mapping* derives directly from Definition 2. It allows to deduce a *potential mapping* between a relation of the generic model (noted P_g) and a relation of the specific model (noted P_s) whose domains are linked by a mapping and ranges are also linked by a mapping.

In rule R_1, *mapping(?C_g, ?C_s)* expresses a mapping between a class of the generic model and a class of the specific model. It is either defined by the author or inferred from additional mappings automatically deduced.

5.3.2 Deducing compatible restriction mappings

In order to deduce compatible restriction mappings, we compare the functional property, inverse functional property and cardinalities of two relations linked by a *potential mapping*. We have eight rules deducing compatible restriction mappings.

Two rules compare the functional property of two relations, which derive directly from Definition 3. Either the two relations have the same functional property (cf. R_2 in Table 3.7) or the functional property of the Ps is more restrictive than the one of Pg (cf. R3 in Table 3.7).

Similarly, there are two rules comparing the inverse functional property of the two relations (cf. R_4, R_5 in Table 3.7), which derive directly from Definition 4.

Besides, there are six rules comparing cardinality of the two relations, which derive from Definition 5. The condition *(Cardinalitymax($R_{s,i,j}$) ≤ Cardinalitymax($R_{g,k,l}$) and Cardinalitymin($R_{s,i,j}$) ≥ Cardinalitymin($R_{g,k,l}$))* in Definition 5 is expressed in four rules (cf. R_6, R_9, R_10, R_{11} in Table 3.7) as with SWRL we can only express AND operator. Similarly, the condition *(Cardinalityvalue ($R_{s,i,j}$) ≤ Cardinalityvalue)* in Definition 5 ($R_{g,k,l}$) is expressed in two rules (cf. R_7, R_8 in Table 3.7).

5.3.3 Deducing a probable mapping

The rule R_{12} inferring a *probable mapping* derives directly from Definition 6. It deduces a *probable mapping* between a relation of the generic model (noted P_g) and a relation of the specific model (noted P_s) that satisfies at the same time the four conditions below:

1. they are linked by a *potential mapping*;
2. they are linked by a *compatible functional mapping*;
3. they are linked by an *compatible inverse functional mapping*;
4. they are linked by a *compatible restriction mapping*.

We distinguish between two kinds of probable mappings.

1. *Equivalent probable mapping* that can be deduced using one rule R_{13} (cf. Table 3.7). The rule derives directly from Definition 7.
2. *Specialization probable mapping* that can be deduced using four distinct rules R_{14}, R_{15}, R_{16} and R_{17} (cf. Table 3.7). The rules derive from Definition 8. The definition expresses the following formula.

   Probable link ∧ *(Restrictive range* ∨ *restrictive functional* ∨ *restrictive inverse functional* ∨ *restrictive cardinality).*

   As the disjunction operator does not exist in SWRL, we have expressed this formula in four separate rules.

   - R_{14} to express: *probable link* ∧ *restrictive range*.
   - R_{15} to express: *probable link* ∧ *restrictive functional*.
   - R_{16} to express: *probable link* ∧ *restrictive inverse functional*.
   - R_{17} to express: *probable link* ∧ *restrictive cardinality*.

It may happen that relations of the specific model are more permissive than those of the generic model. Thereby, inconsistencies mappings are deduced. In the following we describe all the inconsistencies that can be deduced between two relations.

5.4 Inconsistency deduction rules

Inconsistencies relate to potential mappings and derive directly from cardinalities.

If a relation from the specific model is more permissive than the potential mapped relation of the generic model, then cardinalities of the two relations are incompatible. The cardinality may concern cardinalities, functional properties or inverse functional properties of the mapped relations. The process submits the deduced inconsistencies to the author. The author is advised to restart the process again by specifying other mappings between classes than those specified during the current process or to modify his model (cf. Section 6).

Let us to consider rule R_{18}, which compares two relations linked by a potential mapping, and where the relation of the specific model (noted Ps) is more permissive than the relation of the generic model (noted Pg). That means that, it may be instances in the specific model that do not respect the cardinality imposed by the relation of the generic model. This is why, when deducing such inconsistency mapping, we invite the author to modify his model or to restart the merging process by selecting other mappings between classes.

We have defined at all 5 rules to deduce inconsistencies, one rule for functional property R_{18}, another for inverse functional property R_{19} (cf. Table 3.7) and three for cardinality, as for cardinality the two relations may have a minimum, a maximum or a defined value. We have defined one rule for each case R_{20}, R_{21} and R_{22} (cf. Table 3.7).

Once all structural knowledge have been checked between all two relations, only equivalent or specialization probable links and inconsistency mappings are proposed to the author. Concerning the probable links, as their name indicates it, they are only probable and not sure (semantically). They have to be validated. On the other hand, inconsistencies mappings indicate possible problems for defining a consistent merged model and the author is notified about that.

5.5 Applying the third step on John's use case

First of all, the generic and specific models are transformed as instances of the defined meta-model. Each class of the specific model becomes an instance of the `Application Class` class, with the `model` attribute equal to `specific`. Similarly, each class of the generic model becomes an instance of the `Application Class` class, with the `model` attribute equal to `generic`. Furthermore, the equivalence (respectively specialization) mappings between classes of the generic and specific models (defined or deduced) are modeled using the `equivalenceMapping` (respectively `specializationMapping`) relation of the `Application Class` class.

After that, the deduction process can be performed on the defined meta-model and its instances. Several mappings are deduced. We describe some of them in Table 3.8. Only mappings in bold are proposed to the author.

Generic model	Specific model	Tested condition	Mapping
pre-requisite	successor	domain, range	potential mapping
		functional prop	compatible functional property
		compatibe functional property compatible inverse functional property compatible restriction	**probable equivalent mapping**
pre-requisite	part-of	domain, range	potential mapping
		functional prop	compatible functional property
		compatibe functional property compatible inverse functional property compatible restriction	**probable equivalent mapping**
abstraction domain:CONCEPT range:RESOURCE	abstraction domain:NOTION, range:DESCRIPTION	domain, range	potential mapping

		functional prop	restrictive functional property
		restrictive functional property restrictive inverse functional property compatible restriction	**sub equivalent probable mapping**

Table 3.8 – A Part of deduced mappings between relations of the generic and specific models

6 Step 4/4: validation of mappings and presenting inconsistencies

This step is interactive. It considers two distinct deductions: deduction of probable mappings and deduction of inconsistencies.

In the following we note R_g a relation of the generic model and R_s a relation of the specific model. We also use C_g to denote a class of the generic model, and C_s to denote a class of the specific model.

6.1 Validating deduced mappings between relations

All probable mappings are presented to the author. The author has to validate each deduction, choose the appropriate deduction when a relation is involved in different probable mappings or reject a given deduction. In order to suggest the correct actions to be chosen, the process analyzes relation per relation of the generic model. It distinguishes the following cases:

R_g is linked by only a probable mapping to R_s. In this case, the process checks whether R_g is linked by inconsistency mappings to relations of the specific model having the same domain and range as R_s

1. When there are also inconsistency mappings (cf. Figure 3.18), the author is informed and is invited to validate or to reject the deduction of probable mappings and of inconsistency mappings.

2. When there no inconsistency mappings (cf. Figure 3.19), the author is invited to validate or to reject the deduction of probable mappings.

R_g is linked by several probable mappings to relations of the specific model. These relations of specific model must have the same domain and range. In this case, the process checks whether R_g is linked by inconsistency mappings to relations of the specific model having the same domain and range as the considered relations of the specific model.

1. When there are also inconsistency mappings (cf. Figure 3.20), the author is informed and is invited to choose between these probable mappings (zero, one or more) and of inconsistency mappings.

2. When there are no inconsistency mappings (cf. Figure 3.21), the author is invited to choose between these probable mappings (zero, one or more).

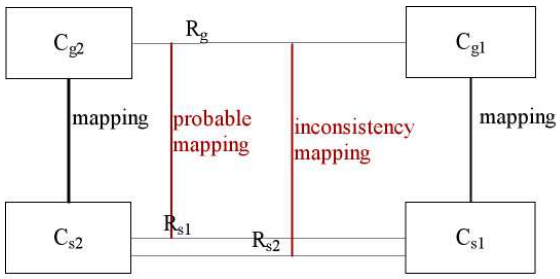

Figure 3.18 – R_g is linked by only a probable mapping to R_{s1} and by inconsistency mappings to relations of the specific model having the same domain and range as R_{s1}

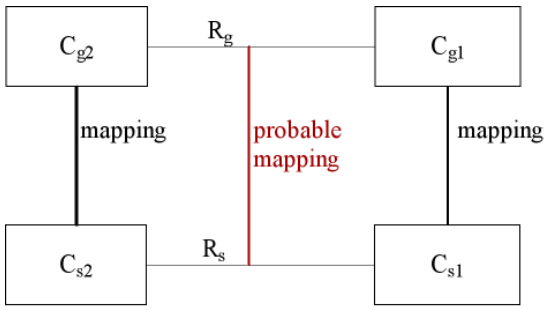

Figure 3.19 – R_g is linked by only a probable mapping to R_{s1} and by no inconsistency mappings to relations of the specific model having the same domain and range as R_{s1}

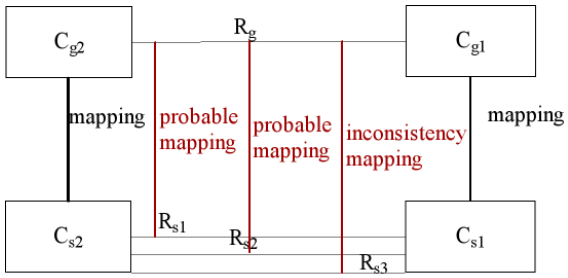

Figure 3.20 – R_g is linked by a probable mapping to R_{s1} and to R_{s2} and by inconsistency mappings to relations of the specific model having the same domain and range as R_{s1}

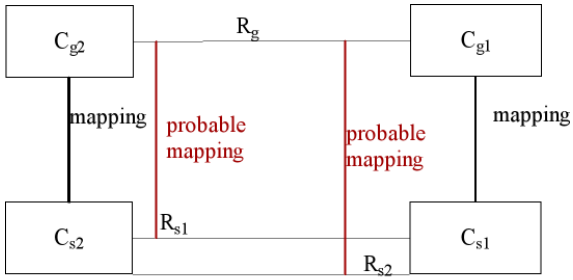

Figure 3.21 – R_g is linked by a probable mapping to R_{s1} and to R_{s2} and by no inconsistency mappings to relations of the specific model having the same domain and range as R_{s1}

Note that, when the author validates or chooses a probable mapping, this later become a sure mapping. Similarly, when the author rejects an inconsistency mapping, this later is no more considered as a problem preventing from building the merged model. If others inconsistency mappings subsist, they will be presented to the authors (cf. Section 6.2). Otherwise, the merged model will be built (cf. Section. 6.3).

6.2 Presenting inconsistency mappings

Inconsistency mappings notify problems, particularly because they prevent from building a coherent merged model. Consequently, the process advices the author to review his mappings between classes, either:

1. the author may found mistakes in his definition of classes. The process advices him to start the process again by specifying new mappings between classes;
2. the author is sure about his mappings between classes. The process advices him to check his model and suggests him to carefully modify his model. In this case, the author has to ensure that his modifications will not affect the instances of his model. Remember that, the author targets to reuse existing reasoning on instances of his model, therefore the instances are important and must not be lost.

6.3 Building the merged model

We propose an incremental reasoning to build the merged model. We consider class per class of the generic model. A class of the generic model can have no mapping with any class of the specific model, have one or more equivalence mappings with classes of the specific model, or have one or more specialization mappings with classes of the specific model. Table 3.9 presents reasoning on each of these cases.

Tested case C_g *has*	Verifications on C_g, C_s linked by the tested mapping	Impact on the merged model
No mapping		C_g is added with its relations to the merged model.
One or more equivalence mappings.	C_g, C_s must have the same number of relations, and each relation of C_g has an equivalence mapping with a relation of C_s.	If these verifications are satisfied, both classes with their relations[18], and checked equivalence mappings are added to the merged model, otherwise the process stops[19].
One or more specialization mappings.	C_g has a number of relations less than or equal to the number of relations of C_s, and each relation of C_g has a probable mapping with a relation of C_g.	If these verifications are satisfied, both classes with their relations and checked mappings are added to the merged model, otherwise the process stops.

Table 3.9 – Reasoning to build the merged model.

[18]Classes and relations are added only if they don't belong yet to the merged model. They are not duplicated in the merged model

[19]The process stops and no merged model is proposed to the author

6.4 Applying the fourth and last step on John's use case

When applying the process of validating deduced mappings between relations, we note that the relation *pre-requisite* of the generic model has several probable mappings: a probable mapping with the relation *successor* and another one with the relation *part-of* of the specific model. Both relations *successor* and *part-of* have the same domain and range, but there is no inconsistency mapping with a relation having the same domain and range as the relation *successor*. Therefore, according to the proposed process (cf. Figure 3.21), this later will propose to John to choose between these two deductions.

A detailled analysis about which mappings are proposed to John and which action John has to perform is done in the fourth situation in Chapter 8, Section 1.

7 Summary

In this part of the thesis, we have proposed a process enabling an author to integrate his models and instances with generic models. This process assumes the models to be merged are relatively small and that the author has a very good understanding of his models. Consequently, he is responsible for semantic validation.

The process is generic and can be performed on any generic and specific models expressed in OWL. It relies on the author to start the design process. He has to specify a minimum set of mappings between classes from which the system automatically deduces all the other possible mappings and eventual inconsistencies. Above all, this process enables to check and to validate the model resulting from the merging process.

To sum-up the process is characterized of:

- *Conceptualization of 8 patterns*. The patterns allow to deduce additional mappings between classes of the author's and generic models. They are based on recurrent structures that can be present when merging the author's and generic models.
- *Definition of a generic solution, based on a modified version of the OWL meta-model*. Therefore, the process can be applied on any two OWL models.
- *Expression of constraints in a declarative way by 22 rules*. Indeed, we have formalized the structural deductions needed by our process in 22 rules.

Basing on these characteristics, the process builds a merged model that includes at the same time the generic and the specific model given as inputs and includes also mappings between the two models. The mappings are very precise. As for structural deductions it is done automatically, and for semantic validation we refer to the author's knowledge. Therefore, the precision of mappings is 100%.

We have implemented this process as a Protégé plug-in. The implementation and the installation of the plug-in are presented in Chapter 7, and an experiment with the plug-in is given in Chapter 8.

We argue that this process would help authors to reuse existing reasonings (reuse adaptation, in the AH field). Note that, particularly in the AH field, when inside the merged model, some elements of the generic model are not specialized by at least one element of the specific model, this implies that, the part of adaptation exploiting these non specialized elements cannot be reused. On the other hand, when inside the merged model, some elements of the specific model have not a mapping with at least one element of the generic model, this implies that, the instances of the non linked elements of the specific model will not be exploited in the adaptation, unless the author extends manually the adaptation. For this purpose, in our plug-in, in addition of

proposing this process, we propose an overview of the non specialized elements of the generic model and of the non linked elements of the specific model. Please refer to Chapter 7 for further details.

Part IV

Assisting in the expression of adaptive navigation

In this part of the thesis, we focus on the authoring process of the adaptation model, which is most often the less intuitive part to be authored in AH by non technical persons, like teachers, for example, who do not have the required logic or programming skills.

Indeed, authors have to specify an adaptation model, in which they describe resources to propose to users having distinct characteristics and different knowledge, in order to reach their specific goals. This is done through the definition of multiple adaptation strategies. By an adaptation strategy, we mean *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 AH face numerous challenges when defining their adaptation strategies.

- The first challenge concerns the expression of adaptation strategies. It is often done using condition-action rules or event-condition-action rules, which is complex and time-consuming. Recent solutions propose graphical tools or languages using constructors to support the expression adaptation.
- The second challenge concerns the reuse of adaptation strategies from one system to 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*" [70]. This paradigm endorses expressing adaptation at a high level, independently of all AHSs and then translating this adaptation into a particular AHS.
- The third challenge concerns the granularity in writing adaptation strategies. Its target is to avoid writing the common parts of adaptation strategies several times.

As shown in Chapter 2, Section 2.2, till now, there have been no works concerning building complex adaptation strategies, independent of any system by combining simple ones.

In this part of the thesis, we present a new framework addressing these three challenges. The framework concentrates on the ease of defining adaptation strategies, at a fine granularity, on combining them and on the facility of reusing existing adaptation strategies. Furthermore, it enables authors to express their adaptation strategies at a high level, independently of any adaptation engine.

We also propose a study of the expressivity of knowledge represented by our framework versus GLAM and LAG. We have studied the expressivity of their modeling of adaptation and also the expressivity of the elements on which the adaptation is expressed. From these studies, we have come up with an integrated vision of the basic actions that can be used in defining adaptation, and also with an unifying vision of modeling the domain model. These studies have served as a basis for our definition of translators to GLAM and LAG.

This part is organized as follows. Chapter 4 presents our framework enabling authors to define their own adaptation strategies. Afterward, Chapter 5 proposes a study of the expressivity in our framework versus the expressivity in GLAM and in LAG. Finally, Chapter 6 describes how our framework can be plugged to GLAM and LAG.

CHAPTER 4

Expressing adaptive navigation using adaptation patterns

In this chapter, we present a methodology to express adaptation strategies by concentrating on the ease of defining adaptation strategies, independently of any adaptation engine, at a high level and in an easy manner.

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 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.

We implemented this process as a Protégé plug-in (cf. Chapter 7, Section 2). This has been followed by the implementation of translators and by several experimentations which can be found later in this thesis (cf. Chapter 8, Section 2.1).

This work has been published in the IEEE journal paper Transaction Learning Technologies [77], the international conference on Intelligent Systems Design and Applications *ISDA'2010* [75], and the Journées Francophones d'Ingénierie des Connaissances *IC'2010* [76].

The chapter is organized as follows. It presents in Section 1 related work on the expression of adaptation, and demonstrates the intuition of our work in Section 2, with a use case. Section 3 reviews the main aspects of our proposal. Section 4 presents the description of elementary adaptation patterns and their organization in a typology, and Section 5 describes how elementary adaptation patterns can be used to define adaptation strategies, and illustrates each step of the process on the use case.

1 Related work in expressing adaptive navigation in Adaptive Systems

Most often, during the authoring process of adaptation on existing domain and user models, authors ask themselves two questions [7]:

Q1 *what kind of adaptation can they provide for users?*

Q2 *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 1.1), while for producing adaptation, authors use the adaptation language developed in their own university or the one adviced by people in their community. We present in Section 1.2 some of existing adaptation languages.

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 1.3).

1.1 What kind of adaptation could be provided?

The well-known Brusilovsky taxonomy [6] is undoubtedly the most used typology of adaptation. It describes several methods of adaptation that can be combined together. The methods are organized into two non disjoint groups: adaptive presentation and adaptive navigation support, as presented in Chapter 2, Section 1.1. This typology assumes that the available resources can be modified and restructured during the adaptation process. Hence, not all methods of this typology are suitable when there is no control over the distributed resources.

As, in this thesis we focus on the expression of adaptive navigation, we get a particular interest on methods included in the adaptive navigation support group. This group includes six methods:

- *Guidance*: means that users are supervised step by step. It is done by proposing one link at a time to users. It may be either local or global guidance. Global guidance means that the system calculates complete navigation paths, while local guidance means that the system calculates only the next step each time (e.g., it is proposed by Interbook [8]).
- *Link sorting/ordering*: defines the priority of all the links of a particular page.
- *Link hiding*: hides, removes or disables links to users (e.g., AHA! [21] hides links that are not relevant to users).
- *Link annotation*: suggests links to users. The suggestions are often expressed using visual cues (e.g., WHURLE [49] uses colors for suggestions).
- *Link generation*: generates new links on a page.
- *Map generation*: proposes a reorganization of links to be presented to users. It is implemented as a combination of other techniques. For example, AHA! [21] proposes this reorganization of links in a separate frame.

Besides, Stash and al. [67] have proposed a classification of external actions that can be implemented in AHS. This classification is summarized in Table 4.1 (as described in Stash

thesis [66]). It includes *actions on individual items*[1] (e.g, selection, showing items or links to items), *actions on a set of items* (e.g, ordering items), *hierarchical actions* (e.g, define a depth navigational path using actions on children compared to the current item proposed to users) and *actions on the overall environment* (e.g, changing the layout).

Basic actions on items	Selection Showing the content of an item Showing a link to an item
Hierarchical actions on items	Actions on child items of the current item Actions on parent item of the current item
Actions on groups of items (e.g., sibling)	Ordering Performing "action on items" on each group of items
Action on the overall environment	Changing the layout of the presentation

Table 4.1 – Refined classification of actions in adaptive strategies according to [66]

As cited by Stash, the selection of items has no equivalent among the methods of the adaptive navigation support group defined by Brusilovsky. However, it can be implemented as a part of the adaptive multimedia presentation or altering fragments defined by Brusilovsky. The ordering information can be implemented through direct guidance, link sorting, link annotation or link hiding. Providing learners with navigational support can be implemented through the map adaptation.

1.2 How can authors express their adaptation?

Multiple solutions are offered to authors to express adaptation. We have grouped the existing solutions in three main categories.

1.2.1 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, like in AHA [21], WHURLE [49] or GLAM [39] (an example of a GLAM adaptation strategy is given in the next section). 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 *Graph Author Tool* for AHA! [21] (cf. Chapter 2 in Section 2.2.1) uses visualization in order to help authors. For each instance of concept, the tool applies templates in order to generate adaptation rules. Regardless, authors are captive to a particular system. Adaptation strategies expressed in a system cannot be reused outside this system. They have to be rewritten.

1.2.2 Generic adaptation languages accompanied by translators to existing adaptation engines

Some generic languages (independent of any system) have been proposed to specify adaptation, as presented in Chapter 2, Section 2.2.2. Among them, the LAG language [12], which is an implementation of the specification of the adaptation language defined in the LAOS model [13]. It includes conversion to the WHURLE [49], Blackboard [54] and AHA! [21] adaptation engines. However, LAG is like a programming language, which is not very suitable for non technical authors (an example is given in the next section).

[1] An item refers to a resource in the AH field.

Recently a new Generic Adaptation Language (GAL) has been developed to describe adaptive hypermedia [65]. 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 easily be reused.

1.2.3 Hypertext and adaptation patterns

Some design patterns for expressing personalization in web applications have been proposed [20], based on commonly used design structures. Before going further in these contributions, let us first introduce the notion of design patterns.

Design patterns

We cannot speak about design patterns without citing the definition of the pioneer of design patterns, the architect Alexander. He describes a design pattern as "*.. a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such way that you can use this solution a million times over, without ever doing it the same way twice*" [31]. They have been firstly used in architecture, afterward they have been used in other fields and particularly in software engineering.

In short, a design pattern describes a recurrent solution to a common problem in software design. The proposed solution is generic and cannot be directly translated to code. A design pattern is described using the characteristics presented in Figure 4.1.

Pattern Name and Classification: the pattern's name that helps in identifying the pattern.
Intent: a short description explaining the goal of the pattern and the reason for using it.
Motivation: a scenario illustrating the design problem and a context in which this pattern can be used.
Applicability: situations in which this pattern can be usable. It must highlight how to recognize these situations.
Solution: a graphical representation of the pattern using a notation based on the Object Modeling Technique (OMT) [58].
Constituents: the set of the classes and objects used in the pattern and their responsibilities.
Collaboration: a description of the interactions between the participants of this pattern.
Consequences: a description of the results, side effects, and trade offs caused by using this pattern.
Implementation: a description of an implementation of this pattern.
Sample Code: an illustration of how this pattern can be used in a programming language.
Known Uses: examples of real usages of this pattern.
Related Patterns: a discussion compared to other patterns, its similarities or main differences, the pattern with which it can be combined.

Figure 4.1 – Description of the structure of a design pattern [31]

Using this structure, several design patterns have been proposed. They have been organized in different groups according to their purpose (creational, structural or behavioral) and their

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

scope (applied on classes or objects) [31].

In practice, design patterns have proved several advantages such as speeding up the development process by providing tested and proven development paradigms, etc.

Adaptation patterns based on design patterns

Garzotto et al. [32] have proposed patterns to express adaptation strategies according to learning styles. More precisely, they have used Felder/Silverman learning style model to define their patterns. For each learning style attribute of this model, a pattern has been proposed. It is defined using three of the characteristics of the design patterns: *Name*, *Intent* and *Solution*. These three characteristics are explained further in Figure 4.2 and are illustrated by an example in Figure 4.3 .

Name: specifies the name of the pattern.

Intent: describes the problem for which this pattern can be used.

Solution: uses four dimensions to describe the solution

- *the content:* specifies resources that would be proposed to users.
- *the possibilities of navigation and interactions:* describes how users are going to navigate through the proposed resources.
- *the activities:* describes the actions that can be performed on resources by users (like: answering exercises, tagging)
- *the lay-out:* describes the presentation tasks (like colors, fonts).

Figure 4.2 – Structure of a design pattern as described by Garzotto et al. [32]

Name: Global learner.

Intent: addresses the needs of a global learner.

Solution: uses four dimensions to describe the solution

- *the content:*
  - provides preview of the topic,
  - highlights advanced concepts,
  - provides information on relationships to the context of a topic - theoretical/conceptual,
  - provides information on relationships to relevant topics,
  - includes exercises at any level of detail about a topic and those involving creativity and involving generating alternative solutions that differ from the classical ones.
- *the possibilities of navigation and interactions:*
  - provide the learner with a wide set of navigation facilities,
  - support top-down learning,
  - allow learners to look for advanced concepts and to exercise even when all prerequisite elements are not yet fully explored.
- *the activities:*
  - allow learners to input alternative solutions beside offering the selection among a set of standard solutions
  - allow learners to input comments and criticism
- *the lay-out:* In the different pages, it highlights challenging exercises and topics.

Figure 4.3 – Example of a design pattern as defined by Garzotto et al. [32]

However, 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 either combined together or modified, i.e, authors have to find a pattern corresponding to their desired adaptation strategy, otherwise, they can not express it.

Cristea et al. [5] have identified design tasks and problems that a designer of AEH must consider. These tasks and problems have been organized in five groups, as follows:

1. A group related to a learner model. It specifies the relevant attributes that have to be captured in the learner model.

2. A group related to detection mechanisms of learner characteristics. It specifies the structure of the learner model and the mechanisms that would be used to update it.

3. A group related to instructional strategy definition. It addressed an adaptation strategy for each distinct learner characteristic.

4. A group related to instructional view definition. It specifies all information needed by the adaptation. This includes the appropriate resources to be proposed, the structure of navigation or presentation.

5. A group related to adaptation mechanism definition. It specifies how the system acts after learner updates. For example: which layout will be proposed or which user characteristic is going to be updated.

Each group has been associated to one pattern as it defines a single problem. The proposed patterns consider only three characteristics of the design patterns: *Name*, *Intent* and *Solution*. However, similarly to the work done by Garzotto et al. [32], there is no 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.

1.3 Expressing adaptive navigation in 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 education [7]. Most of the existing systems are built upon an existing AHS (e.g., [12] on top of [21]). Multiple issues are to be faced in order to develop open corpus based AHS [5, 9], including automatic hypertext creation, indexing of open corpus resources and content preparation. None of these systems faces the problem of expressing adaptation, by AH authors, in a simple way.

1.4 Summary

We have discussed here solutions helping authors to find which adaptation they can propose and how they can express it. However, till now, there have been no works concerning building complex adaptation strategies, independent of any system by combining simple adaptations.

In this thesis, we focus on this specific point. Adaptation strategies must be defined at a fine granularity. Our aim is thus to help authors define their own adaptations, independently of any adaptation engine, at a high 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 this chapter.

2 Motivation through Jane's use case

Assume that Jane, who is a course author wants, to build an adaptive course from her material, i.e., Jane is going to author an AEH. 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.

2.1 Description of Jane's domain and user models

Jane proposes a domain model[3] (cf. Figure 4.4), in which she considers the addressed notions as instances of the class *Concept*[4]. The concepts must be learned in a particular order, that is defined through the relation *prerequisite*. 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 opts thus for textual and audio definitions, textual examples and audio exercises addressing the database concept, textual definitions and textual exercises addressing the relational_model concept. For example, a textual definition can address the relational_operators concept. In terms of prerequisite relations, the database concept can be a prerequisite of the relational_model concept, which in turn can be a prerequisite of the relational_operators concept.

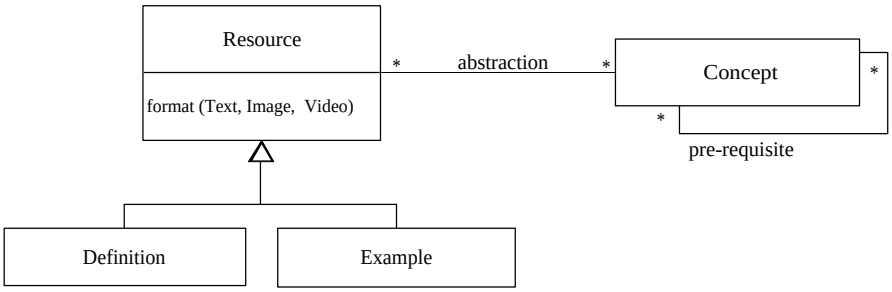

Figure 4.4 – Jane's domain model in UML

Jane considers the following student characteristics (cf. Figure 4.5):

- 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;

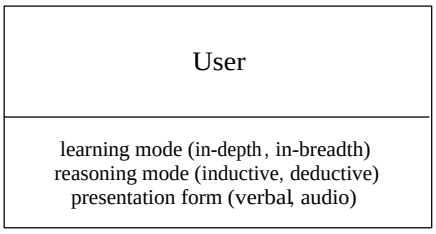

Figure 4.5 – Jane's user model in UML

- reasoning mode: an *inductive* reasoning mode means that the student has access to

[3] UML has been chosen here only for visual reasons.

[4] Here, names of classes have the first letter in upper-case and are in italic, and names of instances have the same name as the class for which they belong in lower-case.

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, an *audio* presentation form is for those preferring audio ones.

2.2 Description of Jane's adaptation

Among the adaptation strategies that Jane wants to propose, we only 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 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 AHS. However, they are not easy to implement and require good backgrounds. See as an illustration, the implementation of $S1$ using GLAM in Figure 4.6 (for GLAM syntax see Section 1.1.1), and using LAG in Figure 4.7 (for LAG syntax see Section 2.2.2). This implies that Jane already has her domain and user models in the format understood by her AHS.

Naturally, Jane expressed $S1$ in three parts:

S1-1 concerns students whose learning mode is *in-depth*;
S1-2 concerns students with an inductive *reasoning mode*;
S1-3 concerns students preferring *audio* resources.

These parts can be considered independently of one another and may compose 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 *textual* format if available, otherwise in an *audio* format.

To enable Jane to easily define her strategies, i.e. in the most natural way possible, we offer the possibility to specify each part of an adaptation 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 *prerequisite* 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 only has 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.

3 Main aspects of the EAP framework

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

S1 is defined in three steps (for GLAM syntax see Chapter 2, Section 1.1.1):
Step1: defining GLAM rules

R1 *type(r, Example) ∧ format(r, audio) ∧ abstraction(r, concept1) ∧ abstraction(currentR, concept2) ∧ prerequisite(concept2, concept1) ∧ prerequisite*(concept1, goal) → Read(r, degree)*

R2 *type(r, Example) ∧ format(r, text) ∧ abstraction(r, concept1) ∧ abstraction(currentR, concept2) ∧ pr-requisite(concept2, concept1) ∧ prerequisite*(concept1, goal) → Read(r, degree)*

R3 *type(r, Definition) ∧ format(r, audio) ∧ abstraction(r, concept1) ∧ abstraction(currentR, concept2) ∧ prerequisite(concept2, concept1) ∧ prerequisite*(concept1, goal) → Read(r, degree)*

R4 *type(r, Definition) ∧ format(r, text) ∧ abstraction(r, concept1) ∧ abstraction(currentR, concept2) ∧ prerequisite(concept2, concept1) ∧ prerequisite*(concept1, goal) → Read(r, degree)*

R5 *type(r, Example) ∧ format(r, audio) ∧ abstraction(r, concept1) ∧ prerequisite*(concept1, goal) → Read(r, degree)*

R6 *type(r, Example) ∧ format(r, text) ∧ abstraction(r, concept1) ∧ prerequisite*(concept1, goal) → Read(r, degree)*

R7 *type(r, Definition) ∧ format(r, audio) ∧ abstraction(r, concept1) ∧ prerequisite*(concept1, goal) → Read(r, degree)*

R8 *type (r, Definition) ∧ format (r, text) ∧ abstraction(r, concept1) ∧ prerequisite (concept1, goal) → Read(r, degree)*

Note that: each of currentR, concept1 and concept2, presents a variable indicating an instance of *Resource* class.
R1 proposes audio examples according to a depth-first navigational path based on concepts using the relation *prerequisite* between concepts, 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 (for each user characteristic, we associate a set of rules, otherwise the user characteristic is not considered in adaptation) :

- The value *audio* of the user characteristic *presentation form* is associated to R1, R3, R5, R7
- The value *textual* of the user characteristic *presentation form* is associated to R2, R4, R6, R8
- The value *inductive* of the user characteristic *reasoning mode* is associated to R1, R2, R3, R4, R5, R6, R7, R8
- The value *depth-first* of the user characteristic *learning mode* is associated to R1, R2, R3, R4, R5, R6, R7, R8

Step3: defining GLAM meta-rules

MR1 The value *audio* of the user characteristic *presentation form* > The value *textual* of the user characteristic *presentation form*

MR2 R1 ⊃ R5	***MR3*** R1 ⊃ R6
MR4 R1 ⊃ R7	***MR5*** R1 ⊃ R8
MR6 R2 ⊃ R5	***MR7*** R2 ⊃ R6
MR8 R2 ⊃ R7	***MR9*** R2 ⊃ R8
MR10 R3 ⊃ R5	***MR11*** R3 ⊃ R6
MR12 R3 ⊃ R7	***MR13*** R3 ⊃ R8
MR14 R4 ⊃ R5	***MR15*** R4 ⊃ R6
MR16 R4 ⊃ R7	***MR17*** R4 ⊃ R8
MR18 R1 ⊃ R3	***MR19*** R2 ⊃ R4
MR20 R5 ⊃ R7	***MR21*** R6 ⊃ R8

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

[a]The goal is a particular concept to be reached by students

Figure 4.6 – 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)
)
```

Figure 4.7 – Jane's $S1$ in the LAG format

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 [31] (for an introduction on design patterns, see Section 1.2.3). We argue that designing an adaptation strategy is a kind of conception: authors design similar adaptation strategies 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.

In the following, first, we describe the general structure of author's domain and user models. Afterward, we explain the main steps of the EAP framework. Finally, we apply the EAP framework on Jane's use case.

3.1 Structure of author's domain and user models used by the EAP framework

The EAP framework accepts any domain and user models that are composed of a set of classes, properties and relations between classes.

As for the domain model, the EAP framework makes a clear separation between concepts and resources. Therefore, the author's domain model must include either concepts alone, or resources alone or both concepts and resources with a clear separation between both. When a domain model includes both resources and concepts, a particular relation is defined in order to link the class modeling concepts and the class(es) modeling resources. We refer to this relation by *abstraction* and no constraints are defined on it.

3.2 Steps to define a new adaptation strategy

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, which we introduce in Section 4. The instantiation process and the combination process are described in Section 5.

4 Elementary adaptation patterns

The notion of elementary adaptation patterns that we propose, is an abstraction of Jane's parts *S1-1*, *S1-2*, *S1-3*. Furthermore, we define 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 4.1). The syntax and semantic of elementary adaptation patterns are described in Section 4.2, and their typology is defined in Section 4.3.

4.1 Fundamental criteria for defining elementary adaptation patterns

Like each part of $S1$ 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 4.1.1) and to organize the selected resources (cf. Section 4.1.2).

4.1.1 Criteria used to select resources

Criteria used to select resources are based on the domain model, where the available 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 (called *Resource*) to be proposed to users, and the class representing all the domain concepts (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 or breadth-first.

4.1.2 Criteria used to order the selected resources

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

We have looked over the Brusilovsky typology (cf. Section 1.1) methods of the adaptive navigation support group excluding those modifying resources (e.g, hiding links belonging to content of resources). Only *direct guidance*, *adaptive ordering* and *adaptive link annotation* have been considered. We have grouped the *direct guidance* and *adaptive ordering* in one operation as both of them define a kind of order either on a link or on several links per time. Therefore, two operations come from the Brusilovsky typology.

We have also looked over the classification of external actions in AHS defined by Stash and al. [66]. Similarly as before, we only consider the actions having impact on the navigation, which include: *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 items, parents or children. So, we only consider the *selection*, *show* and *order* actions. Note that the *show* action cannot be used alone, it is necessarily combined to other actions. The combination of *order* and *show* actions is equivalent to the order operation defined by Brusilosvky. Therefore, we can neglect it, and retain only the combination of *select* and *show* actions.

On the other hand, we have looked over AHS implementing adaptive navigation like AHA! [22], WHURLE [49], GLAM [39] 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 retained two operations from the Brusilovsky typology and one combination of actions from the Stash classification and one adaptation from the GLAM platform. Thus, we conclude that there are four basic modes to select resources in a setting of adaptive navigation support, which are described below.

1 - Selection only mode

- *Description:* it provides a set of resources, which are all proposed to users, i.e, only the selected resources are proposed to users, the other resources are not proposed.
- *Example:* we may propose only definitions.
- *Comparison with existing works:* it 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.

2 - Recommended selection mode

- *Description:* it 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.
- *Example:* in e-learning, we may recommend definitions rather than examples. The user can access to both types of resource, but a typographic indication enables the user to identify which resources are recommended.
- *Comparison with existing works:* it is equivalent to the *adaptive link annotation* described by Brusilovsky. There is no equivalent in the actions described by Stash et al.

3 - Ordered selection mode

- *Description:* it 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.

- *Example:* in e-learning, concepts can be selected and ordered using the prerequisite relation defined between concepts.

- *Comparison with existing works:* it is equivalent to the *adaptive ordering* 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.

4 - Alternate selection mode

- *Description:* it 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.

- *Example:* we propose textual resources when they are available, and audio resources in the absence of textual resources.

- *Comparison with existing works:* neither Brusilosvky nor Stash et al. have considered this selection mode.

4.2 Description of elementary adaptation patterns

Here, we present in Section 4.2.1 a general definition of an elementary adaptation pattern. Afterward in Section 4.2.2 we detail the syntax defined for an elementary adaptation pattern. Finally, in Section 4.2.3, we present the semantic associated to the defined syntax.

4.2.1 Definition of an elementary adaptation pattern

We propose the following definition for elementary adaptation patterns, based on the definition of design patterns [31].

Definition 9 *An elementary adaptation pattern describes a generic solution for a generic elementary adaptation problem.*

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

Definition 10 *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.*

4.2.2 Syntax of an elementary adaptation pattern

Figure 4.8 presents the characteristics retained from [31] 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]. This grammar is described in Figure 4.9. It includes a set of non-terminal elements expressed between brackets, and a set of terminal elements expressed between quoats. For people not familiar with the EBNF syntax, we give examples of the solution part respecting the proposed grammar in Appendix B. These examples are also accompanied by an informal description.

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.

Figure 4.8 – Description of elementary adaptation patterns

4.2.3 Semantic of an elementary adaptation pattern

We give an informal description of the semantics 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 *DM* elements as general elements in the grammar (cf. Figure 4.10). Afterward, we defined predicates to select resources or concepts. The predicates are:

- *instanceOf*: *instanceOf(r, c)* is true, for all resources *r* that are instances of 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 instance *i2* by relation *rel*.
- *linked-transitive*: *linked-transitive(i1, i2, rel)* is true, for all instances *i1* that are linked directly or indirectly to instance *i2* by relation *rel*.
- *distance*: *distance(i1, i2, rel, n)* is true, for all instances *i1* that are distant from instance *i2* by *n* instances using relation *rel*.

These predicates compose 3 types of expressions:

- $<Exp_{cls}>$ for expressions on classes.

⟨*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*⟩ * .

Figure 4.9 – Syntax of the characteristic *Solution*

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

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

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 of a property

Figure 4.10 – Description of general elements

- <*Id1*> | <*Id2*> means that the set of resources selected with the expression identified by *Id2* is an alternative to the one selected with the expression identified by *Id1*.

4.3 Typology of elementary adaptation patterns

We have defined a library of 22 elementary adaptation patterns using the criteria defined in Section 4.1. An elementary adaptation pattern is based simultaneously on:

- one of the 4 selection modes of resources to be proposed;
- one of the 3 elements of the domain model involved in the selection process and when the element is a relation, we also consider:
  - one of the 2 types of navigation through the resources or the concepts graph. The 2 navigation modes are applied for all the selection modes except for the *selection only* mode, which proposes a set of resources selected using criterion on classes, properties or relations.

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 4.11). 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. Appendix B, Table B.2) is used to define *S1-1* (*S1-1* consists of ordering concepts according to a depth-first navigational path using the relation pre-requisite, and presents resources linked to these concepts).

The pattern P4.3 (cf. Appendix B, Table B.4) is used to define *S1-2* (*S1-2* consists of presenting only audio resources if they are available otherwise presents textual resources).

The pattern P2.2 (cf. Appendix B, Table B.2) is used to define *S1 -3* (*S1-3* consists of presenting examples before definitions).

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

5 Using the EAP framework to define adaptation strategies

This Section gives further information on each step of the process of authoring adaptation strategies using the EAP framework (cf. Section 3).

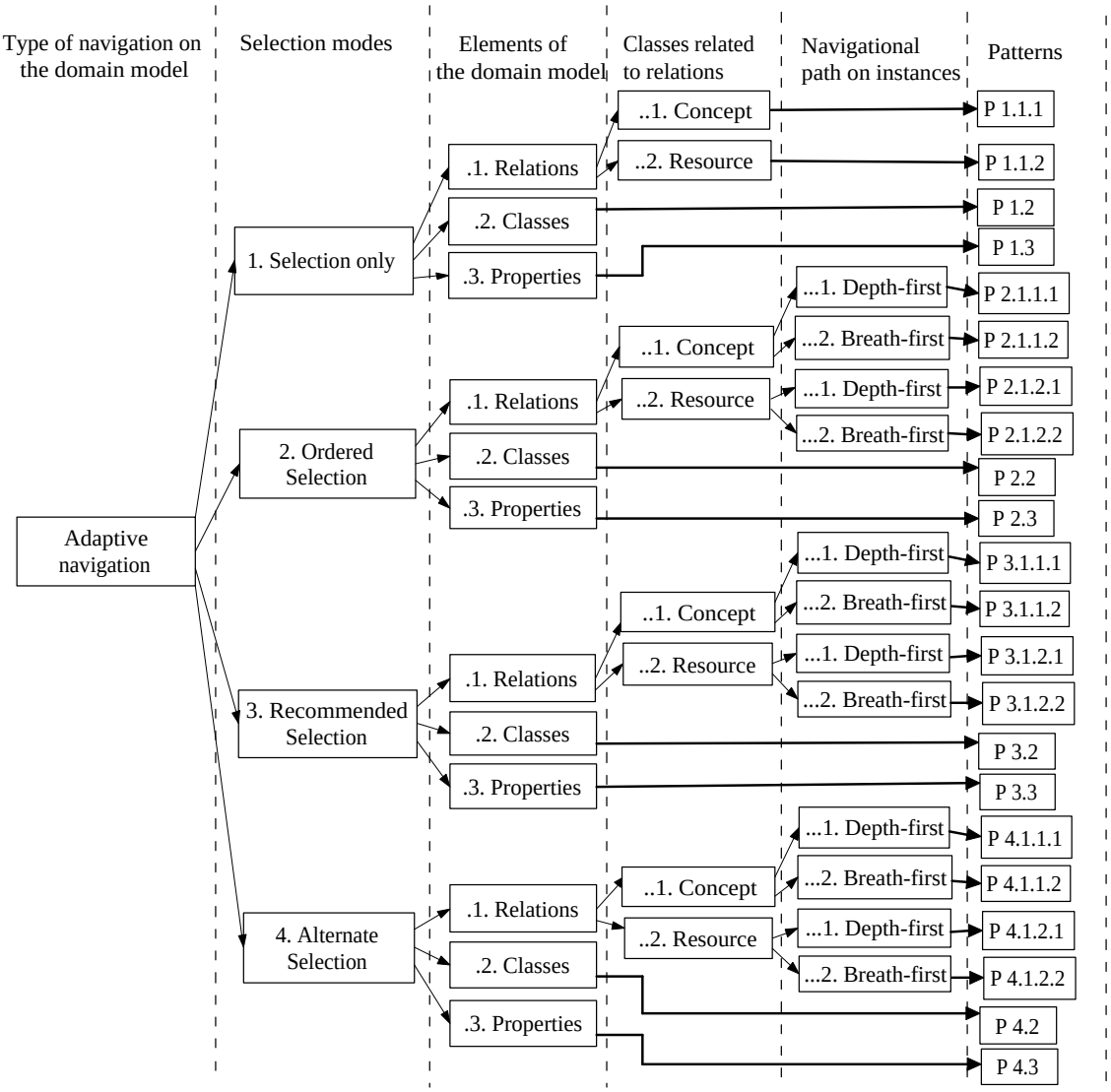

Figure 4.11 – Typology of elementary adaptation patterns

We first describe step 1 related to the instantiation process of elementary adaptation patterns (cf. Section 5.1). Then, we present step 2 related to the association of elementary adaptations with user characteristics, and finally we detail step 3 related to the combination process (cf. Section 5.3).

5.1 Step 1/3: 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.

5.1.1 Elementary adaptations

We define elementary adaptations as follows:

Definition 11 *An elementary adaptation is obtained after an instantiation of an elementary adaptation pattern on a particular domain model.*

Elementary adaptations have 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.

5.1.2 Applying the first step on Jane's use case

When Jane wants to express the *S1-1* part, she selects the pattern *P2.1.1.1* and instantiates it with relation *prerequisite* (for informal description, see Section 2, for formal description see Figure 4.12).

Name : Ordered Selection - Depth first- Prerequisite - Concept

Intent : This pattern proposes resources according to a depth first navigational path on concepts.

Solution :

- Expressions
  E_1*:linked(currentR, concept', abstraction)* $\wedge$ *linked-transitive(concept, goal,* $prerequisite$*)* $\wedge$ *linked(r, concept,* $abstraction$*)* $\wedge$ *linked(concept, concept',* $prerequisite$*)*
  E_2*: linked-transitive(concept, goal,* $prerequisite$*)* $\wedge$ *linked(r, concept,* $abstraction$*)*
  According to E_1: selected resources are linked to concepts using $abstraction$. these concepts can reach the $goal$ using $prerequisite$ 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 $prequisite$.
- 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*.
- currentR: 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.
- prerequisite: 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.

Figure 4.12 – The elementary adaptation *S1-1*

Following this principle, Jane expresses the *S1-2* part. She selects the pattern *P4.3* and instantiates it with the property *format* and attributes *audio* and *textual* (for informal description, see Section 2, for formal description see Figure 4.13).

Name : Alternate Selection-format-audio/text

Intent : This elementary adaptation proposes audio resources if they are available otherwise, it proposes textual resources.

Solution :

- Expressions
  E_1: *characteristicOf (r, format, =, audio)*
  E_2: *characteristicOf(r, format, =, text)*
  According to E_1, selected resources are all of audio format.
  According to E_2, selected resources are all of text format.
- Meta-expressions
  $E_1 \mid E_2$
  According to this meta-expression, only audio resources are proposed if available otherwise textual resources would be proposed.

Constituents :

- r: a variable which represents an instance of the class *Resource* or of one of its specializations.
- format: a variable which represents the property *format*.
- audio (resp. text): a variable which represents value *audio* (resp. *text*) of the property *format*.

Figure 4.13 – The elementary adaptation *S1-2*

Jane also expresses the *S1-3* part. She selects the pattern *P2.2* and instantiates it with the classes *Example* and *Definition* (for informal description, see Section 2, for formal description see Figure 4.14).

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*.
  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*.

Figure 4.14 – The elementary adaptation *S1-3*

5.2 Step 2/3: linking elementary adaptations with user characteristics

After having defined multiple elementary adaptations, each of them must be associated to one user characteristic (step 2 in Section 3).

5.2.1 Defining associations

Each user characteristic has to be associated to one elementary adaptation, in order to be considered in the adaptation proposed to users. When a user characteristic is not associated to an elementary adaptation, that means that this characteristic is not considered in the adaptation.

5.2.2 Applying the second step on Jane's use case

This step is the easiest step. Jane has to associate each elementary adaptation previously defined with user characteristics. She has to associate:

- *S1-1* with *in-depth* learning mode;
- *S1-2* with *inductive* reasoning mode;
- *S1-3* with *audio* presentation form.

When users have a profile composed of several characteristics involved in adaptation, complex adaptation strategies have to be defined. They are obtained by combining elementary adaptations, each one being associated with a characteristic of the user profile. The combination process is described below.

5.3 Step 3/3: combining elementary adaptations

Combining elementary adaptations together defines a combined adaptation.

5.3.1 Process of combining elementary adaptations

We define combined adaptations as follows:

Definition 12 *A combined adaptation defines a set of resources that satisfies all constraints imposed by multiple elementary adaptations simultaneously.*

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 here). 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. But, 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 participates. We express this process in two sequential steps:

1. Build different sets of identifiers of expressions, one set for each different criterion.
2. Build one adaptation from the sets built in previous step.

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 criterion 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}};
- to a set of {Set_{prop}}.

Sets in a group correspond to exluded criteria. For example: a resource is only definition, and it cannot be an exercise at the same time (following classical object model).

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:

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

where

- CE_p 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 with 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)).

	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_{3-1} = instanceOf(r, Example) E_{3-2} = instanceOf(r, Definition)	$E_{3-1} \prec E_{3-2}$
S1-3	E_{2-1} = characteristicOf(r, format, =, audio) E_{2-2} = characteristicOf(r, format, =, text)	$E_{2-1} \mid E_{2-2}$

Figure 4.15 – Description of *S1-1*, *S1-2*, *S1-3*

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.

5.3.2 Applying the third step on Jane's use case

We focus now on the way *S1-1*, *S1-2* and *S1-3* are combined in order to produce $S1$. More precisely, we detail the combination process of their characteristic *Solution*, which is performed automatically as follows. It has as input 3 elementary adaptations expressed on 3 different elements of the domain model.

After step 1 of the combination process, 3 sets are built, one adaptation per set (cf. Figure 4.15).

After step 2, one combined adaptation is built, which is composed of 8 expressions and 44 meta-expressions. The deduced expressions are the following:

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

- $E_{c,4}$ = $E_{1-1} \wedge E_{2-2} \wedge E_{3-2}$ = linked-transitive(r, goal, prerequisite) $\wedge$ linked(rCurrent, r, prerequisite) $\wedge$ characteristicOf(r, format, =, text) $\wedge$ instanceOf(r, Definition)
- $E_{c,5}$ = $E_{1-2} \wedge E_{2-1} \wedge E_{3-1}$ = linked-transitive(r, goal, prerequisite) $\wedge$ characteristicOf(r, format, =, audio) $\wedge$ instanceOf(r, Example)
- $E_{c,6}$ = $E_{1-2} \wedge E_{2-1} \wedge E_{3-2}$ = linked-transitive(r, goal, prerequisite) $\wedge$ characteristicOf(r, format, =, text) $\wedge$ instanceOf(r, Definition)
- $E_{c,7}$ = $E_{1-2} \wedge E_{2-2} \wedge E_{3-1}$ = linked-transitive(r, goal, prerequisite) $\wedge$ characteristicOf(r, format, =, text) $\wedge$ instanceOf(r, Example)
- $E_{c,8}$ = $E_{1-2} \wedge E_{2-2} \wedge E_{3-2}$ = linked-transitive(r, goal, prerequisite) $\wedge$ characteristicOf(r, format, =, text) $\wedge$ instanceOf(r, Definition)

Deducing meta-expressions is more complicated, we summarize the process in Table 4.2. It has four columns: the first column includes meta-expressions belonging to the initial elementary adaptations. The second column includes the deduced meta-expressions belonging to $S1$. The third column presents the meta-expressions in conflict with the deduced one and the fourth column presents either the retained meta-expression or why the deduced meta-expression was not retained. We also crossed the meta-expressions that have been already deduced.

S1-1, *S1-2*, *S1-3* Meta-expressions	*S*1 meta-expressions	conflicts with existing ones	retained/ not retained meta-expressions
$E_{1-1} \prec E_{1-2}$	$E_{c,1} \prec E_{c,5}$		$E_{c,1} \prec E_{c,5}$
	$E_{c,1} \prec E_{c,6}$		$E_{c,1} \prec E_{c,6}$
	$E_{c,1} \prec E_{c,7}$		$E_{c,1} \prec E_{c,7}$
	$E_{c,1} \prec E_{c,8}$		$E_{c,1} \prec E_{c,8}$
	$E_{c,2} \prec E_{c,5}$		$E_{c,2} \prec E_{c,5}$
	$E_{c,2} \prec E_{c,6}$		$E_{c,2} \prec E_{c,6}$
	$E_{c,2} \prec E_{c,7}$		$E_{c,2} \prec E_{c,7}$
	$E_{c,2} \prec E_{c,8}$		$E_{c,2} \prec E_{c,8}$
	$E_{c,3} \prec E_{c,5}$		$E_{c,5} \prec E_{c,5}$
	$E_{c,3} \prec E_{c,6}$		$E_{c,5} \prec E_{c,6}$
	$E_{c,3} \prec E_{c,7}$		$E_{c,5} \prec E_{c,7}$
	$E_{c,3} \prec E_{c,8}$		$E_{c,5} \prec E_{c,3}$
	$E_{c,4} \prec E_{c,5}$		$E_{c,6} \prec E_{c,5}$
	$E_{c,4} \prec E_{c,6}$		$E_{c,6} \prec E_{c,6}$
	$E_{c,4} \prec E_{c,7}$		$E_{c,6} \prec E_{c,7}$
	$E_{c,4} \prec E_{c,8}$		$E_{c,6} \prec E_{c,4}$
$E_{2-1} \mid E_{2-2}$	$E_{c,1} \mid E_{c,7}$	$E_{c,1} \prec E_{c,7}$	priority is given to $\prec$
	$E_{c,1} \mid E_{c,3}$		$E_{c,1} \mid E_{c,3}$
	$E_{c,1} \mid E_{c,4}$	$E_{c,1} \prec E_{c,4}$	priority is given to $\prec$
	$E_{c,1} \mid E_{c,8}$	$E_{c,1} \prec E_{c,8}$	priority is given to $\prec$
	$E_{c,2} \mid E_{c,7}$	$E_{c,2} \prec E_{c,7}$	priority is given to $\prec$
	$E_{c,2} \mid E_{c,3}$	$E_{c,2} \prec E_{c,3}$	priority is given to $\prec$
	$E_{c,2} \mid E_{c,4}$		$E_{c,2} \mid E_{c,4}$
	$E_{c,2} \mid E_{c,8}$		$E_{c,2} \mid E_{c,8}$
	$E_{c,5} \mid E_{c,7}$		$E_{c,5} \mid E_{c,7}$
	$E_{c,5} \mid E_{c,3}$	$E_{c,3} \prec E_{c,5}$	priority is given to $\prec$
	$E_{c,5} \mid E_{c,4}$	$E_{c,4} \prec E_{c,5}$	priority is given to $\prec$
	$E_{c,5} \mid E_{c,8}$	$E_{c,5} \prec E_{c,8}$	priority is given to $\prec$
	$E_{c,6} \mid E_{c,7}$	$E_{c,7} \prec E_{c,6}$	priority is given to $\prec$
	$E_{c,6} \mid E_{c,3}$	$E_{c,3} \prec E_{c,6}$	priority is given to $\prec$
	$E_{c,6} \mid E_{c,4}$	$E_{c,4} \prec E_{c,6}$	priority is given to $\prec$
	$E_{c,6} \mid E_{c,8}$		$E_{c,6} \mid E_{c,8}$

$E_{3-1} \prec E_{3-2}$	$E_{c,1} \prec E_{c,2}$		$E_{c,1} \prec E_{c,2}$
	$E_{c,1} \prec E_{c,4}$		$E_{c,1} \prec E_{c,4}$
	~~$E_{c,1} \prec E_{c,6}$~~		(repetition)
	~~$E_{c,1} \prec E_{c,8}$~~		(repetition)
	$E_{c,5} \prec E_{c,2}$	$E_{c,2} \prec E_{c,5}$	adaptation on relations is a higher priority
	$E_{c,5} \prec E_{c,4}$	$E_{c,4} \prec E_{c,5}$	adaptation on relations is a higher priority
	$E_{c,5} \prec E_{c,6}$		$E_{c,5} \prec E_{c,6}$
	$E_{c,5} \prec E_{c,8}$		$E_{c,3} \prec E_{c,8}$
	$E_{c,3} \prec E_{c,2}$		$E_{c,3} \prec E_{c,2}$
	$E_{c,3} \prec E_{c,4}$		$E_{c,3} \prec E_{c,4}$
	~~$E_{c,3} \prec E_{c,4}$~~		(repetition)
	~~$E_{c,3} \prec E_{c,8}$~~		(repetition)
	$E_{c,7} \prec E_{c,2}$	$E_{c,2} \prec E_{c,7}$	adaptation on relations is a higher priority
	$E_{c,7} \prec E_{c,4}$	$E_{c,4} \prec E_{c,7}$	adaptation on relations is a higher priority
	$E_{c,7} \prec E_{c,6}$		$E_{c,7} \prec E_{c,6}$
	$E_{c,7} \prec E_{c,8}$		$E_{c,7} \prec E_{c,8}$

Table 4.2 – Process of deducing meta-expressions of $S1$

6 Summary

This chapter proposes the EAP framework, in which adaptation strategies are defined at a finer granularity than when using existing adaptation languages. This is obtained by the definition of 22 elementary adaptation patterns (cf. Appendix B) to express the adaptive navigation. We have proposed a typology to organize them and to be able to look easily over.

The elementary adaptation patterns are based, on the one hand, on a criterion to select resources and, 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 any 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, as we have illustrated it on all the chapter using Jane's use case.

The generated adaptation strategies are expressed at a high level and are independent of any adaptation engine. In the next chapter, we are going to describe how they could be executed by existing adaptation engines.

CHAPTER 5

Expressivity of EAP framework versus GLAM, LAG

After having studied existing solutions allowing to express adaptation (cf. Chapter 4, Section 1.2), and having proposed the EAP framework (cf. Chapter 4), that introduces a finer granularity in expressing adaptation, several questions have come out, like: *what facilities do the EAP framework and existing solutions propose to authors?*, *Do the EAP framework and existing solutions allow expressing similar adaptation?* or *Are there lessons learned from the design of the EAP framework and existing solutions that could suggest features that should be in any adaptation language?*

Answering such questions leads to study the expressivity of the EAP framework versus existing solutions. There are several existing solutions. Each one has its specificities and constraints for expressing adaptation. It is not possible to study deeply all existing solutions. Therefore, choices have to be made. However, existing solutions can be grouped in two main groups[1]: a group allowing to express adaptation using condition-action or event-condition-action rules [39, 22], and a group allowing to express adaptation using procedural languages [12, 65]. Consequently, we have decided to study the expressivity of the EAP framework versus a solution of each group.

- The GLAM platform will be the solution we are going to study for the first group. It is the only system that today supports the alternate selection mode (cf. Chapter 4, Section 4.1.2). Furthermore, GLAM has multiple characteristics in common with the EAP framework as modeling the domain model and the goal of users.
- The LAG language will be the solution we are going to study for the second group. In addition of proposing a different approach for modeling AHS, it is plugged on several existing adaptation engines, like AHA!.

First, we have studied the way to model elements on which the adaptation is expressed (cf. Section 1). Afterward, we have studied the way to model adaptation (cf. Section 2).

This chapter presents two major outcomes.

1. A unifying vision of modeling the available resources used by the EAP framework, GLAM and LAG. This unifying vision is built on three levels: a generic, a specific and an instance level which can be applied on other adaptive hypermedia systems.
2. A unifying typology of adaptation used by the EAP framework, GLAM and LAG. This new typology describes on one hand, the various selection operations that can be done on the available resources, and on the other hand the main basic actions that are generally performed to build adaptation. The basic actions define operations on resources previously selected according to different selection processes. The selection processes and the basic actions use different criteria, whether they are performed in the EAP framework, GLAM and LAG. We hence discuss the criteria used in each of them.

The work relative to the LAG language has been done in collaboration with Alexandra Cristea[2], the author of this language.

[1] We eliminate the group on *Hypertext and adaptation patterns* as they lack in formalization
[2] www.dcs.warwick.ac.uk/∼acristea/

1 Study of the expressivity of domain models used by the EAP framework, GLAM and LAG

For modeling available resources, the EAP framework exploits a domain model, expressed in OWL, GLAM also exploits a domain model, but expressed in RDF(S)[3], while LAG exploits two models, a domain and a goal model, expressed in the CAF [18] format.

Additionally, the domain model used by the EAP framework or by GLAM are organized differently than the model used by LAG. For the domain model used by the EAP framework or GLAM, it supports any graph structure modeling the available instances. This is due to the fact that the EAP framework or GLAM are not specifically designed for educational applications, as well as from its roots lying in semantic web technologies. Conversely, the domain and goal models used by LAG require that their instances to be modeled in a hierarchy (and thus participate obligatorily in a hierarchical relation, although others are permitted). This is due to the fact that CAF is mainly aimed at educational resources, which are traditionally organized in hierarchies.

In the following, we use the notations *DM* for domain model and *GM* for goal model. Furthermore, we model the elements of DM and GM used by the EAP framework, by GLAM and by LAG in UML. This choice is mainly motivated by the fact that UML is, a widely accepted standard, as well as a formal and a concise language. Afterward, we propose an unifying approach for modeling DM and GM used by the EAP framework, by GLAM and by LAG. Finally, we discuss differences of modeling DM and GM used by the EAP framework, by GLAM and by LAG.

1.1 UML representation of domain models used by the EAP framework, GLAM and LAG

We present successively, UML representation of DM used by the EAP framework, then DM used by GLAM, finally, DM used by LAG.

1.1.1 UML representation of domain models used by the EAP framework

The EAP framework considers domain models composed of a set of classes, of properties and of relations (cf. Chapter 4, Section 3). The generic domain model in the EAP framework (cf. Figure 5.1) makes a clear separation between concepts (in a *Concept* class) and resources (in a *Resource* class). The *Concept* and *Resource* classes have both of them a *name* attribute. The *Resource* class has additionally a *content* attribute and is linked to a *Property* class.

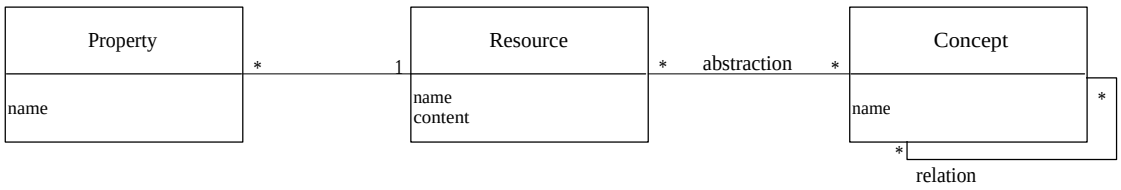

Figure 5.1 – A UML class diagram of the generic domain model used by the EAP framework

The *Property* class describes the characteristics of the resources. A resource may have several characteristics, but a characteristic refers to only one resource. The *Property* class can be

[3]http://www.w3.org/TR/rdf-schema/

specialized several times. It has been modeled as a class instead of a property in order to propose more extensibility, the number of properties of a resource being not defined a priori. The concepts are linked together by at least one relationship, and resources can be also linked together. The semantics of these two relationships is not a priori defined but they can be hierarchical relations or relatedness relations (as in the model used by LAG or others). All the multiplicities are multiple in both cases. Furthermore, a concept is described by one or more resources. This is modeled by the abstraction relationship.

1.1.2 UML representation of domain models used by GLAM

The generic GLAM domain model (cf. Figure 5.2) includes:

- a *Constituent* class that models available concepts;
- a *Resource representation* class that models available resources that may have an undefined number of properties;
- a relation having as domain and range the *Constituent* class;
- a relation having as domain the *Constituent* class and as range the *Resource representation* class;
- a relation having as domain and range the *Resource representation* class;
- a set of classes, that we are not going to cite, which models mathematical properties of the proposed relations. These classes are proposed to allow the GLAM adaptation engine to reason on available resources while executing AH.

As we can note, except classes modeling mathematical properties of relations, all the other elements of the GLAM generic domain model are also elements in the domain model used by the EAP framework. Consequently, the generic domain model used by the EAP framework can be seen as an abstraction of the generic domain model used by GLAM.

1.1.3 UML representation of domain models used by LAG

Figure 5.3 is the UML class diagram representing the domain (and goal) models used by LAG. Concepts are represented by the *Domain_Model* class. They are characterized by attributes. The *Attribute* class describes resources viewed as components of a given concept. Both classes have a *name* attribute, which in the case of attributes, denotes, for simplicity, the type of an attribute (e.g., an attribute called 'theory' is of the type 'theory'). The *Attribute* class has additionally a *content* attribute, which points to the actual resource. Hierarchical relationships are also allowed between domain concepts, represented here as a *relation* between two instances of the *Domain_Model* class with the cardinalities 1..*. Additionally, *relatedness* relationships can be defined between two concepts (or two instances of the *Domain_Model* class) with both multiplicities equal to 1 (this allows to represent that 'a concept A can be related to another one'). *Hierarchical* relations of concepts are obligatory in the MOT system (cf. Chapter 2, Section 2.1.2), i.e., all concepts need to have a parent concept, with the exception of the root concept. *Relatedness* relations depict other types of relations between concepts in the domain. Please note however that all these relations (*hierarchical*, *relatedness*) are domain-specific relationships, and are separated from pedagogical relations (such as prerequisites[4]), under the principle of separation of concerns.

The goal model filters and restructures the domain model, with respect to pedagogical goals. This is depicted here via the *Goal_Model* class linked to the *Attribute* Class and having attributes

[4]In LAG, prerequisite is modeled in the goal model.

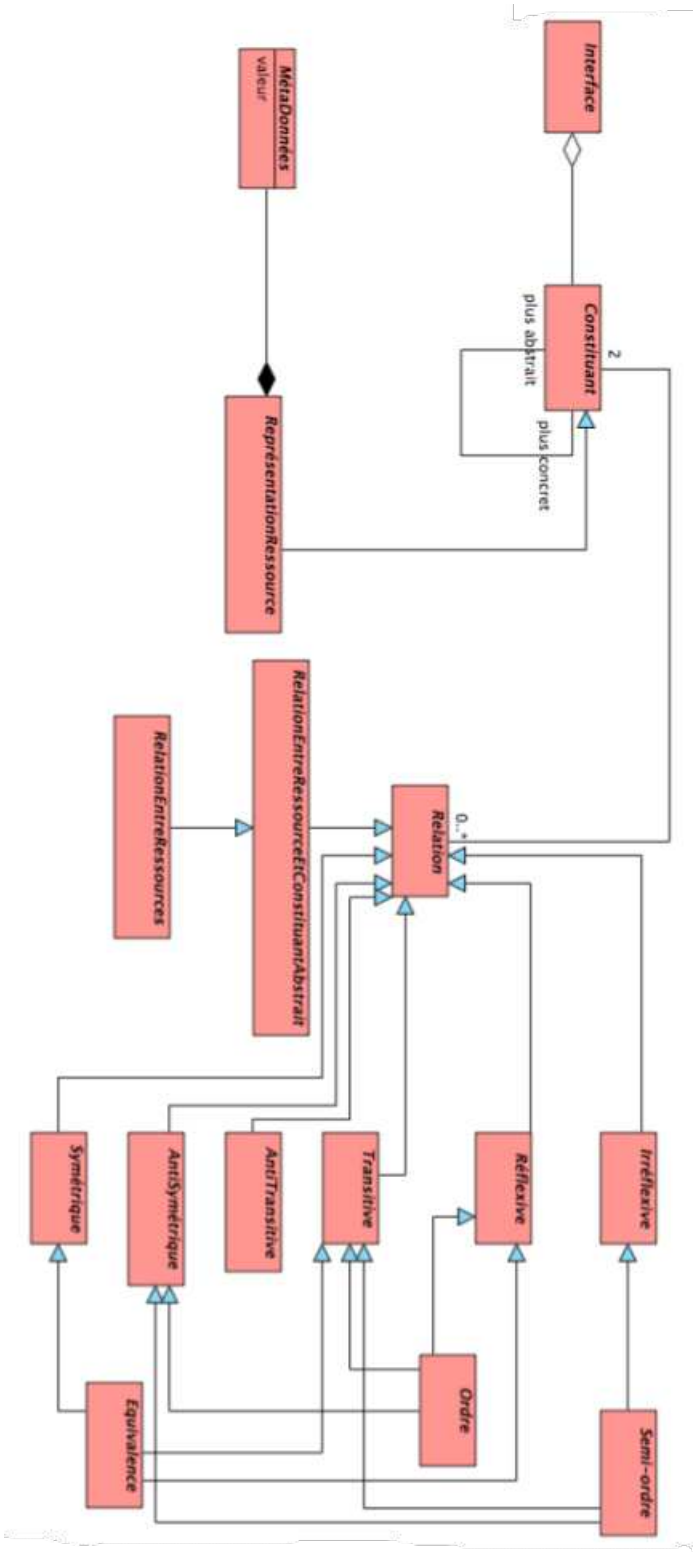

Figure 5.2 – A UML class diagram of the generic domain model used by GLAM [38]

usable to represent pedagogical characteristics: a *label* of string type (which can have values such as, e.g., 'visual' or 'verbal' to denote content appropriate for those respective types of learners) and a *weight* of integer type (which can additionally describe the label, e.g., a visual content with weight '100' could be suitable for visual learners only, whereas a visual content with weight

'30' would be also usable by others). Any type of desired pedagogical characteristics can be added this way. The association class, *Link*, is depicting the association between *Goal_Model* and *Attribute* in order to include valuable information: a string named *label*, an integer named *weight* and a supplementary integer named *order*. The latter represents one aspect of the *prerequisite* relation, the ordering of desired resources, according to the pedagogical goals. Please note that the concept hierarchy and order in the goal model can be completely different from the hierarchy in the domain model, as the former are a reflection of pedagogical goals and planning, whilst the latter is only a new of the domain, regardless of how it will be pedagogically interpreted. This separation of concerns allows for the same domain model to be used with different pedagogies and thus with different goal models.

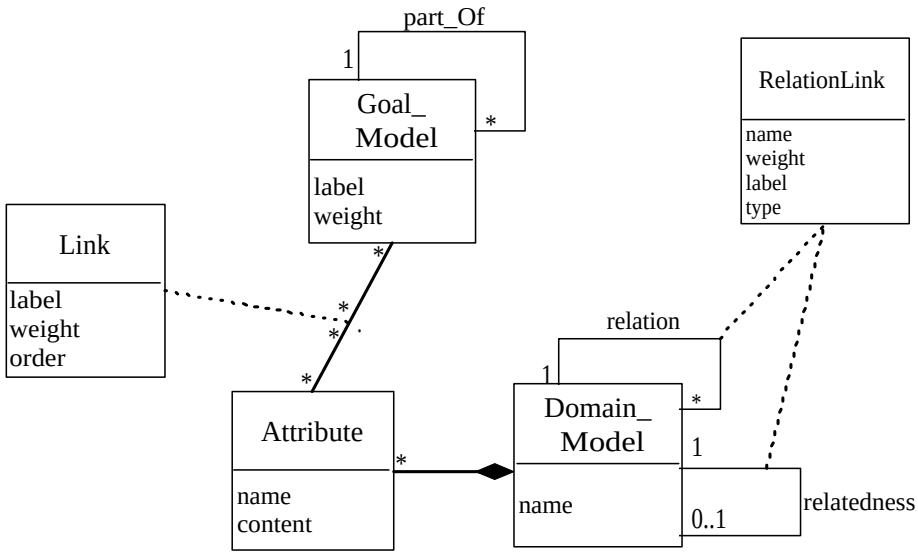

Figure 5.3 – A UML class diagram representing the domain and goal models used by MOT

1.2 A unified vision of the domain model used by the EAP framework, GLAM and LAG

1.2.1 A unified vision of the domain model in AH

Modeling the domain model is mostly expressed at three levels of modeling, even if it is not often shown explicitly.

1. A generic level is for any (generic) author, independently of the application domain, to create any AH.

2. A specific level is for a particular author (or several authors) to create a particular class of AH, in which he (they) consider(s) the particularities of his (their) application domain.

3. An instance level is also for a particular author, but the aim is to create a particular AH and to specify the actual resources to be used by the adaptation, not just the class of AH.

For example, in the case of AEH, the generic level enables authors to describe the structure of any course, the specific level enables them to describe the structure of a particular course and the instance level includes particular lessons to be presented to users.

These three levels are thus useful to represent content of resources to be proposed to users, relations that are defined between resources and possible characteristics about them.

The three levels of modeling are used to describe DM (and GM) models used by the EAP framework, GLAM and by LAG. Note that, the DM used by the EAP framework is very close to the one used by GLAM. For this reason, in our study we speak only of the EAP framework and LAG. In the following, we first present the unifying vision of the domain (and goal) models used by the EAP framework and LAG. Then, we illustrate the unifying vision on a running scenario involving Jane (cf. Chapter 4, Section 2) by using the EAP framework and LAG.

1.2.2 Applying the unified vision on DM (and GM) used by the EAP framework and LAG

The EAP framework exploits a single model, DM, which includes in the same model the structure of the available resources and the modeling of the goal, thus forgoing the paradigm of separation of principles, and the reusability, in favor of compactness. The EAP framework can be modeled according to the three levels (cf. right side of Figure 5.4). The generic level includes the generic domain model. The specific level includes the specific domain model which is a specialization of the generic domain model, and the instance level describes the instances of the specific model.

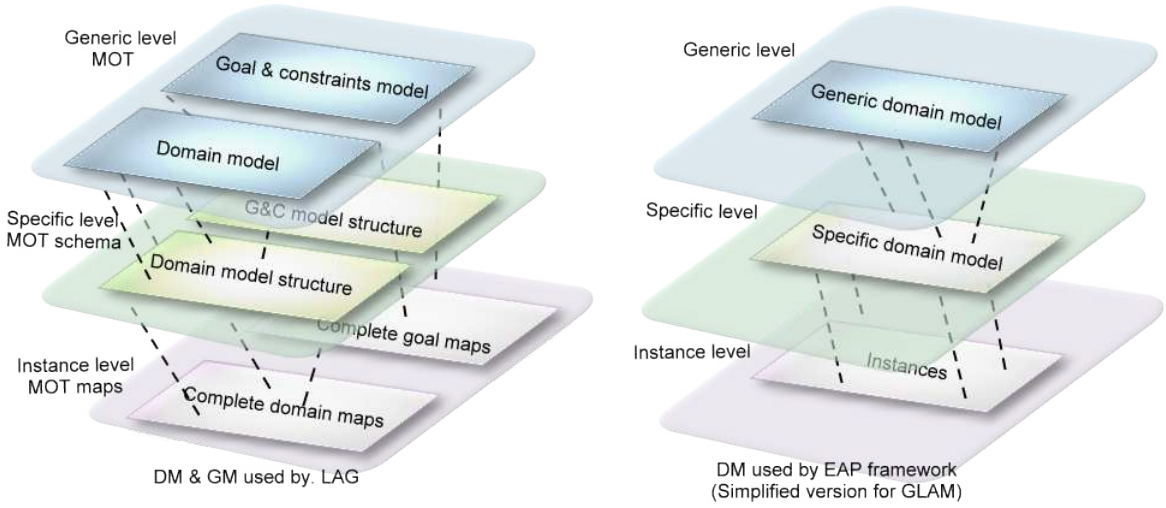

Figure 5.4 – The three domain (goal) modeling levels used in each approach

On the other hand, LAG exploits DM and GM that can be defined according to three levels as described above (illustrated on the left side of Figure 5.4). At the generic level, two models are described: the domain model which describes the possible structures for any available resources and the goal and constraints model which presents all possible reorganizations of the structure of the available resources. At the specific level, two models are also described, which are instantiations of the models at the generic level: a specific domain model and goal model structure, which are instantiations of the domain and goal model, respectively. At the instance level, two concrete maps are described that are instances of the models included in the specific model: the complete domain maps and complete goal map, respectively.

Now, we analyze in Figure 5.5 the DM (and GM) used by the EAP framework and by LAG at each level of modeling in more details. Note that the transition between the generic level and the specific level is performed differently in each approach.

About the DM used by the EAP framework. The transition between the generic level and the specific level is a specialization process. The *Resource* Class has to be specialized by at least one class. The number of specializations of the *Property* class corresponds to the number of specific properties useful for characterizing the resources of a given application domain. This corresponds somewhat to the attributes describing a concept in the LAG approach. Moreover, the semantics of the relation defined on the *Concept* and on the *Resource* classes

have to be defined at this level. In Figure 5.5, at the specific level, the *Concept* and the *Resource* classes appear with a dotted outline to indicate that they do not belong to the specific level.

About the DM and GM used by LAG. The transition between the generic level and the specific level is an instantiation process. Some classes are instantiated at the specific level, whereas other classes are instantiated later at the instance level. At the specific level, in the DM, the *Attribute* class is instantiated as many times as desired with values for the attribute name. Furthermore, also in the DM, the *hierarchical* relationship between instances of the *Domain_Model* is also instantiated at the specific level. The instantiation can correspond to a composition relationship (*part_Of*), and is used to specify the characteristics describing a concept. Also at the specific level, in the GM, the ordering and pedagogical labeling of concepts pointing to the attributes in the DM is performed. This allows for the same DM to be reused in different pedagogical settings, with different pedagogical characteristics.

The transition between the specific level and the instance level is similar for DM (and GM) used by each approach. It consists on an instantiation process where the author specifies the actual content of his resources and their characteristics.

So far, we have described the three levels of modeling of the DM (and GM) used by the EAP framework and by LAG. In the following, we revisit these three levels of modeling for a concrete example: Jane's use case (cf. Chapter 4, Section 2) using either the EAP framework or the LAG approach.

1.2.3 Applying the unified vision on Jane's use case for EAP framework, GLAM and LAG

When Jane uses either the EAP framework or LAG, she has to describe the resources and the goals to be achieved by her target learners in the DM (and GM) understood by the used language. For this purpose, in terms of the three layers described above, Jane has to use the generic level and define the specific and the instance levels. In this section, we illustrate in Figure 5.6 the results of this process in the case of Jane using the EAP framework and LAG, respectively, to define her scenario.

In the first case, we consider Jane uses the EAP framework. She first specializes the classes belonging to the model of the generic level as follows.

- She defines a *pre-requisite* relationship between concepts. No relationship is defined between resources.
- She defines three specializations of the *Resource* class: *Definition*, *Example* and *Exercise*.
- She defines a *Format* class as a specialization of the *Property* class. She decides that the *name* attribute of the *Format* class can have the value *text* to denote a textual resource, *image* to denote an image resource, or *audio* to denote a video resource.

Then, Jane specifies her instances and thereby she works at the instance level. Due to lack of space, the instance level in Figure 5.6 describes only the modeling of the *database* concept.

For the instance level used by the EAP framework, we have the following instances:

- three instances of the *Concept* class. They are linked by the *prerequisite* relationship, one per concept having to be modeled;
- four instances of the *Resource* class;
- two instances of the *Format* class.

Figure 5.5 – The three modeling levels in UML of the DM (and GM) used by each approach

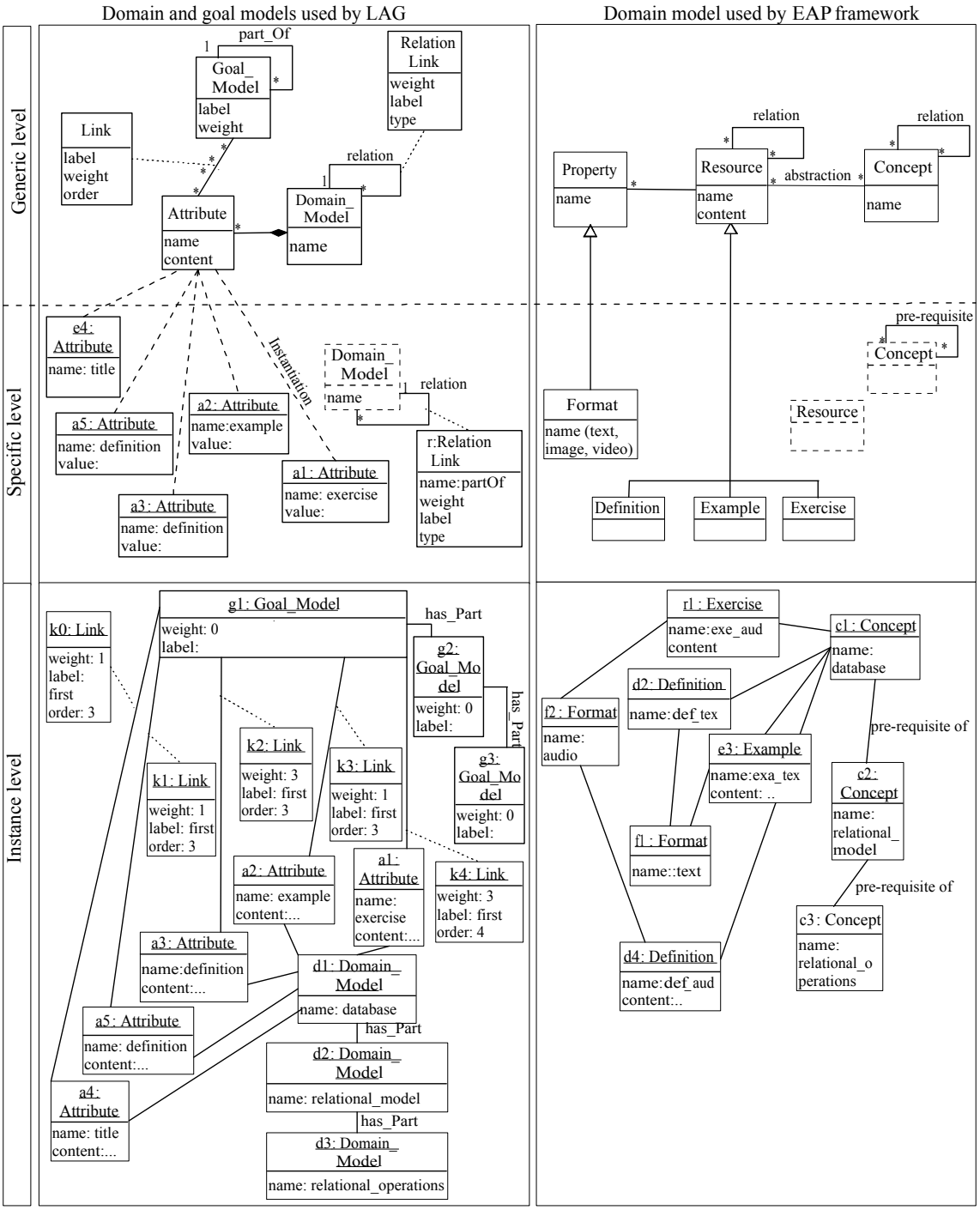

Figure 5.6 – Example of the three modeling levels of the domain (and goal) models used by each approach

Jane also models links between instances of the *Resource* class and of the *Format* class and between the instances of the *Resource* class and of the *Concept* class.

In the second case, we consider Jane uses LAG. She first instantiates the classes belonging to the models of the generic level as follows.

- She decides to use the *hierarchical* relation in a domain model, and not the *relatedness* relation, for instance. In terms of the three layer model, this is equivalent to only define

one instance of the *RelationLink* association class and initialize its name Attribute with the value *part-of*.

- She then defines five instances of the *Attribute* class: one instance with *title* as the value of the *name* attribute, the other ones with *definition* (twice), *exercise* and *example*, respectively. In this way she can decide which characteristics will be used to describe all the concepts of her domain.

Next, Jane specifies her instances and thereby she works at the instance level. This means defining instances of classes and links between instances. Due to lack of space, the instance level in Figure 5.6 describes only the modeling of the *database* concept. In the concrete example illustrated here, this includes the definition of:

- three instances of the *Domain_Model* class, in the form of three domain concepts. The instances are linked by the *has_part* relationship, one per concept having to be modeled;
- additionally, it requires the definition of three domain attribute types, that of *definition*, *example* and *exercise* (beside of the *title* type generated automatically), to be used by the three instances. This corresponds to modeling links between the instances of the *Attribute* class and of the *Domain_model* class;
- the automatic generation of three instances of the *Goal_Model* class, one per instance of the *Domain_Model* class;
- the automatic generation of five instances of the *Link* association class, one instance per instance of the *Attribute* class. This corresponds to modeling links between the instances of the *Goal_Model* class and of the *Attribute* class. Moreover, here, Jane can add pedagogical labels. She uses for the *label* attribute of the *Link* class the value "first" if the instance refers to a resource to be proposed first, otherwise she leaves an empty value. She also uses the value of the *weight* attribute of the *Link* class as "1" for *textual* resources, "2" for *image* resources, "3" for *audio* resources and "0" for other formats.

1.3 Differences of modeling the domain model used by EAP framework, GLAM versus LAG

We have chosen to study the particularities of modeling the domain model used by the EAP framework, GLAM and LAG according to common criteria. These criteria include: *modeling properties of resources*, *modeling only resources* and *modeling of relations*.

Modeling properties of resources

- *The EAP framework/ GLAM*: properties of resources can be as many as needed. They are defined as specializations of the class *Property* of the generic domain model used by the EAP framework and GLAM.
- *LAG*: properties of resources are modeled in the *label* and a *weight* attributes of the *Link* class. When it is necessary to model more than two properties, modeling becomes harder or even impossible. For example, resources may have a format (text or image), a difficulty level (easy, high) and a rank (from 1 to 10). Only two of these three properties can be easily modeled in models used by LAG (although hacks were possible and include encoding two or more properties as a string, but this is not a straightforward method of authoring). Recently, this restriction has been lifted from the systems using LAG, as it was not pedagogy-specific, but only an artificial restriction introduced by implementation limitations. Currently, any number of pedagogical labels and weights can be added to resources in the Goal Model (GM). This is conforming to the LAOS theoretical framework which LAG is based on.

Modeling only resources

- *The EAP framework/ GLAM*: they support a clear separation between modeling concepts and resources. In fact, we can model resources independently from concepts. Therefore, they enable reusing existing taxonomies which are referenced and widely used.

- *LAG*: the existence of the *Attribute* Class depends on the existence of the *Concept* class. For example, we cannot model resources linked by a successor relation without modeling the corresponding concepts. We must define a concept for each resource and must define at least a hierarchical relation between concepts. This is due to the fact that, the vision of people who proposed LAG is that authors will model concepts first, and relate these concepts to specific resources at a later stage. This is very important for reusability of the domain: the same concept can be expressed via various resources. When adaptation is defined, this is applied at the level of concepts, or characteristics of concepts, and again not for particular resources. Such adaptation would never be reusable, as the actual pedagogy behind it is lost [1, 3].

Modeling relations

- *The EAP framework/ GLAM*: the semantics of relationships between concepts or between resources is defined by the author. This allows a lot of freedom in the design, and a great adaptability. As there is not great distinction between pedagogical relations and any other types of relations, they are all instances of the *relation* having as domain and range the *Concept* class. Relations are not predefined. This can be seen as a disadvantage but on an other hand, it is also a strong point because when the application domain is not known in advance, we do not necessarily know all the possible relationships to consider. This freedom allows authors to define all the relations of their choice.

- *LAG*: only five types of relations can be defined:

  - a hierarchical relationship between concepts linking instances of the *Domain_Model* class. Only one relation of that kind can be defined;
  - a relatedness relationship between two concepts;
  - a relationship between resources and concepts, which is defined implicitly;
  - an order relation between goal model concepts, corresponding to prerequisite relations;
  - a hierarchical relation (a sort of composition relation) between goal model concepts, corresponding also to a prerequisite relation.

  This again is not a limitation of the theory behind LAG, as the LAOS framework allows for any type of relations. However, in practice it was observed that teachers tend to use hierarchical structures and prerequisites mostly (if not only) [40]. So allowing them a slightly larger choice was considered, in the first rounds of implementation, sufficient.

So far, we have analyzed these three approaches, the EAP framework, GLAM and LAG, from the content description and manipulation point of view. Both of them propose the modeling of the DM (and GM) in three levels, which has the main advantage to propose distinct visions of the DM (and GM) depending on the level at which authors are interested. In the following, we shall visit the description of adaptation in the EAP framework, GLAM and LAG, respectively.

2 Study of adaptation expressivity of EAP framework, GLAM and LAG

The EAP framework, GLAM and LAG allow expressing similar adaptations using different criteria. In fact, the EAP framework is based on a typology of elementary adaptation patterns proposed in [75], which is based on three criteria: selection modes, elements of the domain model and navigational paths. At the same time, LAG is loosely related to the classification of external actions in AHS as proposed in [69], in which the classification distinguishes basic actions on items, hierarchical actions on items, actions on groups of items and actions on the overall environment, where items can be any component in the framework used, from resources, to concepts, to relations, to whole models. GLAM uses a set of rules and meta-rules to express adaptation. The possible types of adaptation in GLAM are coded in rules and meta-rules.

We first propose, in Section 2.1, an integrated vision of basic actions that can be performed to define adaptation whether using the EAP framework, GLAM and LAG. Afterward, we review the main differences of modeling adaptation using the EAP framework, GLAM and LAG.

2.1 An integrated model for a taxonomy of basic adaptations (based on the EAP framework, GLAM and LAG)

We propose here a new typology (cf. Figure 5.7) aiming at identifying basic actions that can be performed to build adaptation. Adaptation allows defining several adaptation strategies, where an adaptation strategy specifies "*which resources have to be proposed and how they will be proposed to a set of users who share the same characteristics*" [75]. Consequently, in our new typology we distinguish the basic actions having to be performed on the resources from the selection process of resources (action types on resources).

- The selection processes define criteria allowing exploiting resources. They can be related to elements of the domain (and goal) models, user model or presentation model.
- The basic actions define operations on individual resources previously selected according to different selection processes. They have been obtained after we have analysed techniques proposed in most of the used architecture and adaptive systems [22, 39, 49].

The selection processes and the basic actions use criteria which are similar in the EAP framework and GLAM, but different from the ones used in LAG. We present each of them, for the purpose of comparison and extraction of a more generic framework. In this typology, adaptation is studied from two points of view: one based on the internal environment, and another based on the external environment.

The internal environment is considered as being composed of domain and goal concepts, resources, relations, and the models that support them. The external environment is considered here as being composed of presentation-dependent aspects, environment-related aspects, context-related aspects, etc., which can also be modelled as concepts, resources and their relations, and can be divided into various models, depending on the framework used (e.g., AHAM [74], LAOS [13], GAHM [53], GAM [25], etc.).

Concerning actions based on the internal environment, they describe basic actions related to adaptive navigation support and to adaptive content, according to Brusilovsky's taxonomy [6]. They are presented in a typology in the upper half of Figure 5.7. We present each column of this typology going from the left side to the right side.

The EAP framework as GLAM use four criteria to select resources and to perform basic actions. These four criteria are related to the elements of the domain model used by the EAP framework (or GLAM) where resources are structured and described. Furthermore, as the EAP

framework uses a domain model expressed in an object model, thereby, the domain model includes a set of classes, a set of relations and a set of properties (cf. Chapter 4, Section 3). Consequently, the criteria used to select resources concern:

- their belonging to a class in the Resource hierarchy;
- the values of some properties (any property defined as a specialization of the *Property* class);
- the presence of a relation between resources;
- the presence of a relation between concepts linked to resources.

Note that, we may consider the same criteria for GLAM in a simplified manner.

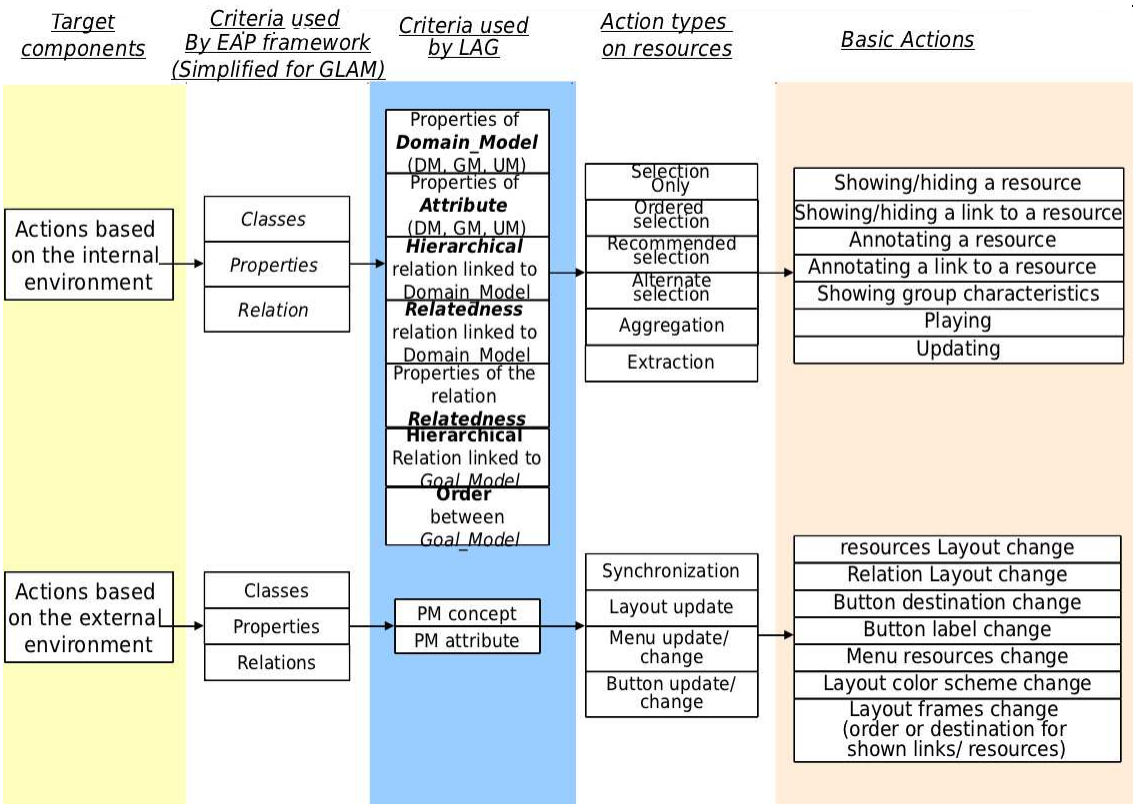

Figure 5.7 – An integrated model for a taxonomy of basic adaptations (based on the EAP framework, GLAM and LAG))

LAG uses seven criteria to select resources and to perform basic actions. They concern elements of the domain and goal models, and also of the user model (cf. Section 1.2). Thereby, the seven criteria concern:

- properties of concepts modeled in the *Domain_Model* and *Goal_Model* class. Concepts are exploited either from the domain model (DM) using their *name*, or from the goal model (GM) using their pedagogical *label* and *weight* or from the user model (UM) using any property indicating the progress of the user in the goal model;
- properties of resources modeled in the *Attribute* class. Similarly to concepts, resources are exploited either from the domain model using their *name*, or from the goal model using their *label* and *weight*, or from the user model using any property indicating their progress in the goal model;

- *hierarchical* relations defined between domain concepts;
- *relatedness* relations defined between domain concepts;
- properties of the *relatedness* relation defined between two concepts. These properties are either the *label* or the *weight* of the relation;
- *hierarchical* relations defined between goal concepts (mostly interpreted in the adaptation specification as prerequisites);
- order relations between goal concepts (mostly interpreted in the adaptation specification as prerequisites).

Note that there are more criteria to select resources used by LAG than by the EAP framework or GLAM. This is due to the fact that LAG uses only two types of relationships (*hierarchical* and *relatedness*) explicitly specified at the generic level, while the EAP framework (or GLAM) uses any type of relationships. As the types are not predefined, they are not specified at the generic level. In the EAP framework (or GLAM), the meaning of the relations having to be considered is defined at the specific level (cf. Section 1.2).

Various actions can be performed on selected resources. We distinguish in the 4th column of the typology between actions for adaptive navigation support [77], and actions triggering adaptive content. For actions proposing adaptive navigation support, we refer to those defined in the EAP framework (cf. Chapter 4, Section 4.1.1).

For actions proposing adaptive content, we propose two main actions:

- Aggregation: it consists of building one resource from two or more resources;
- Extraction: it consists of extracting a resource. This entails cutting out a particular part of a resource in order to define a new resource.

We present in the 5^{th} column the basic actions having to be performed on the resources previously selected. These basic actions are elementary compared to the action types cited before, as they act on a single resource. They are described in the following:

- to show or to hide the content of a resource or a link to a resource;
- to annotate a resource or a link to a resource. This is done by adding characteristics to the concerned resource;
- to show a group of characteristics related to a particular resource;
- to play a resource - This is related to a multimedia resource;
- to update UM characteristics depending on the accessed resource.

Concerning actions based on the external environment, they describe basic actions related to the adaptive presentation behavior. They are presented in a typology in the lower half of Figure 5.7. Similarly to the available resources which are modeled in a domain (and goal) model, the knowledge related to the presentation of resources has to be modeled. Thereby, a particular model of the external environment is needed. Consequently, adaptation criteria used in the EAP framework (or GLAM) exploit classes, properties or relations included in the model of the external environment. Adaptation criteria used in LAG for adapting to the external environment are created by manipulating elements of the presentation model PM (concepts or attributes), whose specification is defined conform to the LAOS framework. Various basic actions can be performed, for example a button destination can change.

This unifying vision of both typologies allowed us to identify the main basic actions that are generally performed to build adaptation and the various selection operations that are involved. The typology shows that these systems, the EAP framework, GLAM and LAG, exploit different aspects for the same target in proposing adaptation.

2.2 Differences of adaptation modeling using the EAP framework, GLAM versus LAG

Modeling adaptation: according to the typology of Brusilovsky, we specify the types of adaptation that are supported by each approach.

- *The EAP framework*: adaptive navigation support.
- *GLAM*: adaptive navigation support, adaptive presentation.
- *LAG*: adaptive navigation support, adaptive presentation.

Methodology: it concerns the global vision of authors when authoring adaptation

- *The EAP framework*: authors focus on what is the desired adaptation to express.
- *GLAM*: authors focus on how to express the desired adaptation, even if the expression of adaptation is divided in two levels (rules and meta-rules).
- *LAG*: authors focus on how to express the desired adaptation, aided by constructors specifically aimed for that.

Modeling information: this concerns the information exploited by the adaptation.

- *The EAP framework*: adaptation considers information modeled in:
  - the domain model (the goal is modeled as a property);
  - the user model.
- *GLAM*: similarly to the EAP framework, it considers information modeled in:
  - the domain model (the goal is modeled as a property);
  - the user model.
- *LAG*: adaptation in LAG considers information modeled in:
  - the domain model;
  - the goal model;
  - the user model;
  - the presentation model.

Updating information: this concerns the models that can be updated.

- *The EAP framework*: it doesn't propose support for writing updates. The updates have to be directly expressed in the format of the used adaptation engine.

- *GLAM*: it supports updating knowledge modeled in the user model. The updates are written independently of rules and meta-rules.
- *LAG*: it supports updating knowledge modeled in the user and presentation models. The updates have to be included with the modeling of adaptation, as the user and presentation models are overlay of the adaptation model.

Reasoning with concepts

- *The EAP framework*: concepts are linked by an abstraction relation. This relation is exploited in adaptation strategies.
- *GLAM*: similarly to the EAP framework, concepts are linked by an abstraction relation, which has to be exploited explicitly by the authors when expressing adaptation.
- *LAG*: domain concepts are exploited through resources they are linked to. There is no constructor allowing to select a particular concept.

Proposing different orders between a set of resources to users

- *The EAP framework*: the order is defined using the meta-expressions expressing the ordered selection mode. Therefore, several orders may be defined between a set of resources. For example: defining definitions before exercises is possible in a first adaptation strategy for inductive students. It is also possible to define exercises before definitions in a second adaptation strategy for deductive students. Note that, it has no meaning to define definitions before exercises and exercises before definitions in the same adaptation strategy. Consequently, only one coherent order between a set of resources is defined per adaptation strategy, but the adaptive system may include several adaptation strategies.
- *GLAM*: the order is expressed using the order meta-rules (cf. Chapter 2, Section 1.1.1). Therefore, several orders may be defined between a set of resources that are selected using rules for distinct users.
- *LAG*: the order of the adaptation is defined in the goal model. Several orders can be created from one domain model, by building different goal models. However, that means that dynamic change of order of items display in a menu (based only on one goal model) at runtime is not possible even for distinct users.

Checking if the content of resources is empty or not (in order to propose alternative resources to users)

- *The EAP framework*: alternatives between resources are expressed using the alternate meta-expression.
- *GLAM*: alternative is possible in GLAM using the exclusion meta-rule. When a rule does not return any resource, another rule is executed. Therefore, when resources are missing, alternative may be proposed.
- *LAG*: LAG does not allow such direct tests. For example, let us assume that we want to define an adaptation proposing textual resources if available otherwise video resources. This cannot be expressed for the moment in LAG. Alternatives can be suggested, but if a resource is unavailable, it will just not be displayed.

3 Summary

In this chapter, we have presented a unifying vision of modeling available resources used by the EAP framework, GLAM and LAG in three levels. These three level vision of modeling the domain model includes: a generic level for supporting authoring of any AHS, a specific level, supporting authoring AHS for a particular application domain and finally an instance level, supporting authoring of a specific AHS for a particular application domain. We have expressed this unifying vision using UML, a standard widely used for modeling. In this way, we have highlighted a potential method of comparing existing domain model expressivity in any other adaptive delivery systems or adaptation authoring systems and frameworks.

On the other hand, we have also visited the expression of adaptation in the EAP framework, GLAM and LAG. This has resulted in the definition of a unified vision of a typology on adaptation. This new typology identifies the main basic actions that are generally performed to build adaptation and the various selection operations that can be done. Again, this new adaptation typology can be potentially used to evaluate adaptation expressivity in other adaptive delivery approaches, be they systems or frameworks.

CHAPTER 6

Translating generated adaptation strategies to existing adaptation languages

After studying the expressivity of the EAP framework, GLAM and LAG, we present in this chapter successively in Section 1 and in Section 2, the definition of the EAP framework on the top of the GLAM platform and on the top of the LAG language.

The work of plugging the EAP framework to GLAM has been published in the IEEE journal paper Transaction Learning Technologies [77].

We have implemented the conversion process of the EAP framework to LAG as an extension of the EAP framework. We didn't implement the conversion process of the EAP framework to GLAM, as this process is direct as it will be presented in this chapter.

1 Plugging the EAP framework to the GLAM platform

The EAP framework and GLAM aims to be independent of any application domain model. Thus they propose more freedom for authors to design their AH. Nevertheless, the EAP framework and GLAM propose different approaches: the EAP framework is a pattern-based system, while GLAM is a rule-based system.

In order to be able to plug the EAP framework to the GLAM platform (for introduction to the GLAM platform see Chapter 2, Section 1.1.1), we first consider conversion of domain and user models used by the EAP framework to domain and user models used by GLAM (cf. Section 1.1). Afterward, we translate adaptation strategies expressed using the EAP framework to adaptation in the GLAM format (cf. Section 1.2).

1.1 Conversion of domain and user models used by the EAP framework to ones used by GLAM

The EAP framework accepts domain models, composed of a set of classes, properties and relations. It proposes a clear separation between concepts and resources for better flexibility. Similarly, it accepts user models, composed of a set of classes, properties and relations. Authors have only to define the properties and their associated value. Elements composing these domain and user models correspond to same elements in the GLAM generic domain and the GLAM generic user models.

Furthermore, GLAM adaptation engine assumes that resources and user characteristics are modeled in RDF[1] format, i.e., all OWL resources (e.g. OWL properties and OWL classes) are RDF resources by nature.

1.2 Conversion of adaptation strategies from the EAP framework to GLAM

Using the EAP framework, an adaptation strategy is defined in 3 steps (cf. Chapter 4 Section 3). The resulted adaptation is described by the characteristics: *Name*, *Intent*, *Solution* and *Constituents*. Here, we focus only on the formal part of the adaptation, which is the characteristic *Solution*. It is composed of a set of expressions and of meta-expressions, as follows.

- Expressions of an adaptation select resources to be proposed to users. Thereby, each of them is based on a single criterion or a conjunction of the EAP framework criteria defined in the selection process (cf. Chapter 4 Section 4.2). To be more precise, each expression of an adaptation may include:
  - At most one expression on classes. For example: `instanceOf(resource, Definition)` proposes all instances of the class *Definition*, where the class *Definition* models resources which are *Definitions*.
  - Several or none expressions on relations. For example: `link (resource, concept`$_1$`, abstraction)` proposes resources that are linked to concept$_1$ by the relation *abstraction*.
  - Several or none expressions on properties. For example: `characteristicOf(resource, format, =, text)` proposes resources having a textual format.
- Meta-expressions of an adaptation specify how the selected resources are going to be proposed. Numerous meta-expressions can be defined, and they can be of three types (cf. Chapter 4 Section 4.2).

[1] www.w3.org/RDF/

We summarize in Table 6.1 the structure of the characteristic *Solution*.

Solution:

- Expressions
  E_i: $(\text{expression}_{classes})^{0..1} \wedge (\text{expression}_{reations})^* \wedge (\text{expression}_{properties})^*$
  where
  - $\text{expression}_{classes}$: instanceOf(resource, Class)
  - $\text{expression}_{relations}$: link(resource, concept, relation) | link(concept, concept$_1$, relation) | link(resource, resource$_1$, relation) | link-transitive(resource, concept, relation) | link-transitive(concept, concept$_1$, relation) | link-transitive(resource, resource$_1$, relation)
  - $\text{expression}_{properties}$: characteristicOf(resource, property, operation, value)
- Meta-expressions
  - $E_i \prec E_j$
  - $E_i \uplus E_j$
  - $E_i \mid E_j$

Figure 6.1 – Structure of the characteristic *Solution* of an adaptation strategy written using the EAP framework

Consequently, we proposed two conversion processes as follows.

1.2.1 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 in GLAM as follows.

- The condition part of R_i is based on E_i. Syntactic conversions are needed as the EAP framework and GLAM do not use the same syntax. We review these conversions in Table 6.1.
- The conclusion part of R_i is generated with a desirability degree set to "medium".

1.2.2 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 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 possibly higher than its previous one (resp. R_j takes a degree of desirability lower than the desirability degree of R_i and possibly lower than its previous one). 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.

Note that, all types of adaptation strategies generated using the EAP framework are convertible to the GLAM format. Therefore, they can be executed by the GLAM adaptation engine.

Expressions of the EAP framework	The condition part of a GLAM rule
Expressions on classes	
Denotes all resources that are instances of the class *Resource*. It is expressed by:	
`instanceOf(resource, Resource)`	`type(resource, Resource)`
Expressions on relations: two possible conversions	
1- Propose all elements (resources or concepts) related directly to a particular element (resource or concept).	
`Linked (element, element, relation)`	`relation(element, element)`
2- Propose all elements (resources or concepts) related directly or indirectly to a particular element (resource or concept).	
`Linked (element, element, relation)`	`relation(element, element)`
Expressions on properties	
Denotes all resources having the property and satisfying the comparison test using the operator op[2] and the value val. It is expressed by:	
`characteristicOf(resource, property, = , val)`	`property(resource, val)`

Table 6.1 – Conversions of the three types of the EAP framework expressions to GLAM

2 Plugging the EAP framework to LAG

Plugging the EAP framework to LAG is more complicated than plugging the EAP framework to the GLAM framework, since they have much more differences than the EAP framework with GLAM. In fact, GLAM expresses adaptation in a declarative way which makes it close to the EAP framework, even if GLAM is based on a rule-based system and the EAP framework is a pattern-based system. On the other hand, LAG expresses adaptation in an imperative way and using constructors.

Furthermore, we cite before that the EAP framework aims to be independent of any application domain model while LAG is specific for education applications. Also, the EAP framework makes a clear separation in designing the domain, user models and adaptation strategies, while LAG exploits domain and goal models and merges the modeling of user model and adaptation strategies. The EAP framework proposes a set of elementary adaptation patterns that can be instantiated and combined together while LAG proposes a set of constructors that can be associated together and are evaluated.

In order to be able to plug the EAP framework to LAG (for introduction to LAG see Section 2.2.2), we first consider conversion of domain model used by the EAP framework to domain and goal models used by LAG (cf. Section 2.1). Afterward, we translate user model used by framework and adaptation strategies expressed using the EAP framework to adaptation in the LAG format (cf. Section 2.2).

2.1 Conversion of domain model used by the EAP framework to domain and goal models used by LAG

The EAP framework and LAG uses two different structures of the domain model. In the EAP framework the available resources are modeled in a domain model, where LAG uses a

[2]the operator must be equal, other operators are not supported in GLAM

domain and a goal model. In order to define a conversion as generic as possible, we have studied similarities per level of modeling the domain model (and goal model) used by the EAP framework and LAG.

Thereby, in the following, we present equivalent elements of the domain model (and goal model) used by the EAP framework and LAG at each level of modeling[3]. We have started by considering first the generic level considered by the EAP framework and LAG. Afterward, we have considered the specific and the instance levels which are defined by authors. Between each two equivalent elements, we have defined a mapping.

Definition 13 *A mapping is defined between two elements of domain (or goal) models used by different approaches, if and only if these elements express the same notion.*

The defined mappings between elements of the domain model (and goal model) used by the EAP framework and LAG specify the basis of conversions between the two approaches.

Mappings between elements of the domain model used by the EAP framework and elements of the domain and goal models used by LAG

Mappings specified between these models are usable regardless of author models. We have summarized the potential mappings between the domain model used by the EAP framework and the domain and goal models used by LAG in Table 6.2.

Elements of DM used by the EAP framework	Elements of DM & GM used by LAG
The value of the *name* attribute of the Concept class	The value of the *name* attribute of the *Domain_Model* class
The *name* of the *Resource* class	The value of the *name* attribute of the *Domain Attribute* class.
The value of the content attribute of the Resource class	The value of the content attribute of the Attribute class

Table 6.2 – Mappings between elements of domain and goal models used by both approaches at the generic level

Mappings between elements of the author's domain model used by the EAP framework and elements of the domain and goal model used by LAG:

At this level, the domain and goal model structures used by LAG are more restrictive than the author's domain model used by the EAP framework, as they are specifically targeted to the domain of education. Thus the variety of relations is restricted to the concerned application domain, and the meta-data allowed is clearly divided between domain model meta-data and goal-oriented, pedagogical meta-data. On the other hand, as the domain model used by the EAP framework is not restricted to a particular domain application, we do not need to define the type of relations, but only to specify their domain and range. It is the author who specifies at the specific level the types of relations in his domain model, and his domain model may be any model composed of a set of classes, of relations and of properties. Consequently, we cannot define a whole automatic process of mappings between elements of the domain and goal model structures used by LAG and elements of the domain model used by the EAP framework. This is why, we have defined a semi-automatic process in two steps.

First it proposes a set of potential mappings to the author.

Secondly the author selects the most appropriate mappings according to his understanding of

[3]Complete description of the different levels of modeling resources used by the EAP framework and LAG is described in Chapter 5

his domain model.

In the following, we describe the set of proposals made to the author:

In the domain and goal model structures used by LAG,

- The *Link* class has two attributes, *label* and *weight*. The author has to choose to what specializations of the class Property they correspond to in the domain model used by the EAP framework.
- The *Relation* class has two instances at the specific level, in particular the *relatedness* and *part_of* relations.

We have summarized in Table 6.3 the possible mappings at the specific level that will be exploited according to the author's choices.

Elements of DM used by The EAP framework	Elements of DM & GM used by LAG
The *name* of the *Property* class	The name corresponding to the attribute *label* of the class *Link*, if this property models meta-data of string type The name corresponding to the attribute *weight* of the class *Link*, if this property models meta-data of integer type
The value of the *name* attribute of the *Property* class	The value of the attribute *label* of the class *Link*, if this property models meta-data of string type The value of the attribute *weight* of the class *Link*, if this property models meta-data of integer type

Table 6.3 – Mappings between elements of author domain model used by both approaches at the specific level

2.2 Conversion of user model and adaptation strategies from the EAP framework to LAG

As described before in Section 1.2, the resulted adaptation using the EAP framework is described by the characteristics: *Name*, *Intent*, *Solution* and *Constituents*. For our conversions here, we focus only on the formal part of an adaptation, which is the characteristic *Solution*. Converting adaptation expressions expressed using the EAP framework to LAG is limited by two major constraints:

1. The navigational path requires, additionally to the hierarchical relation between concepts in the goal model, to annotate the first (set of) concept(s) to be shown (and all its resources). This is due to the fact that, whilst the hierarchy in the goal model is prescribing in which order the concepts should be shown, it does not prescribe if they are to be shown or not. This is a separate decision and needs to be done in a separate step. Concretely, this annotation is done by using one of the two properties of concepts and of resources, either *label* or *weight* (e.g., specifying that concepts marked 'start' will be shown at the beginning). Besides, the EAP framework has not a presentation model, as presentation is an in-built effect of handling resources. In the EAP framework, concepts or resources to be proposed are specified in the adaptation and then they are calculated dynamically (i.e., they can be different from one user to another).

2. Relations between concepts in the domain (and goal) models used by LAG can be (for the moment) only of two types: *hierarchical* or *relatedness*. Consequently, only adaptations using these relations can be converted.

For example, Table 6.2 describes two expressions:

- only E1 can be converted to LAG. E1 selects all textual definitions whose concepts lead to the goal, the format property is modeled in the *label* of the resources and knowledge on the goal is modeled in the *weight* of the resources.
- E2 cannot be converted to LAG: E2 selects all textual definitions whose difficulty is high and whose concepts lead to the goal. Indeed, one of the three constraints cannot be modeled in the domain (and goal) models, either the format property of resources, the difficulty of resources, or the knowledge about the goal.

E1 : instanceOf(resource, Definition) ∧ link (resource, concept, abstraction) ∧ link-transitive (concept, goal, prerequisite) ∧ characteristicOf(resource, format, =, text)

E2 : instanceOf(resource, Definition) ∧ link (resource, concept, abstraction) ∧ link-transitive (concept, goal, prerequisite) ∧ characteristicOf(resource, format, =, text) ∧ characteristicOf(resource, difficulty, =, high)

Figure 6.2 – Examples of expressions described by the EAP framework

2.2.1 Translation of expressions to the LAG format

Table 6.4 describes conversions of the three types of expressions used by the EAP framework into LAG, by respecting the constraints defined before. We remember that, expressions in the EAP framework allow to select resources to be proposed to users.

The EAP framework	LAG (the code presented here is simplified)
Expressions on classes	
Denote all resources that are instances of the class Resource. Selecting them automatically means showing them. It is expressed by: `instanceOf(resource, Resource)`	The class of a resource is in the value of the *name* attribute of the *Attribute* class. The access to this value is done using the type constructor. `if GM.Concept.type == Resource then (PM.GM.Concept.show = true)`
Expressions on relations: 6 different conversions are considered	
1- Propose all resources related to a particular concept. `Linked (resource, concept, abstraction)`	This corresponds to showing all concepts in a goal model (specially selected for a certain goal): `while true (PM.GM.Concept.show = true)` Alternatively, if some part of the goal model is not to be shown, additional annotations of resources must be added by using *labels* or *weights*.
2 - Propose all resources whose concepts have as parent the concept$_2$. `Linked (concept, concept2, hasParent)` ∧ `Linked (resource, concept, abstraction)`	Using the `child` constructor, this returns the concepts that are children of one of the resources belonging to a given concept. `if 'concept2'.access then(PM.GM.Concept.child.show = true)`
3 - Propose all resources whose concepts have as child the concept$_2$. `linked (concept2, concept, hasParent)` ∧ `Linked (resource,concept, abstraction`	Using `parent` constructor. `if 'concept2'.access then(PM.GM.Concept.parent.show=true)`

4 - Propose all resources whose concepts have as direct or indirect parent the concept$_2$. `linked-transitive (concept, concept`$_2$`, hasParent)` $\wedge$ `Linked (resource, concept,abstraction)`	Using the `level` constructor, this returns the level of a particular resource inside the hierarchy of concepts. `if GM.Concept.level>'concept`$_2$`'.level then (PM.GM.Concept.show = true)`
5 - Propose all resources whose concepts have as direct or indirect child the concept$_2$. `linked-transitive (concept`$_2$`, concept, hasParent)` $\wedge$ `Linked (resource,concept, abstraction)`	Similarly to the previous conversion 4. `if GM.Concept.level<=âconcept`$_2$`â.level then (PM.GM.Concept.show = true`
6 - Propose all resources whose concepts are linked by a relatedness relation to concept$_2$. `linked (concept, concept`$_2$`, relatedness)` $\wedge$ `Linked (resource, concept, abstraction)`	`while true (` `'concept`$_2$`'.Relatedness.Concept.show = true)`
Expressions on properties	
Denote all resources having the property and satisfying the comparison test using the operator op and the value val. It is expressed by: `characteristicOf(resource, property, op, val)`	As a property may be expressed using the attributes *labels* or *weights*, two conversions are thus possible. Here is the conversion where a property is expressed using the attribute *label*: `if GM.Concept.label op val then (PM.GM.Concept.show = true)`

Table 6.4 – Conversions of the three types of the EAP framework expressions to LAG

2.2.2 Translation of meta-expressions to the LAG format

Table 6.5 describes conversions of the three types of meta-expressions used by the EAP framework into LAG, by respecting the constraints defined before. We remember that, meta-expressions in the EAP framework specify how the selected resources are going to be proposed to users.

The meta-expression expressing the ordered mode selection $E_i \prec E_j$, means that, the set of resources selected by E_i is first proposed. Once they have been viewed, the set of resources selected by E_j is proposed and the set selected by E_i is no more accessible. Only one set is proposed per time. In LAG, resources have to be shown and hidden explicitly, i.e. show first the set of resources selected by E_i, once they had been viewed, hide them and show the set of resources selected by E_j. Thus, the meta-expression is mapped in two strategies (cf. Row 2 in Table 6.5).

The meta-expression expressing the recommended mode selection $E_i \uplus E_j$, means that, both sets of resources selected by E_i and E_j will be proposed in the same time to the user, such as the set of resources selected by E_i is recommended rather than the one selected by E_j. A typographic indication can be used to distinguish between the resources that are recommended rather than those are not. In LAG, this is done by showing all resources in the order of their recommendation and by hiding the other ones. Thus, the meta-expression is mapped into two strategies (cf. Row 3 in Table 6.5).

The EAP framework	LAG (the code presented here is simplified)

Define an imposed order between selected resources. It is described by: $E_i \prec E_j$	It is mapped into 2 strategies, with different priorities, and these two strategies will be performed according to these priorities, each strategy containing one of the expressions
	
	Strategy 1, priority 1 :
	`if UM.GM.Concept.access == true then (UM.GM.Concept.beenthere+=1)`
	`if UM.GM.Concept.beenthere == 1 then (Ei.show = true)`
	Strategy 2, priority 2 :
	`if UM.GM.Concept.beenthere == 2 then (Ej.show = true Ei.show = false)`
	where UM.GM.Concept.beenthere is a variable that can be defined in the user model. Furthermore, the set of resources selected by E_i and E_j must have the same order in the goal model as the one defined by the meta-expression.

Define an optional order between selected resources. It is described by: $E_i \uplus E_j$.	It is mapped into 2 strategies with different priorities and they will be performed according to these priorities, each strategy containing one of the expressions:
	
	Strategy 1, priority 1 :
	`if UM.GM.Concept.access == true then (UM.GM.Concept.beenthere+=1)`
	`if UM.GM.Concept.beenthere == 1 then (Ei.show = true)`
	Strategy 2, priority 2 :
	`if UM.GM.Concept.beenthere == 2 then (Ej.show = true)`
	The set of resources selected by E_i and E_j must have the same order in the goal model as the one defined by the meta-expression.

Define alternatives between selected resources. Note that, the alternative resources are proposed only if the desired resources are not available. It is described by: $E_i \mid E_j$.	This cannot be converted as LAG does not propose any constructor to check empty resources. However, alternatives may be proposed by annotating concept using the label or weight attribute. `if (GM.Concept.label == alternative) then (PM.GM.Concept.show = true)`

Table 6.5 – Conversions of the three types of the EAP framework meta-expressions to LAG

3 Summary

This chapter focuses on the possibilities of defining the EAP framework on top of existing solutions in order to execute adaptation expressed using the EAP framework. More precisely, we have presented how the EAP framework can be defined on top of the GLAM platform (a

rule-based system) and also on top of the LAG language (a procedural language).

To conclude this part, we have promoted 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*.

So far, we have presented our EAP framework enabling authors to create their own adaptation strategies, at a high level and 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 the one hand, on a criterion to select resources and, 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.

The generated adaptation strategies are expressed at a high level and independent of any adaptation engine. In order to be executed, we have described how generated adaptation strategies will be translated to GLAM, a rule-based system, and to LAG, a procedural language. GLAM and LAG have been intentionally chosen, where each of them belongs to a distinct solution to express adaptation (either a solution based on rule languages or a solution based on procedural languages).

We have also proposed a study of the expressivity of the EAP framework, versus GLAM and versus LAG. This study has been conducted on their modeling of adaptation and also on their modeling the elements on which the adaptation is expressed. From these studies, we have come up with an integrated vision of the basic actions that can be used in defining adaptation, and also with an unifying vision of modeling the domain model.

Part V

Implementation, Experiments & Evaluations

CHAPTER 7

Implementation

In this chapter, we present the implementation of our contributions, that have been described previously in parts III and IV.

First, we have implemented the process of merging models by specialization. This process can be performed on any two OWL models, not necessary models used in the AH field (cf. Section 1).

Second, we have implemented the process of specifying adaptation strategies according to the EAP framework. We have also allowed their execution using existing adaptation engines, more particularly through the interaction with LAG (cf. Section 2). Note that, we did not implement a translator from the EAP framework to the GLAM framework, as the translation is direct (for further details on the translation process of the EAP framework to the GLAM platform see Chapter 5, Section 1).

Both implementations rely on semantic web technologies, like using OWL[1] language for modeling and SWRL[2] for expressing reasoning. Consequently, we have chosen to integrate our implementations to the Protégé tool[3] supporting such technologies. This later is considered as the most used tool by the semantic web community to create, edit, and reason on OWL models. It has been developed by a team in the Stanford university over the last 10 years in various minor and major reiterations[4]. The tool is open source. Therefore, it has motivated several developers to propose extensions. Several plug-ins have been proposed, going from visual edition of OWL models (ontoVizTab), to reasoning on OWL models using jess (Jess tab) and so on.

[1]http://www.w3.org/TR/owl-features/
[2]http://www.w3.org/Submission/SWRL/
[3]http:protege.stanford.edu/
[4]Protégé tool is available at *protege.stanford.edu/*

1 Implementation of the merging/specialization process

In part III, we have described a merging/specialization process of two models. We have implemented this process as a Protégé plug-in, called MESAM. In Section 1.1, we describe the architecture of the MESAM plug-in, which is an abbreviation of *Model mErging by Specialization of Abstract and generic Models*. Afterward, in Section 1.2, we detail the installation of the MESAM plug-in on a local machine. Finally, a general overview of its implementation is discussed in Section 1.3.

1.1 Architecture of the MESAM plug-in

As described in Figure 7.1, the plug-in includes two parts.

First, a knowledge part gathers generic models, the meta-model and deduction rules (4 rules related to patterns and 22 others are related to mappings and inconsistency). All these components are reusable across applications.

Second, the process part is made of some components performing interaction with an inference engine (in our case Jess) and the Protégé OWL editor. We have used the OWL Protégé API[5] to manipulate OWL models, as editing OWL models or generating meta-model instances from OWL models, and the SWRL Jess Bridge[6] to execute SWRL rules[7] using the Jess inference engine[8].

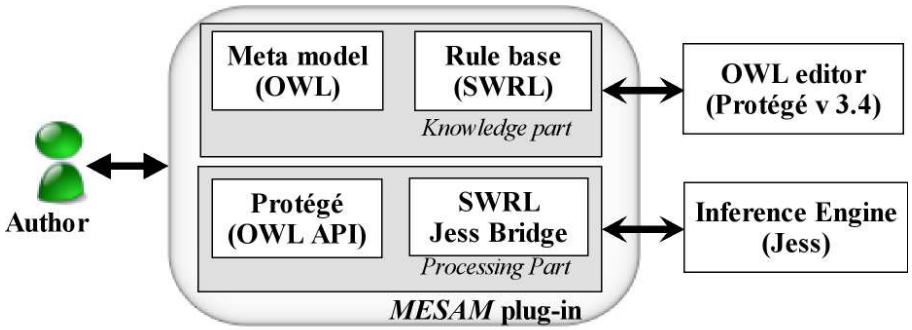

Figure 7.1 – Architecture of the MESAM plug-in

1.2 Installation of the MESAM plug-in

In order to be able to use the MESAM plug-in, Protégé v3.4 has to be downloaded from the wiki of Protégé and installed on a local machine. Furthermore, as the plug-in needs a rule engine to make inferences, it must be installed separately and configured to be used with Protégé v 3.4. At present, only the Jess rule engine is supported[9].

Once Protégé v3.4 and Jess have been installed and configured to be used together, you have to download the *fr.supelec.csd.mesam.zip* file either from the wiki of Protégé[10]or from the Wiki of Supélec[11] then to unzip it in the Protégé plug-ins directory. The plug-in MESAM is now ready to be used as any other Protégé v3.4 plug-in.

[5]http:protege.stanford.edu/plugins/owl/api/
[6]http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessBridge
[7]http://www.w3.org/Submission/SWRL/
[8]http://www.jessrules.com/
[9]Information on installing Jess is described at *http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessInstall*
[10]The MESAM plug-in is downloadable from *http://protegewiki.stanford.edu/wiki/MESAM*
[11]http://wwwdi.supelec.fr/software/cahier

1.3 Interaction with the MESAM plug-in

Here, we describe how an author can interact with the plug-in, what facilities the plug-in proposes to guide him in order to obtain a consistent merged model.

At first, the developed plug-in provides an interface allowing the author to indicate the OWL files of both the generic and specific models to be merged (cf. (1, 2) Figure 7.2). Then, it guides the author through the other steps of the process. These steps are described below.

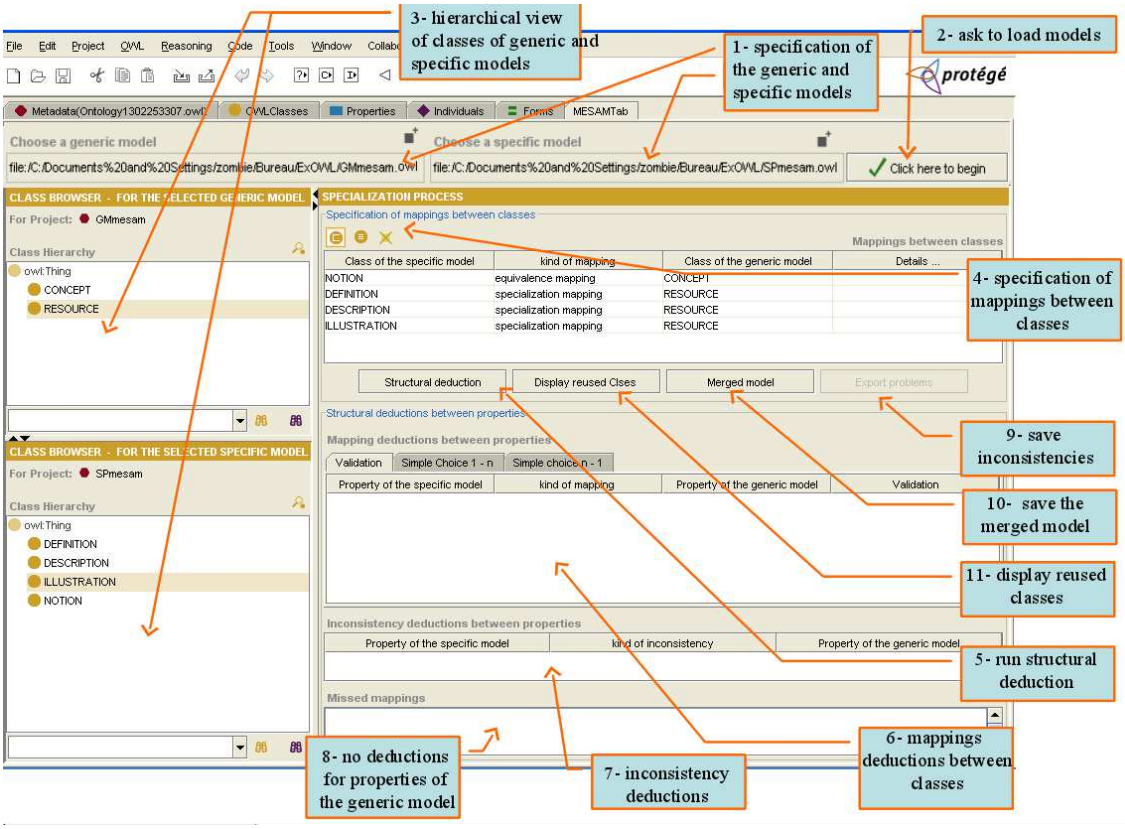

Figure 7.2 – Workspace of the MESAM interface

1.3.1 Specification of equivalence or specialization mappings

The developed plug-in proposes a vision of the reused generic and specific models in separate windows, as it can be seen in (cf. (3) Figure 7.2). The author can specify either equivalence or specialization mappings between the classes of the two models (cf. (4) Figure 7.2). Given these correspondences, additional correspondences between classes, properties or relations are generated using a reasoning module (based on deductions of Chapter III, Section 4). The hypothesis underlying this work is that it is much easier for authors to specify simple correspondences from small numbers of classes in the models, and then to evaluate mappings returned by the system. The consistency of the merged model is automatically checked.

1.3.2 Validation of structural deductions

After running the deduction process (cf. (5) Figure 7.2), the plug-in deduces mappings and inconsistency problems. The different deductions are presented in separate windows as it can be seen in (cf. (6, 7, 8) Figure 7.2). The author can confirm, choose the right one among several solutions, and can view inconsistency problems that are in the specific model to eventually

modify it outside the plug-in. Problems can be exported in a textual file (cf. (9) Figure 7.2). If no inconsistency problems are deduced, the plug-in proposes the creation of the merged model (cf. (10) Figure 7.2).

1.3.3 Printing reused classes, relations and properties

The merged model includes elements of both generic and specific models. The elements concern classes, relations and properties. It may happen that some elements of the generic model haven't been specialized or elements of the specific model haven't been reused in the merged model. This is due to the fact that the author doesn't define a mapping between each element of the specific model and one element of the generic model or a mapping between one element of the specific model and each element of the generic model.

Consequently, the plug-in proposes to the author to have a global view of the elements of the generic model that have been used or not. It also proposes a similar view for elements of the specific model. These features have been implemented by a 3rd year engineering student from Romania in current April - June 2009.

Note that, the meta-model (defined in Chapter III, Section 5 on which reasoning is made) is loaded inside the plug-in, then meta-model instances are generated from the specific and generic models. We have chosen to keep these processes invisible to the author. So, in our plug-in, the author will have the illusion to work over the specific and generic models.

2 Implementation of the EAP framework

In part IV, we have described the EAP framework enabling authors to specify adaptation at a fine granularity, independently of any adaptation engine, at a high level and in an easy manner. We have implemented this solution as a Protégé plug-in, called eapTab, which is an abbreviation of *Elementary Adaptation Patterns Tab*. In Section 2.1, we describe the architecture of the eapTab plug-in. Afterward, in Section 2.2, we detail the installation of the eapTab plug-in on a local machine. Finally, a general overview of its implementation is discussed in Section 2.3.

2.1 Architecture of the EAP plug-in

As described in Figure 7.3, the plug-in includes two parts.

First, a knowledge part gathers the library of elementary adaptation patterns and combination rules. The library is modeled in OWL[12], where each elementary adaptation pattern is an OWL class and is defined as a specialization of a class called *ElementaryAdaptationPattern*. Besides, the combination rules implement the combination process (cf. Chapter 4, Section 5.3) in a declarative way using SWRL rules[13] and the swrlx build-ins[14].

Second, the process part is made of components performing interaction with an inference engine (in our case Jess) and the Protégé OWL editor. We have used the OWL Protégé API[15] to manipulate the author's domain and user models, the library of elementary adaptation patterns and their instantiations. We have also used the SWRL Jess Bridge[16] to execute SWRL rules using Jess.

[12] www.w3.org/TR/owl-guide/
[13] www.w3.org/Submission/SWRL/
[14] The swrlx built-ins augment swrl rules with additional functionalities, e.g, creating new instances
[15] http://protege.stanford.edu/plugins/owl/api/
[16] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessBridge

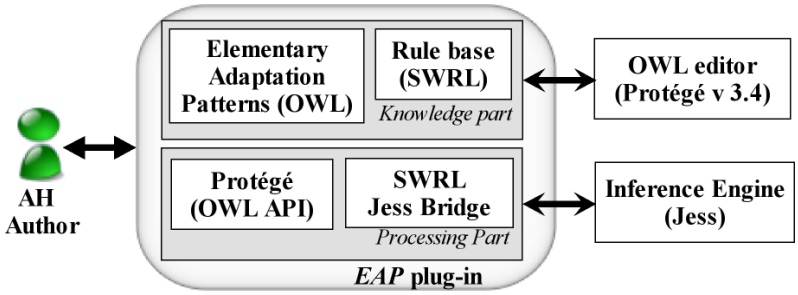

Figure 7.3 – Architecture of the eapTab plug-in

2.2 Installation of the EAP plug-in

Similarly to the installation of the MESAM plug-in 1.2, Protégé v3.4.2 (or v3.4.3) has to be downloaded and installed on a local machine. Furthermore, as the plug-in needs a rule engine to make inferences, it must be installed separately and configured to be used with Protégé v 3.4.2 (or v3.4.3). At present, only the Jess rule engine is supported[17].

Once Protégé v 3.4.2 (or v3.4.3) and Jess are installed and configured to be used together, get fr.supelec.csd.eap.zip file from the Wiki of Supélec[18] and unzip it in the Protege plugins directory (cf. Section 1).

2.3 Interaction with the EAP plug-in

Here, we describe how an author can interact with the plug-in, which facilities the plug-in proposes to guide him in order to define adaptation strategies.

First, the plug-in asks the author to load his user and domain models in OWL format (cf. (1) Figure 7.4). Then, it assists the author to define adaptation strategies according to the steps specified in the EAP framework (cf. Chapter 4, Section 3). We describe below how these steps are performed using the eapTab plug-in.

2.3.1 Definition of elementary adaptations

The plug-in proposes authors to add elementary adaptations one by one (cf. (3) Figure 7.4). For each new elementary adaptation, the author selects the appropriate elementary adaptation patterns by selecting the criteria on which it is expressed. For example: by selecting *ordered selection* mode, and the element of the domain model *classes*, the plug-in automatically selects the pattern P2.2 (cf. Appendix B, Table B.2).

Once the elementary adaptation pattern has been selected (cf. (1) Figure 7.5), the author may access its description (cf. (2) Figure 7.5), specify the name of the elementary adaptation, its intent and the elements of the author domain model on which the elementary adaptation will be expressed. Afterward, he may add this new elementary adaptation (cf. (3) Figure 7.5) to the list of elementary adaptations (cf. (4) Figure 7.4).

2.3.2 Association of elementary adaptations with user characteristics

After defining each elementary adaptation, the author may associate it to a user characteristic or possibly wait till he has defined all needed elementary adaptations. The plug-in integrates

[17]Information on installing Jess is described at *http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessInstall*
[18]http://wwwdi.supelec.fr/software/cahier

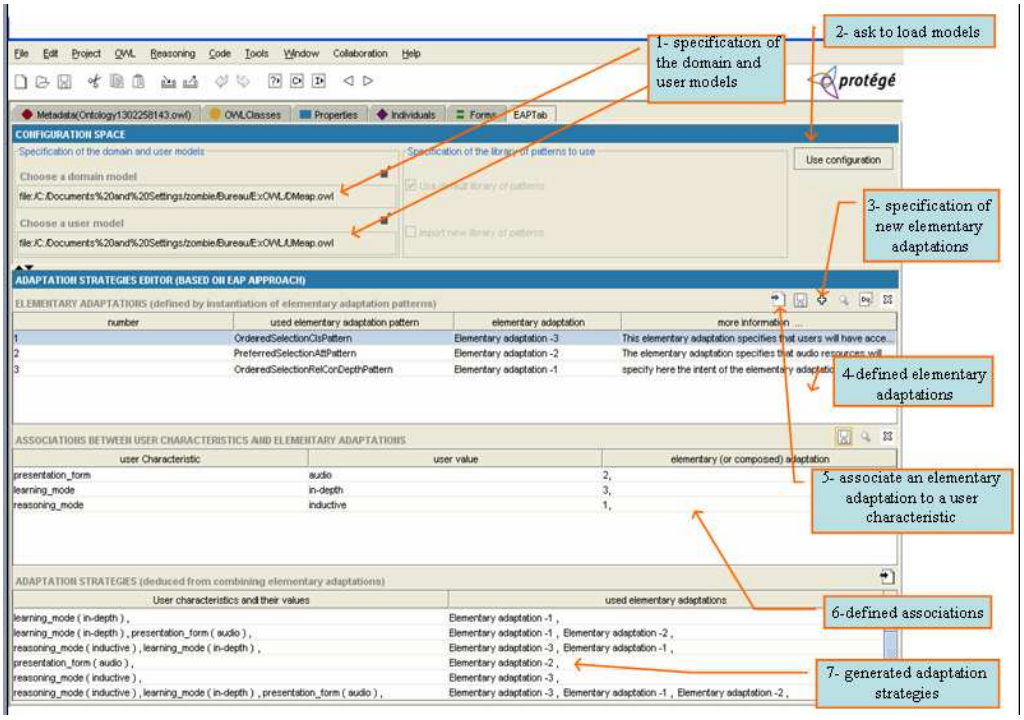

Figure 7.4 – Workspace of the eapTab plug-in

a button (cf. (5) Figure 7.4) allowing the author to associate the selected elementary adaptation from the list of defined elementary adaptations with one of the available user characteristics.

2.3.3 Definition of adaptation strategies

The plug-in also assists the author in defining adaptation strategies. Similarly to performing the previous steps, the plug-in proposes a button to generate adaptation strategies. These adaptation strategies are based on the combination of values of user characteristics that a user may have at a given instant.

Therefore, the plug-in builds several adaptation strategies, each of them corresponds to a composition of user characteristics that a user may have (cf. (7) Figure 7.4). $S1$ is the last line in the list of generated adaptation strategies.

2.4 Plugging the EAP framework to LAG

We implemented a prototype of conversion of adaptations described using the EAP framework to LAG. This prototype has been implemented in JAVA and has been integrated to eapPTab.

Once the author has defined his adaptations using eapTab, the plug-in proposes exporting the adaptation to LAG. This conversion process takes as input the following elements:

- the domain, user models expressed in OWL;
- adaptation strategies described in OWL having the same structure than the elementary adaptation patterns.

It proposes as output the following elements.

- The domain and goal models in the CAF XML-based format.

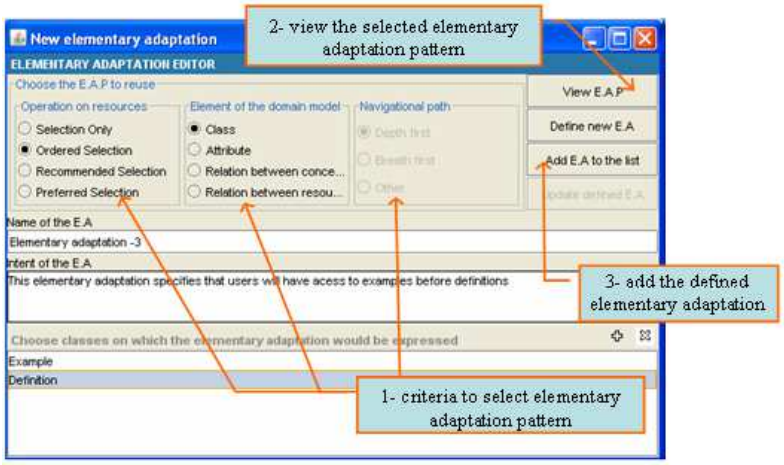

Figure 7.5 – Workspace of the elementary adaptation editor

- The adaptation, user and presentation models described using the LAG adaptation language. Note that this decision to merge adaptation model, user model and presentation model is a simplification from the LAOS framework, which defines them separately. Nevertheless, the LAG language allows for clear distinction of parameters coming from each of the three models (for example, all user model variables start with UM, all presentation model variables start with PM, etc.), so a clear differentiation is possible.

The process of conversion is based on the studies presented in Chapter 5 and is the following:

Conversion of the domain model The domain model used by the EAP framework is converted to a domain and goal model used by LAG according to the process described in Chapter 5, Section 2.1.
As the EAP framework is independent of any application domain and as LAG is only for e-learning applications, the domain model used by the EAP framework may have more expressivity than the domain (and goal) models used by LAG. In such cases, the domain model used by the EAP framework cannot be converted to domain (and goal) models used by LAG. In the following, we cite the cases where such conversions cannot be performed.

- The domain model used by the EAP framework includes a typology of resources, and more than two properties. In this case, the typology of resources is expressed in the *name* attribute of the *ATTRIBUTE* class (cf. Chapter 5, Figure 5.3), one property is expressed using the *label* attribute of the *LINK* class (cf. Chapter 5, Figure 5.3) and the other one property is expressed using the *weight* attribute of the *LINK* class (cf. Chapter 5, Figure 5.3). However, the additional properties expressed in the domain model used by the EAP framework cannot be converted. Therefore, in such case there is no way to convert the domain model and the author will be notified about this.
- The domain model used by the EAP framework includes more than two relations between concepts. In this case, only two of them can be converted (*hierarchical* and *relatedness*). The other relations cannot be converted. Therefore, the author will be informed about this and the domain model will not be converted.

Conversion of the user model The user model used by the EAP framework is not converted to a separate user model used by LAG. As adaptation in LAG is defined, for simplification and compactness, within the adaptation specification, when authors initialize and manipulate (specify updates for) user attributes, authors don't need to specify a user model separately

from the adaptation specification. For that, user characteristics defined in the user model used by the EAP framework are converted to user attributes in the generated adaptation in the LAG format.

Conversion of adaptation Adaptations described using the EAP framework are converted according to the process and the constraints described in Chapter 5, Section 2.2.

- First, expressions included in adaptations to be converted are considered. The expressions on classes and on properties are directly converted to LAG. However, expressions on relations need additionally to annotate resources in the goal model by interpreting either their *labels* or their *weights*. For example, for an adaptation to be converted specifying a navigational path on the concept graph, we have to annotate the first concept to be proposed inside the goal model used by LAG. This annotation is done using the property *label* of goal concepts, if it is not used for characterizing other pedagogical characteristics; otherwise, the annotation is done using the property *weight* of the goal concepts.
- Second, meta-expressions included in adaptations to be converted are considered. The priority and recommendation meta-expressions are converted to LAG. Furthermore, we have to organize the resources within the goal model using the same order expressed by the converted meta-expressions. This is done using the property order of the *Link* class in the goal model that will be used by LAG. Consequently, the goal model is modified during the process of converting adaptation. The alternate meta-expression is not converted to LAG (cf. Chapter 5, Section 2.2).

These conversions of adaptations are based on the conversion on the elementary adaptation patterns[19]. We have thus proposed a conversion process for the elementary adaptation patterns defined in the EAP framework. We have converted the elementary adaptation patterns (cf. Appendices B) except the elementary adaptation patterns using the alternate selection mode P4.1.1.1, P4.1.1.2, P4.1.2.1, P4.1.2.2, P4.2 and P4.3 which haven't been converted to LAG as this later does not support to check whether a resource is empty or not. The elementary adaptation patterns expressed on concepts directly (P1.1.1, P2.1.1.1, P2.1.1.2, P3.1.1.1 and P3.1.1.2) cannot also be converted to LAG as this later does not propose constructors to access concepts but only resources. These conversions are expressed in AppendixC

3 Summary

So far, we have presented the implementation of our contributions. These implementations propose a support for defining merging models by specialization and for defining adaptation strategies.

In the following, we present experiments and evaluations of our contributions that have been conducted thanks to the development made.

[19] An adaptation expressed using the EAP framework is defined by the instantiation of one or more elementary adaptation patterns and the combination of these instantiations.

CHAPTER 8

Experiments and evaluations in e-learning

In this chapter, we present experiments and evaluations that have been conducted using MESAM tab and EAP tab in the e-learning domain.

Section 1 presents our experiments followed using MESAM tab on models included in John's use case (cf. Chapter 3, Section 2.4).

Section 2 presents our evaluations using the eapTab to determine whether it is easier and faster to express adaptation strategies using EAP framework versus the GLAM platform in a first time, and using LAG in a second time. These evaluations have been done with volunteers on Jane's use case (cf. Chapter 4, Section 2).

1 Experiments of the MESAM plug-in in the adaptive e-learning hypermedia domain

In this experiment, our aim is to show how the MESAM plug-in could be useful for authors of AH. For this purpose, we have executed MESAM plug-in on models included in John's use case (cf. Chapter 3, Section 2.4).

1.1 Experimental settings

John's use case (cf. Chapter 3, Section 2.4) includes a generic model with 2 classes and 7 relations and a specific model with 4 classes and 17 relations.

To perform the merging process, the MESAM plug-in relates on the author to start the process by defining mappings between classes. We have played the role of an author. Therefore, we have defined mappings between classes. More precisely, we have defined 7 different situations[1], each situation considers different mappings between classes (cf. line 2 in Table 8.1). Afterward, we have run the plug-in on these situations and have summarized the obtained results in line 3 and line 5 in Table 8.1. We also compare the total number of deduced mappings versus the correct mappings that have to be defined (cf. line 4 in Table 8.1).

1.2 Obtained results

Table 8.1 presents results of execution of the MESAM plug-in according to particular situations.

Situations	sit 1	sit 2	sit 3	sit 4	sit 5	sit 6	sit 7
Number of mappings	4	4	3	3	4	2	1
Correct mappings that can be defined							
- CONCEPT and NOTION	√	√		√		√	√
- RESOURCE and ILLUSTRATION	√		√	√		√	
- RESOURCE and DESCRIPTION	√	√	√	√			
Wrong mappings that can be defined							
- RESOURCE and DEFINITION	√	√	√				
- CONCEPT and DEFINITION					√		
- CONCEPT and ILLUSTRATION		√			√		
- CONCEPT and DESCRIPTION					√		
- RESOURCE and NOTION			√		√		
Deductions proposed to the author							
- for validation	10	7	5	7	2	4	3
- for choice between 2 mappings	6 * 2	4 * 2	4	5 * 2	2 * 2	3	0
- about inconsistency	0	0	4	0	6	0	0
Total of proposed mappings	22	15	13	17	12	10	3
Correct mappings that should be proposed	16	11	6	12	0	7	2
Missing mappings	1	3	4	0	5		1
Built merged model	X	X	X	√	X	√	X

Table 8.1 – Execution of MESAM plug-in according to several situations on models of John's use case

[1]Note that, we may have 80 situations, as there are 4 classes in the specific model and each of them may be in three cases (linked to the CONCEPT or RESOURCE class or not linked). We don't consider the case where no class of the specific model is related to a class of the generic model, as in this case the process does not make deductions.

The fourth situation (column *sit 4* in Table 8.1) corresponds to the john's use case (cf. Chapter 3). It assumes that John only defines correct mappings, between the classes *CONCEPT* and *NOTION*, *RESOURCE* and *ILLUSTRATION*, *RESOURCE* and *DESCRIPTION*. The plug-in deduces 7 mappings that have to be validated by the author. This includes mappings between:

(V1) *name_of_concept* and *title*;

(V2) *abstraction* having as domain the class *CONCEPT*, as range the class *RESOURCE* and *abstraction* having as domain the class *NOTION*, as range the class *ILLUSTRATION*;

(V3) *abstraction* having as domain the class *RESOURCE*, as range the class *CONCEPT* and *abstraction* having as domain the class *ILLUSTRATION*, as range the class *NOTION*;

(V4) *name_of_resource* and *illustration_identifier*;

(V5) *abstraction* having as domain the class *CONCEPT*, as range the class *RESOURCE* and *abstraction* having as domain the class *NOTION*, as range the class *DESCRIPTION*;

(V6) *abstraction* having as domain the class *RESOURCE*, as range the class *CONCEPT* and *abstraction* having as domain the class *DESCRIPTION*, as range the class *NOTION*.

(V7) *name_of_resource* and *description_identifier*;

The plug-in also deduces 5 cases, where John has to choose between two mappings. He can choose no, one or two mapping (s) in each case.

C1 *content_of_resource* and *illustration_content*/*content_of_resource* and *presentation_of_illustration*.

C2 *format* and *illustration_content*/*format* and *presentation_of_illustration*.

C3 *content_of_resource* and *description_content*/*content_of_resource* and *description_format*.

C4 *format* and *description_content*/*format* and *description_format*.

C5 *pre-requisite* and *successor*/*pre-requisite* and *part-of*.

At all the plug-in proposes 17 mappings where the merged model will include 12 mappings. We argue that it remains manageable by the author, as 7 mappings are almost sure and there are only 5 cases where the author has to choose between 2 mappings.

Note that, the plug-in would have return better results if relations in both specific and generic models have been better defined, for example, *format*, *presentation_of_illustration* and *description_format* have been defined with a distinct types of *content_of_resource*, *illustration_content* and *description_content*. In this case, the author wiould have 11 validations to make and only one case with choice between two deductions. Conversely, the plug-in returns more choices if relations in both specific and generic models have multiplicities equals to *. This is due to the fact that, the plug-in exploits structural knowledge to deduce mappings and lets the semantic to be validated by the author himself.

Consequently, the results provided by the plug-in do not only depend on the initial set of mappings defined by the author between classes of the two models but also on the constraints defined on relations of the models themselves.

The fifth situation (column *sit 5* in Table 8.1) corresponds to the situation where the author only defines wrong mappings. He specifies mappings between the classes *CONCEPT* and *ILLUSTRATION*, *CONCEPT* and *DESCRIPTION*. *CONCEPT* and *DEFINITION* and *RESOURCE* and *NOTION*.

The plug-in deduces 2 mappings that have to be validated by the author. This includes mappings between:

(V1) *name_of_concept* and *definition_identifier*;

(V2) *format_of_resource* and *title*;

(V3) *content_of_resource* and *title*.

The plug-in also deduces 2 cases, where John has to choose between two mappings. He can choose no, one or two mapping (s) in each case.

C1 *name_of_concept* and *illustration_content/name_of_concept* and *presentation_of_illustration*.

C2 *name_of_concept* and *description_content/name_of_concept* and *description_format*.

The plug-in deduces 6 inconsistency mappings and informs the author about 5 relations of the generic model that have no probable mappings. We notice that in such a situation (i.e., with only wrong mappings between classes) the plug-in deduces more inconsistency and missing mappings.

From these executions, we note that more there are wrong mappings between classes, more the plug-in doesn't find structural mappings between relations of specific and generic models. Therefore, more missing mappings between relations are detected.

2 Evaluation of the EAP tab versus existing Adaptive Systems

We have conducted two successive evaluations: evaluation of the EAP tab versus GLAM (cf. Section 2.1), and evaluation of the EAP tab versus LAG (cf. Section 2.2).

2.1 Evaluation of the EAP framework versus GLAM, a 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 versus using GLAM.

2.1.1 Evaluation settings

We asked course teachers[2] from Supélec[3] and INRIA[4] to define adaptation strategies according to Jane's use case (cf. Chapter 4, Section 2).

As in our experiments, we focus on the expression of adaptation, we expressed the domain and user models according to each solution. We performed the evaluation on each volunteer separately (mainly due to their different availabilities). Seven volunteers[5] performed the evaluation. They were between 20 and 30 years old, with between 1 and 7 years experience in higher education. Among the volunteers, there were 6 men and 1 woman and all of them were new to both solutions.

2.1.2 Obtained results

We divided the sequence of experiments into two steps *before* and *after* specifying adaptation strategies. We detail them below.

Before specifying adaptation strategies.

[2]working on ICT (Information Communcation & Technology): wireless network, energy or computer science
[3]www.supelec.fr/
[4]www.inria.fr/saclay/recherche/
[5]In [50], the author proved that with at least 5 persons we can obtain significant results.

We argue that some skills may introduce a biais in our experiment. For example: people often manipulating rules, will easily and quickly express adaptation using rules. To avoid this, we defined a questionnaire estimating skills related to our experiments, and the volunteers were asked to fill it in. The questionnaire included 16 questions, which we gathered in 5 groups: principles and implementation of AH, principals and modeling of personalization, use of rule-based languages, use and implementation of design patterns and courses building. The volunteers had to estimate their knowledge between: none, little, intermediate, good or very good.

Figure 8.1 presents skills of the volunteers in a graph with two axes. In the abscissa, we present the groups of questions and in the column, 10% presents one volunteer. For AH skills, 3 volunteers had no background, 3 others only knew the principles and 1 estimated that he had intermediate knowledge. Volunteers had better understanding of personalization: 4 volunteers knew about the principles, whilst 3 others had intermediate understanding. Concerning rule-based languages, 5 volunteers had little understanding, 1 volunteer had already used such language and another one estimated that he had good skills. For design patterns' principles, 3 volunteers admitted having no knowledge at all, 1 volunteer only knew the principles and had never used a design pattern, 1 volunteer estimated he had an intermediate understanding and 2 other volunteers said that they knew them well. For courses building, 3 volunteers had intermediate notions as they had just start teaching and 3 others estimated they had good skills and 1 volunteer estimated he had very good skills.

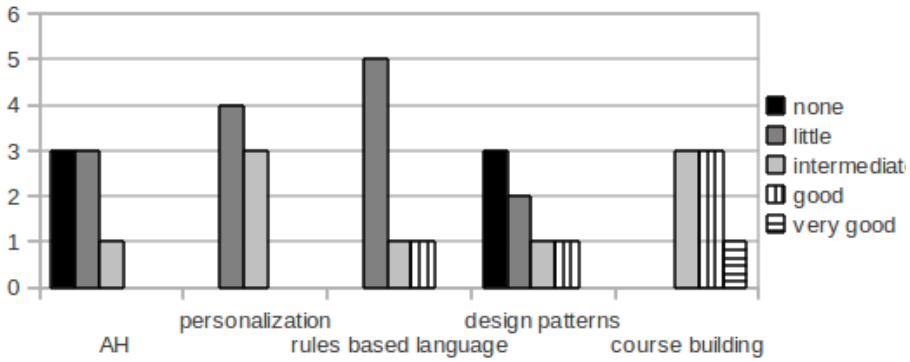

Figure 8.1 – Skills of our volunteers

Thus, none of them had any knowledge on the implementation of an AH, particularly on writing adaptation. They knew more or less what we mean by personalization. Consequently, we first explain general knowledge of AH. We then trained the volunteers using examples to introduce them to both solutions. Finally, they had to define at least $S1$ (cf. Chapter 4, Section 2), first according to the EAP framework using eapTab, secondly according to GLAM using a notepad. Starting with one system does not influence the results as they didn't know which one was our solution.

After specifying adaptation strategies.

Once the volunteers had specified adaptation using both approaches, they were asked to evaluate the difficulty of expressing adaptation strategies using each solution. They had to choose between (very easy, easy, intermediate, difficult and very difficult). We present their answer in Figure 8.2 using a graph with two axes. In the abscissa, we present the answers related to GLAM and eapTab and in the column 10% presents one volunteer. When using GLAM, 4 volunteers (i.e, more than half of the volunteers) found that it was hard, mainly to express navigational paths and to specify the correct meta-rules. Using eapTab, 3 volunteers estimated

that it was simple and 1 volunteer estimated that it was very simple. Therefore, more than half of our volunteers estimated that it was easy to express adaptation strategies using eapTab. Furthermore, the approach of breaking down an adaptation into multiple elementary ones seemed to be pleasant and intuitive for them, as they never think about how they should write their adaptation strategy, but only what they wanted to express. Only ergonomic requirements were given in order to improve the usability of eapTab.

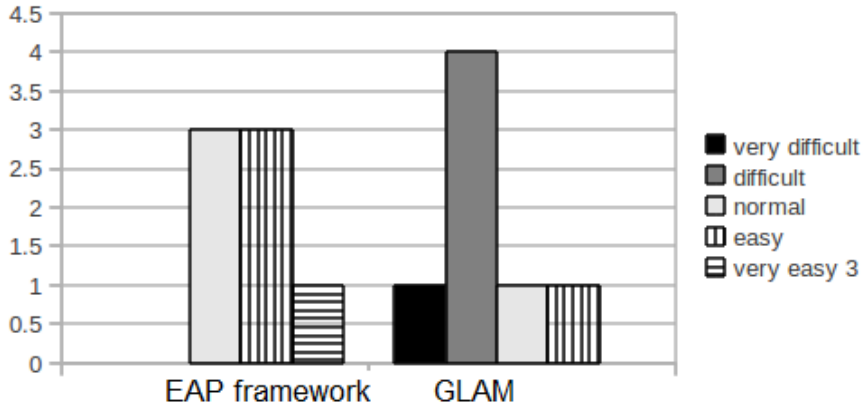

Figure 8.2 – Estimation of difficulty to express $S1$

Additionally to this, we measured the time spent to express adaptation strategies using each approach. We grouped our estimation in 4 intervals (less than 10 min, between 10 and 15, between 15 and 20, more than 20 min). We present these estimations in Figure 8.3 with two axes. In the abscissa, we present the time spent using GLAM or eapTab and in the column 10% presents one volunteer. We noticed that most volunteers were able to create similar adaptation strategies using eapTab within approximately half the time than when using GLAM. Furthermore, they defined only $S1$ when they used GLAM, while they did not hesitate to define others using eapTab. We explain these results by the fact that when using GLAM, authors have to manually find all the conditions that must be satisfied by the proposed resources and manually compose the different conditions. These conditions must be written as rules in GLAM. Then, they have to define which rules are to be applied to which user by writing meta-rules. When using eapTab, volunteers only have to define the elementary adaptations, then to associate them to user characteristics and the combination process is done automatically. They have to trust the solution for the combination process.

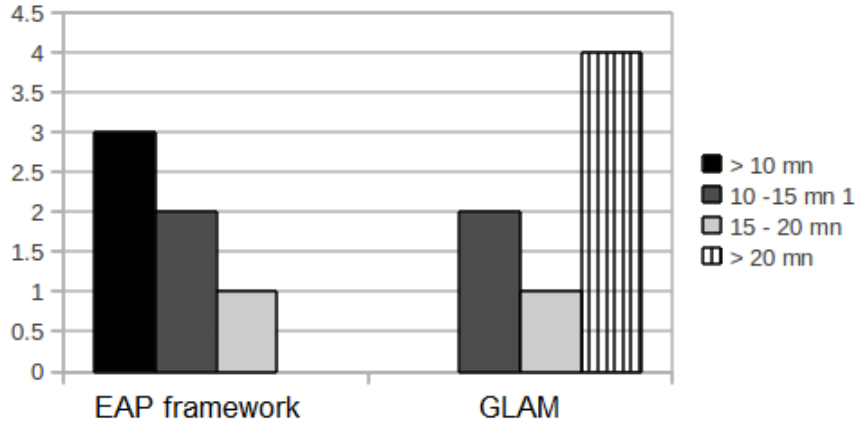

Figure 8.3 – Estimation of time spent to express $S1$

Note that, $S1$ is defined using 8 rules and 6 meta-rules in GLAM (cf. Chapter 4, Figure 4.6), where GLAM rules include repetitive parts, e.g, the selection of definitions is present in 4 rules. Only 3 elementary adaptations (cf. Chapter 4, Section 5.1) are required using the EAP framework.

2.2 Evaluation of the EAP framework versus LAG, a generic adaptation language

Similarly to the previous evaluation, we have carried-out an experiment with real users to see if it is *easier* and *faster* to express adaptation strategies using LAG rather than using the EAP framework or vice versa.

2.2.1 Evaluation settings

We asked assistant professors from Supélec and INRIA to define adaptation strategies following the scenario described in section 3 using LAG and using the EAP framework. The scenario includes (cf. Chapter 4, Section 2):

A domain model (abbreviated DM) with three concepts and a few resources having different types (definition, example, etc) and formats (text, image, etc). This choice was motivated by the fact that we asked assistant professors to book 2 (at most 3) hours for the evaluation. We had also to consider that some of these volunteers are not computer scientists. Also, most of the volunteers do not have background about adaptive hypermedia or authoring adaptation strategies, i.e., they will need more explanation about the context of the evaluation. Consequently, we have included few resources and concepts for illustration purpose. Furthermore, they will not express their adaptation strategies directly on resources but on the domain model structuring the used resources. Even if the domain model is small, it is enough complex. In fact, it includes enough elements to define a large number of different adaptation strategies.

A user model with three user characteristics.

Two adaptation strategies where each of them proposes a set of resources for learners with specific characteristics ($S1$ and $S2$ in Chapter 4, Section 2).

In our experiment, we focus on the expression of adaptation. For that, we have built for the volunteers the domain, goal, or user models, according to each approach. Thus, for expressing adaptation using LAG, we have built domain and goal models in CAF, but we have not built a user model, as the LAG language requires specification of the user model in the adaptation specification. For expressing adaptation using the EAP framework, we have built domain and user models in OWL that can be used by the EAP framework.

Thereby, the volunteers have only to express adaptation strategies using both approaches. We have performed the evaluation for each volunteer separately (this is mainly due to their different availabilities). Eight volunteers [50] have performed the evaluation. They have between 25 and 40 years old, and between 1 to 15 years experience in higher education. Among the eight volunteers, there were 3 women and 5 men. Two of the volunteers have participated in a previous evaluation with the EAP framework. Therefore, we estimate they knew more or less the eaptab [75]. The other 6 volunteers were new to both systems.

2.2.2 Obtained results

We have divided the sequence of the experiments in two steps: before creating adaptation strategies and after creating adaptation strategies. In each step, volunteers have to answer one

questionnaire. We describe each step further in the following.

Step 1: before creating adaptation strategies

The volunteers were asked to fill in a questionnaire about their skills before starting the evaluation. We argue that volunteers' skills may affect the evaluation[6]. For example, people having good skills in AHS can easily express their adaptation strategies. This is why we have defined a targeted questionnaire about specific skills involved when using LAG or the EAP framework. The volunteers were asked to evaluate their skills on object modeling, on AHS, on personalization, on programming languages and on building courses. They had to answer each question by choosing one of the following values: none, little, intermediate, good or very good. We present in Figure 8.4 the skills of the volunteers participating in our experiments using a graph with two axes. The graph includes in the abscissa the different questions of the questionnaire with their possible answers. A distinct gray scale is used for each possible answer. We notice that most of our volunteers have good skills in programming languages, skills in building courses and knowledge on object modeling. However, they have limited skills on AHS, and intermediate understanding of personalization in general.

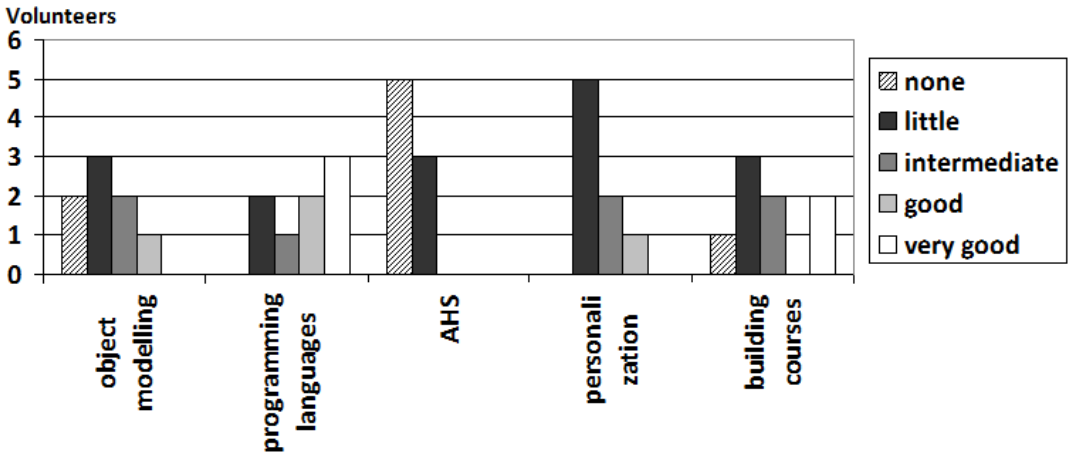

Figure 8.4 – Skills of volunteers participating in our experiments

Consequently, we started first by explaining to volunteers general notions on adaptive hypermedia systems, illustrated by examples in the e-learning domain and describing their utilities for learners. This has given to them a user's point of view of AHS. After that, we have explained to them the role of an author for AHS, which is more technical, and we let the volunteers manipulate the authoring tools (those used in this evaluation: the PEAL tool for the LAG adaptation language, and the eapTab plug-in for the EAP framework) for 10 minutes first, to get somewhat used with our tools. Finally, we have defined domain and goal models and have asked volunteers to express adaptation strategies using both approaches and to complete another questionnaire, in order to evaluate the ease of use of each approach. We have also measured the time spent to express adaptation strategies using each approach.

Note that we did not impose to volunteers to start with either LAG or the EAP framework. They have to choose which approach they want to consider first, and they were not influenced by one or the other approach before the test. We have told them that writing adaptation using LAG is like writing a code using a programming language, whilst writing adaptation using the EAP framework is based on design patterns vision. From the 8 volunteers, 5 volunteers have started with the EAP framework, and the 3 others have started with LAG.

[6]Note that, skills needed in this evaluation are different from the ones needed in the evaluation of the EAP framework and GLAM. Therefore, we have defined specific questionnaire to each evaluation.

Step 2: after creating adaptation strategies

Once volunteers have created adaptation strategies with both approaches, they had to complete a questionnaire in order to evaluate the ease of use of each approach. The questionnaire concerns questions about the difficulty of expressing an adaptation strategy, understanding and reusing of existing adaptation strategies. We have asked volunteers to answer each question by choosing one of the following values: very easy, easy, intermediate, difficult or very difficult. In Figure 8.5 we summarize answers collected from volunteers through a graph with two axes. At the abscissa, there are the different questions of the questionnaire with their possible answers. A distinct grey scale is used for each possible answer. The column represents the numbers of people.

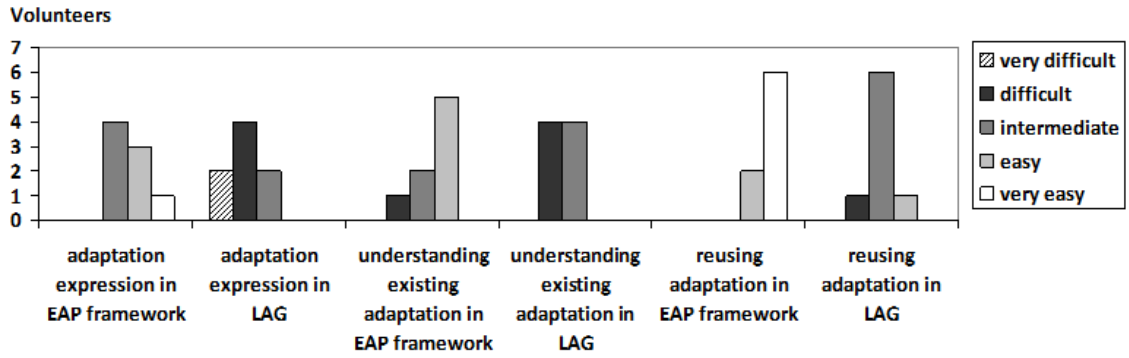

Figure 8.5 – Estimation of difficulty of expressing an adaptation strategy, understanding and reusing of existing one in LAG and in the EAP framework.

Concerning the difficulty of expressing an adaptation strategy, our volunteers' opinions were divided. With the EAP framework, 4 of them think that it does not require lot of effort, 3 others think that it is easy and one of them finds it is very easy to express adaptation strategies. It seems for them very natural to express an adaptation strategy as a combination of elementary units. While with LAG, 2 of them think that it does not require a lot of efforts, 4 others think that it is hard to express adaptation strategy. One volunteer has refused to express the required adaptation strategy in LAG. This is mainly due to the fact that with LAG they had to write adaptation strategies using a grammar which was unfamiliar to them, and they lacked training with.

Concerning the understanding of existing adaptation strategies, most of the volunteers found that with the EAP framework, it is relatively easy to understand the aim of an adaptation strategy. We explain this because in the EAP framework an adaptation strategy is a combination of elementary units and each elementary unit is based on defined elements of the domain model. With LAG, it needs more effort to understand an existing adaptation strategy, as a LAG strategy may includes variables and tests on the used variables.

Concerning the reuse of existing adaptation strategies, all our volunteers found that with the EAP framework, it is easy to reuse existing ones. This is mainly because, once they have defined elementary adaptations, they can reuse them to compose other adaptation strategies. Furthermore, once they have expressed an elementary adaptation using a particular pattern, they find very easy to express another elementary adaptation using the same pattern (on other elements of the domain model). With LAG, it does not require a lot of effort to reuse existing ones. Once they have understood the desired adaptation strategy, they can adapt it according to their needs. Here, they find the approach proposed by the EAP framework easier than the one proposed by LAG, even if the one proposed by LAG is not difficult to perform.

Some volunteers have noticed that with the EAP framework. They do not write any instructions, they have only to select, to instantiate elementary adaptation patterns and to make

associations between user characteristics and the defined instantiations, and the adaptation strategy is generated automatically for them. In other words, they have only to click on the right button and to select the right elements. Thereby, authors using the EAP framework have to trust the eapTab in the combination process. When using LAG, volunteers have to write themselves the adaptation strategy using LAG constructors and respecting the proposed grammar. Even if they are good programmers they estimate they need more effort and time, in order to get used to the LAG syntax and to be sure of the defined adaptation strategy. Consequently, they have estimated that we are evaluating two distinct processes.

Additionally to the questionnaires above, we have measured the time spent to express adaptation strategies using each approach. We present in Figure 8.6 the estimation of the time spent to express adaptation strategies using each approach with a graph with two axes. We have grouped the time spent by the volunteers in four time intervals: less than 5 minutes, between 5 and 10 minutes, between 10 and 15 minutes, more than 15 minutes. In the graph, at the abscissa, the two used approaches are represented. For each approach, there are the four defined time intervals, a distinct gray scale is used for each time interval. On the column, the numbers of people having expressed adaptation strategies are represented. We notice that, when using the EAP framework, most of volunteers have spent between 5 minutes to 10 minutes to complete the scenario. This is due to the fact that they do not have to write instructions. They focus more on what adaptation they require and for whom (what type of users) it is proposed. Some of the volunteers prefer to propose the alternate selection mode rather than selection of a particular type of resource only. When using LAG, most of volunteers have spent between 10 minutes to 15 minutes to complete the scenario. One of the volunteers has even refused to express any adaptation strategy, and thereby he was not included in the graphic. As volunteers have not had enough time to learn about LAG, they were asking for assistance.

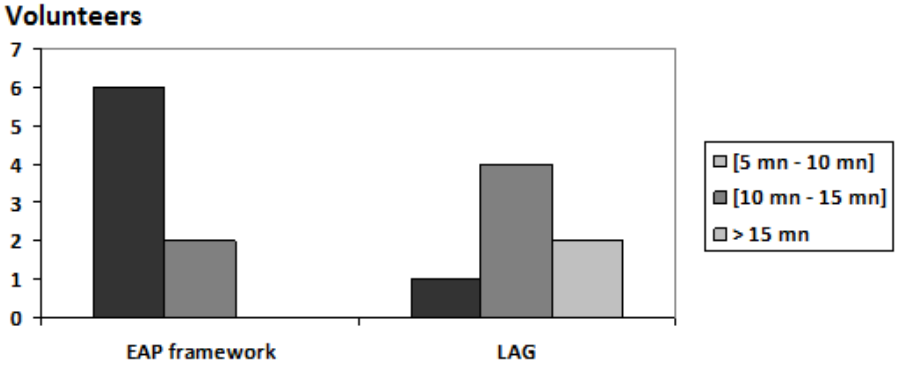

Figure 8.6 – Estimation of time spent by volunteers to perform the evaluation using LAG and using EAP framework.

Additionally, we have asked the volunteers to comment on the two tools they were using to express the two languages. Volunteers commented that they find the eapTab interface very basic and added that it needs improvement especially on guidance for becoming a realistic interface. On the contrary to their eapTab experience, volunteers have noted that they appreciated playing with the PEAL tool when expressing adaptation in LAG. They said that the PEAL tool is ergonomic. It proposes several helps to assist them, for example, LAG constructors are written in distinct colors according to their purpose, when their LAG code includes syntax errors, PEAL detects and highlights the errors.

Summary of the evaluation LAG versus the EAP framework

The evaluation was conducted with only 8 volunteers. Even if they are enough to make some conclusions, in our case, we cannot draw a conclusion on whether it is more easy to use the EAP

framework than LAG to express adaptation and conversely. However, we can say that expressing adaptation using the EAP framework is much faster than expressing adaptation using LAG. Note that some of the volunteers were not from the computer science field. Thereby they did not have technical background and they prefer using the EAP framework to express their adaptation.

Besides, most of the volunteers have identified that we had evaluated not only two different adaptation languages, but two distinct approaches, where one proposes an automatic process to combine several constraints, that means that, authors have to trust the combination process. The other approach lets the author to combine himself the defined constraints to select and to define how the selected resources are going to be proposed. Thereby the first approach was more appreciated than the second, as the latter needs more focus and thus requires more effort to check if their expressed adaptation proposes all the desired resources in the wanted way. The volunteers said that if they have to choose, they will surely base their decision on the expressivity of each approach. They did not evaluate the expressivity of any approach, they have only told us that if they had to choose between both for defining complex adaptation strategies, they will trust the combination process.

3 Summary

In this chapter, we have described experiments using the MESAM plug-in showing the helps that can be provided to authors in several situations, when they define correct mappings between classes, or when define wrong mappings between classes.

We have also described evaluations of the EAP framework versus GLAM and the EAP framework versus LAG. In the first evaluation, volunteers have declared that it is much easier to express adaptation using the EAP framework than using GLAM, they have also spent less time using the EAP framework than GLAM. In the second evaluation, volunteers were divided. They estimate that the EAP framework is much easier for people without computer science knowledge, but LAG could be useful for people having good programing skills. In this later, people will have more control on the combination process of several adaptations as it is manual. For remember LAG is a generic adaptation language, aiming at facilitating expressing adaptation using constructors.

Part VI

Conclusion and future work

CHAPTER 9

Conclusion and future work

1 Conclusion

In this thesis, we have focused on providing assistance to authors when designing their AH. Indeed, designing an AH is a hard and time-consuming task. It also requires from authors to have good skills in modeling data logic and programing skills, which most often authors haven't. We have proposed two main contributions that avoid authors having such skills.

On the one hand, we have proposed a solution enabling authors to integrate their domain and user models into existing systems. This solution proposes a semi-automatic merging/specialization process of two models. Therefore, it allows to consider the entire model of an author, on the contrary of existing solutions in AH which propose to authors to instantiate models of existing systems. Thus, the author has to translate his instances. Our solution can be used in other fields than AH in order to merge by specialization two OWL models.

On the other hand, we have proposed a solution enabling authors to express their adaptation strategies, at a high level, independently of any adaptation engine and at a fine granularity. This solution relies on our definition of 22 elementary adaptation patterns and on a semi-automatic process of combining instances of elementary adaptation patterns. We argue that these 22 elementary adaptation patterns are enough for expressing adaptive navigation, as we have identified exhaustive criteria to select resources and to specify how they will be proposed. Furthermore, we have organized the 22 elementary adaptation patterns in a typology in order to be able to look easily over them. Afterward, we have conducted a study of the expressivity of knowledge of our solution versus expressivity in some existing systems, and we have come up with an unifying vision of the domain model and also with an integrated vision of basic actions that can be used to express adaptation.

We have complemented these theoretical qualitative solutions by the development of plug-ins for the Protégé tool[1]. We propose a plug-in per solution. The MESAM plug-in merges two OWL models using a specialization process. This plug-in is already integrated to the Protégé tool. The eapTab plug-in supports the definition of adaptation strategies by using elementary adaptation patterns. It also includes a module able to translate adaptation strategies written by our solution to LAG format.

Furthermore, we have conducted experimentations and evaluations in the e-learning application domain, which have allowed us to validate our solutions. We have made experimentations of MESAM tab on a use case using the GLAM domain model. We have also made evaluations of the ease of use of the eapTab versus a rule-based language and versus a generic adaptation language. We have used GLAM in the first case, and LAG in the second case. Results have shown that it is much easier to use our solution than a rule-based language or a programing language to express adaptation.

After having summarized our most contributions, we are going below to present possible extensions to these contributions.

2 Future work

The work done in this thesis is only a starting point in the help that can be provided for authors of AH. It has received promising critics in conferences or by reviewers of accepted journal paper. Thus, several future works are envisioned.

Future works for merging/ specialization solution : our solution of merging by specialization two models has been motivated by the need of authors to reuse existing adaptations. An interesting extension is to consider relations between existing adaptations and the author's

[1]Protégé tool is available at *protege.stanford.edu/*

user and domain models, to present to authors reused parts of existing adaptations and which parts of his models is not considered yet by these reused adaptations. Therefore, we envision an extension enabling authors of AH to interact with the adaptation model.

Till today, we have performed experimentations using models of the GLAM platform. It would be useful to perform experimentations using other models of the AH field, like the domain model of the LAOS model or the CAM model, in order to study the impact of our solution on their adaptation specifications.

Even, if our solution has been motivated by problems from the AH field, we have proposed a generic solution that can be applied to other application fields, like reusing existing SPARQL queries. Thus, in order to enlarge the reuse of our solution to more fields, we envision to consider other types of mappings than equivalence and specialization, like inclusion. We argue that our meta-model can include more structural knowledge that will serve to deduce other types of mappings.

Future works for expressing adaptation : we propose a solution allowing to express complex adaptation strategies using elementary adaptation patterns. An elementary adaptation pattern has been defined per criterion to select available resources and per criterion to organize selected resources. It can be instantiated on a particular domain model and associated to a specific user characteristic. For users having several characteristics, we have accompanied this solution by a combination process to define complex adaptation strategies. An interesting extension we are about undertaken is to propose a set of combinations of elementary adaptation patterns defining complex adaptations. Furthermore, to illustrate this contribution, we use our combination process on the felder/silverman learning style. On the contrary of existing works on complex adaptation, using our combination process, authors of AH may modify the proposed complex adaptations, and execute them using several existing adaptation engines.

Till today, we have proposed elementary adaptation patterns for the adaptive navigation. It could be useful to extend our library of elementary adaptation patterns by considering the content adaptation and adaptive presentation.

In our solution, authors focus only on what they would like to express as adaptation rather than the technicalities involving writing such adaptation. It would be interesting to include verifications on the resulted adaptation strategies, like checking whether adaptation strategies lead users to their goal. For this purpose, we envision to propose more formalization of our solution.

Finally, we have been collaborating with Alexandra Cristea from the university of Warwick since February 2010. Today, generated adaptation strategies using our solution can be translated to adaptations in the LAG format. We would like to propose a library of existing adaptation strategies that can be used independently of the language in which they are expressed. This library may be enriched by proposing a translator from LAG to our solution in order to reuse existing LAG adaptations. Note that, we have already specified the translation from LAG to our solution[2], but it remains to develop the translator following our specification.

We have also proposed, on the one hand a unifying vision of modeling domain model at a three levels, and on the other hand an integrated vision of the basic actions that could be proposed in adaptation. It could be interesting to face this unifying vision of modeling the domain model to other existing AHS, such as AHA!. We envision also to confront the integrated vision of basic adaptations to recent generic adaptation languages. We argue that these unifying visions will allow identifying a generic unified model that should encompassed a wider area of AH and adaptive engine than each language separately.

[2]Translation from LAG to our solution hasn't been detailed in this thesis, however it is detailed in our journal paper written with Alexandra Cristea.

In this thesis, till today, we have been working on two distinct contributions: (1) supporting reuse of existing authors' models into existing systems and (2) supporting expression of adaptation at a high level, independently of any system and in an easy manner. We envision to relate both contributions in one integrated system. We are thinking that once the author has integrated his models into an existing system in order to reuse existing adaptations, we will show him existing adaptations using our solution based on elementary adaptation patterns. Therefore, the author will be able to understand easily reused adaptations, and also to extend them using our elementary adaptation patterns.

Furthermore, it would be interesting to consider the evolution of domain and user models and their impact on adaptation strategies defined through our solution.

NATIONAL AND INTERNATIONAL PROMOTIONS

In this section, we present an overview of our publications from 2008 to 2011.

JOURNAL PAPERS

IEEE Transaction Learning Technologies. TLT'11 :
Expressing Adaptation Strategies using Adaptation patterns
N. Zemirline, Y. Bourda, C. Reynaud. (14 pages) accepted, to appear current 2011 (PrePrint ISSN: 1939-1382, IEEE computer Society Digital Library. http://doi.ieeecomputersociety.org/10.1109/TLT.2011.15)

UMAI (paper already written (40 pages) being finalized for submission in late June) :
A Study of expressivity and interoperability of Adaptation Languages: EAP framework versus LAG
N. Zemirline, Y. Bourda, A. Cristea, C. Reynaud.

INTERNATIONAL CONFERENCES

10^{th} International Conference on Intelligent Systems Design and Applications. ISDA'10 :
A Pattern-based Framework for expressing Adaptation Strategies in Adaptive Systems
N. Zemirline, Y. Bourda, C. Reynaud.
On page(s): 336 - 341, IEEE, Print ISBN: 978-1-4244-8134-7

11^{th} International Protégé Conference. Protégé'09 :
MESAM : A Protégé Plug-in for the Specialization of Models
N. Zemirline, Y. Bourda, C. Reynaud, F. Popineau. (3 pages)

16^{th} International Conference on Knowledge Engineering and Knowledge Management. EKAW'08
A pattern and a rule-based Approach for reusing Adaptive Hypermedia Creator's Models
N. Zemirline, C. Reynaud, Y. Bourda, F. Popineau.
On page(s): 17-31, Proceedings. LNCS 5268, Springer, ISBN 978-3-540-87695-3

5^{th} International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. AH'08
Assisting of reuse of Adaptive Hypermedia Creator's Models
N. Zemirline, Y. Bourda, C. Reynaud, F. Popineau.
On page(s): 357-360, Proceedings. LNCS 5149, Springer, ISBN 978-3-540-70984-8

NATIONAL CONFERENCES

Ingénierie des Connaissances. IC'10 :
Réutilisation de patrons d'adaptation - Application aux systemes hypermédia adaptatifs
N. Zemirline, Y. Bourda, C. Reynaud. (12 pages)
Dans Actes des 21emes Journées Francophones d'Ingénierie des Connaissances

TECHNICAL REPORTS: all our technical reports had been published on the LRI web site[3]

LRI'11 n 1541 :
A typology of Adaptation Patterns for Expressing Adaptive Navigation in Adaptive Hypermedia
N. Zemirline, Y. Bourda, C. Reynaud. (16 pages)

LRI'11 n 1540 :
Expressing adaptation strategies using adaptation patterns
N. Zemirline, Y. Bourda, C. Reynaud. (15 pages)

LRI'09 n 1529 :
Leveraging Adaptive Web with Adaptation patterns. Technical report
N. Zemirline, Y. Bourda, C. Reynaud. (10 pages)

[3]They are available at http://www.lri.fr/srubrique.php?news=33

Bibliography

[1] Ian H. Beaumont. User modelling in the interactive anatomy tutoring system anatom-tutor. User Model. User-Adapt. Interact, 4(1):21–45, 1994.

[2] Adriana Berlanga and Francisco J. Garcia. Towards reusable adaptive rules. In Workshop on AH and Collaborative Web-based Systems, ICWE, 2004.

[3] Craig Boyle and Antonio O. Encarnacion. Metadoc: An adaptive hypertext reading system. User Model. User-Adapt. Interact., 4(1):1–19, 1994.

[4] Paul De Bra, A. T. M. Aerts, Geert-Jan Houben, and Hongjing Wu. Making general-purpose adaptive hypermedia work. In WebNet, pages 117–123, 2000.

[5] Alexandra I. Brown, Elizabeth J.and Cristea, Craig D. Stewart, and Tim J. Brailsford. Patterns in authoring of adaptive educational hypermedia: A taxonomy of learning styles. Educational Technology & Society, 8(3):77–90, 2005.

[6] Peter Brusilovsky. Adaptive hypermedia. User Modeling and User-Adapted Interaction, 11:87–110, March 2001.

[7] Peter Brusilovsky. Adaptive navigation support. In P. Brusilovsky, A. Kobsa, and W. Neidl, editors, The Adaptive Web: Methods and Strategies of Web Personalization, volume 4321 of LNCS, pages 263–290. Springer, Berlin Heidelberg, USA, New York, 2007.

[8] Peter Brusilovsky, John Eklund, and Elmar W. Schwarz. Web-based education for all: A tool for development adaptive courseware. Computer Networks, 30(1-7):291–300, 1998.

[9] Peter Brusilovsky and Nicola Henze. Open corpus adaptive educational hypermedia. In P. Brusilovsky, A. Kobsa, and W. Neidl, editors, The Adaptive Web: Methods and Strategies of Web Personalization, volume 4321 of LNCS, pages 672–696. Springer, 2007.

[10] Owen Conlan, Vincent P. Wade, Catherine Bruen, and Mark Gargan. Multi-model, metadata driven approach to adaptive hypermedia services for personalized elearning. In 2^{nd} International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pages 100–111, 2002.

[11] Alexandra I. Cristea. Evaluating adaptive hypermedia authoring while teaching adaptive systems. In SAC, pages 929–934, 2004.

[12] Alexandra I. Cristea and Licia Calvi. The three layers of adaptation granularity. In Proceedings of the 9th international conference on User modeling, UM'03, pages 4–14, Berlin, Heidelberg, 2003. Springer-Verlag.

[13] Alexandra I. Cristea and Arnout De Mooij. Laos: Layered www ahs authoring model with algebraic operators. In WWW (Alternate Paper Tracks), 2003.

[14] Alexandra I. Cristea, Davy Floes, Natalia Stach, and Paul De Bra. Mot meets aha! In PEG, 2003.

[15] Alexandra I. Cristea, Maurice Hendrix, and Wolfgang Nejdl. Automatic and manual annotation using flexible schemas for adaptation on the semantic desktop. In EC-TEL, pages 88–102, 2006.

[16] Alexandra I. Cristea, David Smits, Jon Bevan, and Maurice Hendrix. Lag 2.0: Refining a reusable adaptation language and improving on its authoring. In Proceedings of the 4th European Conference on Technology Enhanced Learning: Learning in the Synergy of Multiple Disciplines, EC-TEL '09, pages 7–21, Berlin, Heidelberg, 2009. Springer-Verlag.

[17] Alexandra I. Cristea and Michael Verschoor. The lag grammar for authoring the adaptive web. In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2 - Volume 2, ITCC '04, pages 382–, Washington, DC, USA, 2004. IEEE Computer Society.

[18] David Cristea, Alexandra I.and Smits and Paul De Bra. Towards a generic adaptive hypermedia platform: a conversion case study. J. Digit. Inf., 8(3), 2007.

[19] Declan Dagger, Owen Conlan, and Vincent P. Wade. An architecture for candidacy in adaptive elearning systems to facilitate the reuse of learning resources. In World Conference on E-Learning in Corporate, Government, Healthcare and Higher Education E-Learn, pages 49–56, 2003.

[20] Juan Danculovic, Gustavo Rossi, Daniel Schwabe, and Leonardo Miaton. Patterns for personalized web applications. In Eur. Conf. Pattern Languages of Program-EuroPLoP, pages 34–43, 2001.

[21] Paul De Bra, David Smits, and Natalia Stash. Creating and delivering adaptive courses with aha! In EC-TEL, pages 21–33, 2006.

[22] Paul De Bra, David Smits, and Natalia Stash. The design of aha! In Hypertext, pages 171–195, 2006.

[23] Paul De Bra, Natalia Stash, and David Smits. Creating adaptive web-based applications. In Tutorial at the 10th International Conference on User Modeling, Edinburgh, Scotland, July - 2005.

[24] R. De-Diego. Metodo de mezcla de catalogos electronicos final year project. Technical report, Facultad de Informatica de la Universidad Politécnica, May - 2002.

[25] Paul De Vrieze, Patrick Van Bommel, and Theo P. Van der Weide. A generic adaptivity model in adaptive hypermedia. In 3^{rd} International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pages 344–347, 2004.

[26] Michele Dicerto and Lucia Oneto. Initial implementation of the domain model tool, page 1 - 37, january 2010. Technical report, January - 2010.

[27] Dejing Dou, Drew McDermott, and Peishen Qi. Ontology translation by ontology merging and automated reasoning. In In Proceedings EKAW2002 Workshop on Ontologies for Multi-Agent Systems, pages 3 – 18, 2002.

[28] Dejing Dou, Drew V. McDermott, and Peishen Qi. Ontology translation on the semantic web. Journal on Data Semantics, 2:35–57, 2005.

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

[30] Jonathan G. K. Foss and Alexandra I. Cristea. The next generation authoring adaptive hypermedia: using and evaluating the mot3.0 and peal tools. In 21^{th} ACM Conference on Hypertext and Hypermedia, pages 83–92, 2010.

[31] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[32] Franca Garzotto, Symeon Retalis, and K. Papasalouros Siassiakos. Patterns for designing adaptive/adaptable educational hypermedia. Advanced Technology for Learning, 1(4):23–38, 2004.

[33] Michael R. Genesereth. Mccarthy's idea. In Proceedings of the European Workshop on Logics in Artificial Intelligence, pages 134–142, London, UK, 1996.

[34] Thomas R. Gruber and Gregory R. Olsen. An ontology for engineering mathematics. In KR, pages 258–269, 1994.

[35] Frank G. Halasz and Mayer D. Schwartz. The dexter hypertext reference model. Commun. ACM, 37(2):30–39, 1994.

[36] Maurice Hendrix, Alexandra I. Cristea, Martin Harrigan, Vincent Wade, Frederic Kleinermann, and Olga De Troyer. Design of cam, page 1 - 52, march 2009. Technical report, March - 2009.

[37] Maurice Hendrix and Martin Harrigan. Initial implementation of the concept adaptation model tool, page 1 - 42, november 2009. Technical report, 11 - 11 - 2009.

[38] Cédric Jacquiot. Modélisation logique et générique des systèmes d'hypermédia adaptatifs. PhD thesis, In collaboration between university of Paris-Sud 11 and Supélec - France, defended in December 2006.

[39] Cédric Jacquiot, Yolaine Bourda, Fabrice Popineau, Alexandre Delteil, and Chantal Reynaud. Glam: A generic layered adaptation model for adaptive hypermedia systems. In 3^{rd} International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pages 131–140, 2006.

[40] Ioannis Kazanidis and Maya Satratzemi. Adaptivity in a scorm compliant adaptive educational hypermedia system. In ICWL, volume 4823 of Lecture Notes in Computer Science, pages 196–206. Springer, 2008.

[41] Evgeny Knutov, Paul De Bra, and Mykola Pechenizkiy. Ah 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques. New Rev. Hypermedia Multimedia, 15:5–38, April 2009.

[42] Konstantinos Kotis, George A. Vouros, and Konstantinos Stergiou. Towards automatic merging of domain ontologies: The hcone-merge approach. J. Web Sem., 4(1):60–79, 2006.

[43] Milos Kravcik. Specification of adaptation strategy by fosp method. New Review Hypermedia Multimedia, pages 429–435, 2004.

[44] Patrick Lambrix and He Tan. Sambo - a system for aligning and merging biomedical ontologies. J. Web Sem., 4(3):196–206, 2006.

[45] Ravi Lourdusamy and Gopinath Ganapathy. Feature analysis of ontology mediation tools. Journal of Computer Science, 4(6), pages 437–446, 2008.

[46] Richard McGuinness, Deborah L.and Fikes, James Rice, and Steve Wilder. An environment for merging and testing large ontologies. In Principles of Knowledge Representation and Reasoning Proceedings of the Seventh International Conference, pages 483–493. Morgan Kaufmann, 2000.

[47] Nuria Medina-Medina, Lina García Cabrera, María José Rodríguez-Fórtiz, and José Parets-Llorca. Adaptation in an evolutionary hypermedia system: Using semantic and petri nets. In 2^{nd} International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pages 284–295, 2002.

[48] Fernando Molina-Ortiz, Nuria Medina-Medina, and Lina Garcia-Cabrera. An author tool based on sem-hp for the creation and evolution of adaptive hypermedia systems. In ICWE Workshops, page 12, 2006.

[49] Adam Moore, Timothy J. Brailsford, and Craig D. Stewart. Personally tailored teaching in whurle using conditional transclusion. In Proceedings of the 12th ACM conference on Hypertext and Hypermedia, 12^{th} ACM Conference on Hypertext and Hypermedia, pages 163–164, New York, NY, USA, 2001. ACM.

[50] Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. In Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems, CHI '93, pages 206–213, New York, NY, USA, 1993. ACM.

[51] Natalya Fridman Noy. Tools for mapping and merging ontologies. In Handbook on Ontologies, pages 365–384, 2004.

[52] Natalya Fridman Noy and Mark A. Musen. The prompt suite: interactive tools for ontology merging and mapping. Int. J. Hum.-Comput. Stud., 59:983–1024, December 2003.

[53] James Ohene-Djan and Alvaro A. Fernandes. Modelling personalisable hypermedia: The goldsmiths model. The New Review of Hypermedia and Multimedia, 8:99–137, 2002.

[54] Guillermo Power, Hugh C. Davis, Alexandra I. Cristea, Craig D. Stewart, and Helen Ashman. Goal oriented personalisation with scorm. In ICALT, pages 467–471, 2005.

[55] Livia Predoiu, Cristina Feier, Francois Scharffe, Jos de Bruijn, Francisco Martin-Recuerda, Dimitar Manov, and Marc Ehrig. D4.2.2 state-of-the-art survey on ontology merging and aligning v2. EU-IST Integrated Project (IP) IST-2003-506826 SEKT, pages 1–125, 31-01-2006.

[56] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. The VLDB Journal, 10:334–350, December 2001.

[57] Chitra Ramesh and Aghila Gnanasekaran. Methodology based survey on ontology management. International Journal of Computer Science and Engineering Survey, 1:437–446, August 2010.

[58] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. Object-oriented modeling and design. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[59] Francois Scharffe. Dynamerge: A merging algorithm for structured data integration on the web. In International Workshop on Scalable Web Information Integration and Service at DASFAA, pages 85 – 94, 2007.

[60] Daniel Schwabe and Gustavo Rossi. An object oriented approach to web-based applications design. TAPOS, 4(4):207–225, 1998.

[61] Joshua Scotton, Sabine Moebs, Jennifer McManis, and Alexandra Cristea. Merging strategies for authoring qoe-based adaptive hypermedia. J.UCS - The Journal of Universal Computer Science, Special Issue on Advances in Authoring of Adaptive Web-based Systems, submitted 2010, to appear.

[62] Patricia Seefelder de Assis, Daniel Schwabe, and Demetrius Arraes Nunes. Ashdm - model-driven adaptation and meta-adaptation. In 4^{th} International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pages 213–222, 2006.

[63] Stuart C. Shapiro. Encyclopedia of Artificial Intelligence. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1992.

[64] Kees van der Sluijs, Jan Hidders, Milos Kravcik, Geert-Jan Houben, and Eva Ploum. Grapple d1.1a version: 1.0 cam to adaptation rule translator (specification), page 1 - 37, february 2009. Technical report, February - 2009.

[65] Kees van der Sluijs, Jan Hidders, E Leonardi, and Geer-Jan Houben. Gal: A generic adaptation language for describing adaptive hypermedia. In 1st International Workshop on Dynamic and Adaptive Hypertext: Generic Frameworks, Approaches and Techniques, pages 13–24, 2009.

[66] Natalia Stash. Incorporating Cognitive/ Learning Styles in General-Purpose Adaptive Hypermedia System. PhD thesis, University of Eindhoven, the Netherlands, defended in 2006.

[67] Natalia Stash, Alexandra I. Cristea, and Paul De Bra. Explicit intelligence in adaptive hypermedia: Generic adaptation languages for learning preferences and styles. In International Workshop on Combining Intelligent and Adaptive Hypermedia Methods/Techniques in Web-Based Education Systems, 2005.

[68] Natalia Stash, Alexandra I. Cristea, and Paul De Bra. Learning styles adaptation language for adaptive hypermedia. In 4^{th} International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pages 323–327, 2006.

[69] Natalia Stash, Alexandra I. Cristea, and Paul De Bra. Adaptation languages as vehicles of explicit intelligence in adaptive hypermedia. In International Journal of Continuing Engineering Education and Life-Long Learning (IJCEELL), ISSN 1560-4624, pages 319–336, 2007.

[70] Craig Stewart, Alexandra I. Cristea, Tim Brailsford, and Helen Ashman. Authoring once, delivering many: Creating reusable adaptive coursewar. In 4th IASTED International Conference on Web-Based Education - WBE 2005. Grindelwald, pages 21–23, 2005.

[71] Gerd Stumme and Alexander Maedche. Fca-merge: Bottom-up merging of ontologies. In IJCAI, pages 225–234, 2001.

[72] Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Fast computation of concept lattices using data mining techniques. In Mokrane Bouzeghoub, Matthias Klusch, Werner Nutt, and Ulrike Sattler, editors, Proceedings of the 7th International Workshop on Knowledge Representation meets Databases (KRDB 2000), Berlin, Germany, August 21, 2000, volume 29 of CEUR Workshop Proceedings, pages 129–139. CEUR-WS.org, 2000.

[73] T Theophanis and M.C. Schraefel. Adaptive presentation supporting focus and context. In Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems, 14th Conference on Hypertext and Hypermedia, 2003.

[74] Hongjing Wu, Geert-jan Houben, and Paul De Bra. Aham: A reference model to support adaptive hypermedia authoring. In Proceedings of the quot;Zesde Interdisciplinaire Conferentie Informatiewetenschap, pages 77–88, 1998.

[75] Nadjet Zemirline, Yolaine Bourda, and Chantal Reynaud. A pattern-based framework for expressing adaptation strategies in adaptive systems. In ISDA, pages 336–341, 2010.

[76] Nadjet Zemirline, Yolaine Bourda, and Chantal Reynaud. Réutilisation de patrons d'adaptation - application aux systemes hypermédia adaptatifs. In Ingénierie des Connaissances. IC2010, page 12, 2010.

[77] Nadjet Zemirline, Yolaine Bourda, and Chantal Reynaud. Expressing adaptation strategies using adaptation patterns. IEEE Transaction Learning Technology, to appear, pages 1– 14, 2011.

[78] Nadjet Zemirline, Yolaine Bourda, Chantal Reynaud, and Fabrice Popineau. A protégé plug-in for the specialization of models. In 11th International Protege Conference, page 3, 2009.

[79] Nadjet Zemirline, Chantal Reynaud, Yolaine Bourda, and Fabrice Popineau. Assisting in reuse of adaptive hypermedia creator's models. In 5^{th} International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pages 357–360, 2008.

[80] Nadjet Zemirline, Chantal Reynaud, Yolaine Bourda, and Fabrice Popineau. A pattern and rule-based approach for reusing adaptive hypermedia creator's models. In EKAW, pages 17–31, 2008.

Appendices

APPENDIX A

OWL Meta-Model

Here, we present parts of OWL meta-model. More precisely, we present the OWL class diagram (cf. Figure A.1), the OWL property diagram (cf. Figure A.2) and the OWL restriction diagram (cf. Figure A.3). These three diagrams have been used in order to define the OWL model (cf. Chapter III, Section 5) on which the structural knowledge of the specialization and merging process has been defined.

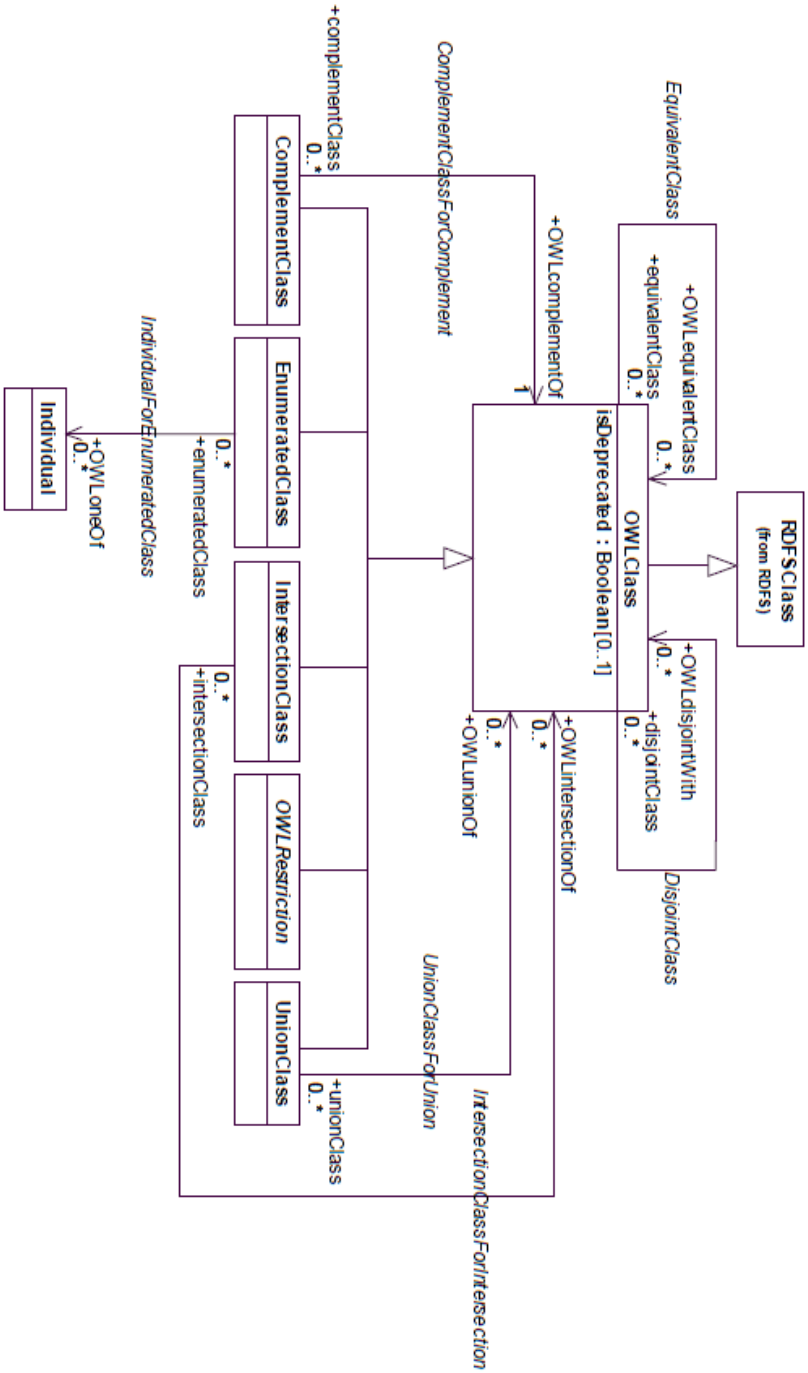

Figure A.1 – The OWL class diagram of the OWL meta-model

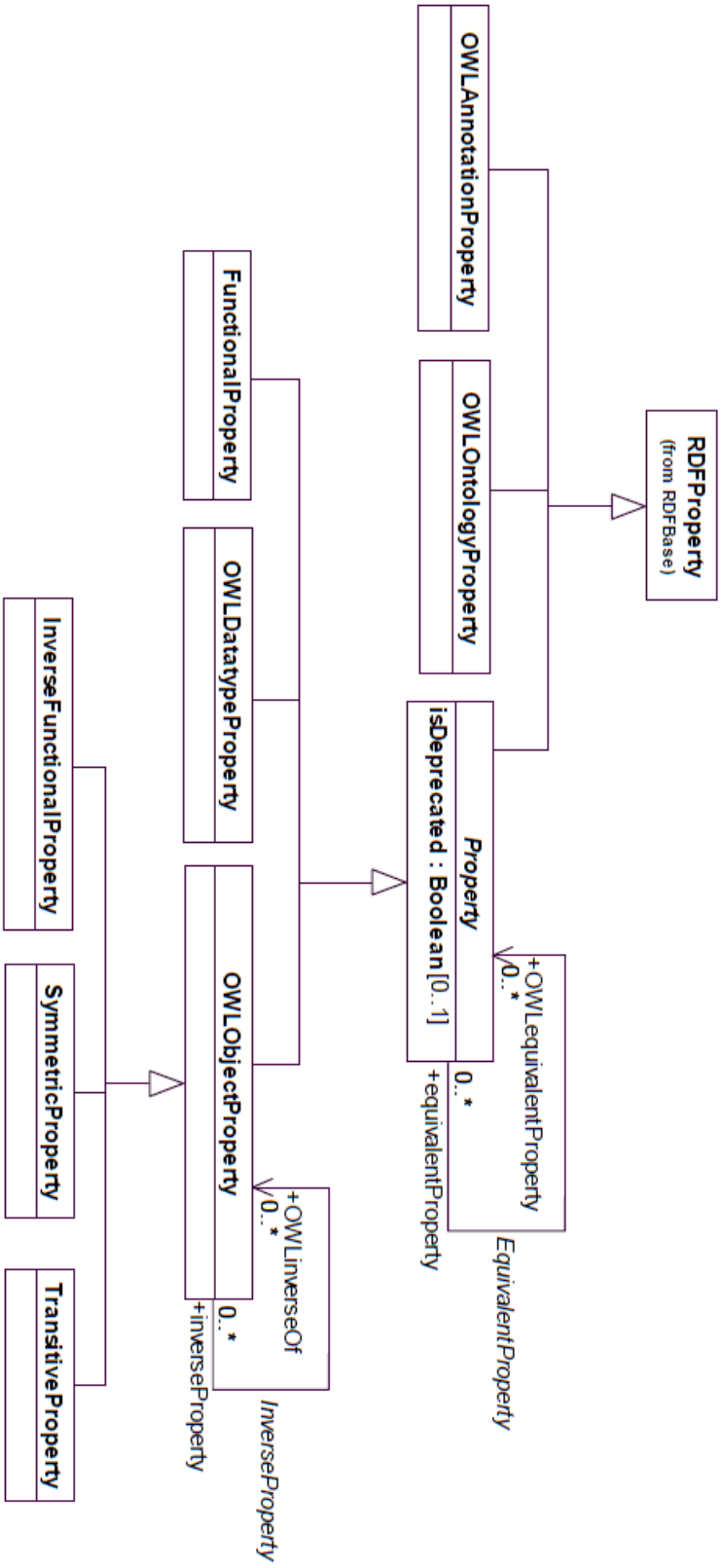

Figure A.2 – The OWL property diagram of the OWL meta-model

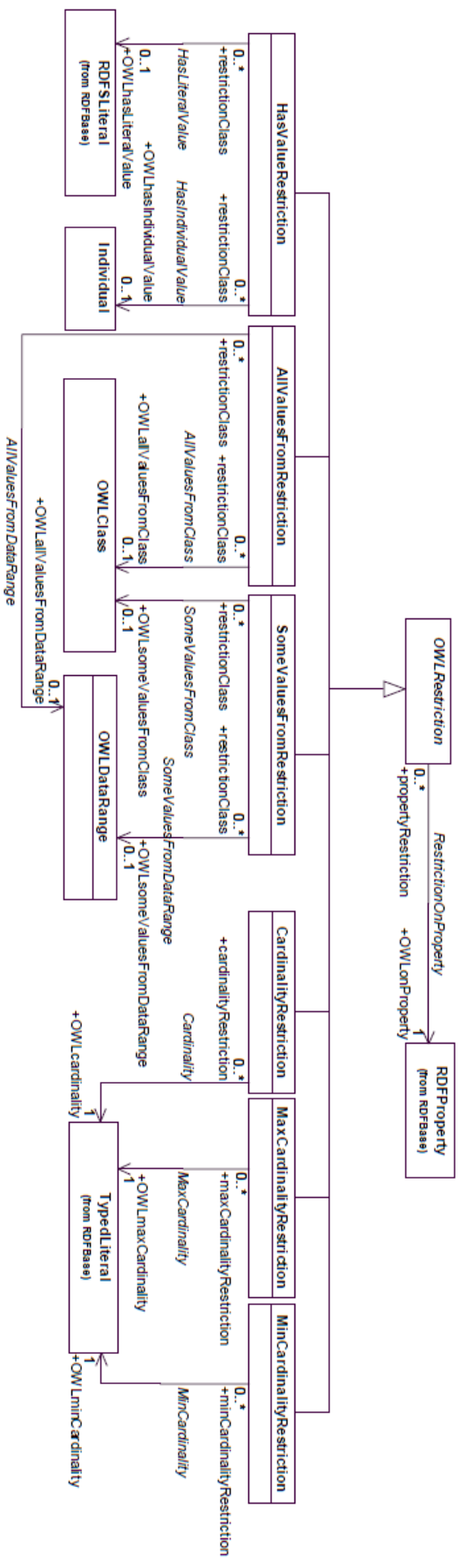

Figure A.3 – The OWL restriction diagram of the OWL meta-model

Appendix B

Library of the defined elementary adaptation patterns

As described before in Chapter 4, we have defined at all 22 elementary adaptation patterns for defining adaptive navigation, which we have organized in a typology (cf. Figure B.1) according to three criteria selection modes, elements of the domain model and possibly a navigational path.

In this appendix, we describe the elementary adaptation patterns per selection mode. The Section 1, Section 2, Section 3, and Section 4 describe successively the elementary adaptation patterns that use the simple selection, ordered, recommended and alternate mode.

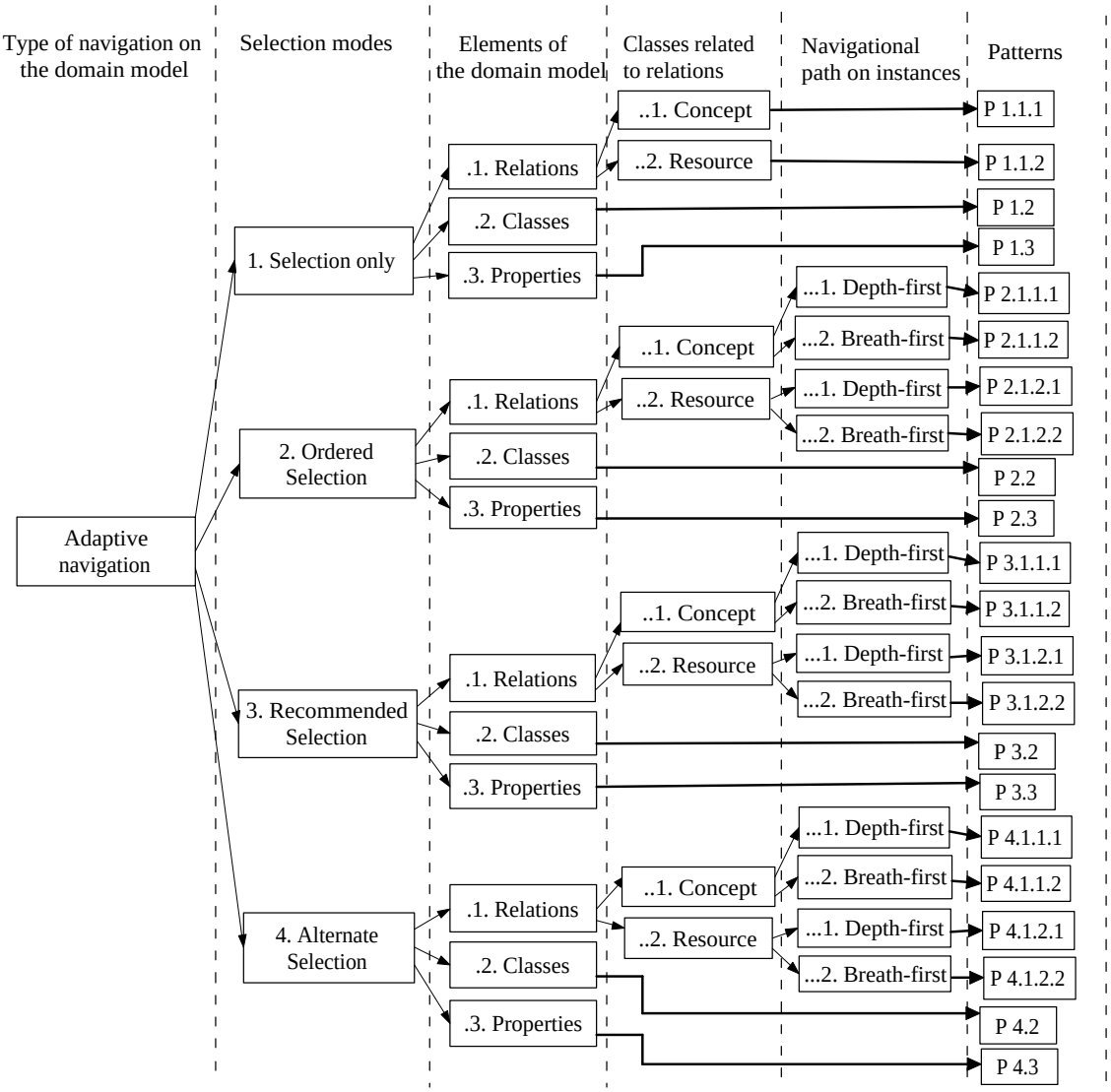

Figure B.1 – Typology of elementary adaptation patterns

1 Elementary adaptation patterns using selection only mode

In Table B.1, we describe the elementary adaptation patterns using the selection only mode. It includes the elementary adaptation patterns P1.1.1, P1.1.2, P1.2 and P1.3 (cf. Chapter 4, Section 4.3, Figure 4.11).

Pattern 1.1.1
Name: Selection Only - Relation - Concept **Intent**: This pattern proposes resources that are linked to concepts by $abstraction$, and where each concept can reach the concept named *goal* directly or indirectly using $relation_i$.

Solution:

- Expression
  - E_1: *linked-transitive(concept, goal,* $relation_i$*)* $\wedge$ *linked(r, concept,* $abstraction$*)*

  According to E_1: selected resources are linked to concepts using $abstraction$, and these concepts are linked directly or indirectly to the $goal$ using $relation_i$.

Constituents:

- concept: a variable representing an instance of the class *Concept*.
- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.
- abstraction: a variable representing 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.

Pattern 1.1.2

Name: Selection Only - Relation - Resource
Intent: This pattern proposes resources that can reach the resource named *goal* directly or indirectly using $relation_i$.
Solution:

- Expression
  - E_1: *linked-transitive(r, goal,* $relation_i$*)*

  According to E_1: selected resources are linked directly or indirectly to the $goal$ using $relation_i$.

Constituents:

- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.

Pattern 1.2

Name: Selection only - Classes
Intent: This pattern allows to select all resources of a specific type.
Solution:

- Expressions
  - E_1: *instanceOf (r,* $Class_1$*)*

  According to E_1: resources proposed to the user must be instances of the class $Class_i$

Constituents:

- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- Class$_i$: a variable representing a subclass of the class *Resource*.

Pattern 1.3

Name: Selection only- particular value of a property
Intent: This pattern allows to select resources according to some values of a property.

Solution:

- Expressions
  - E_1: *characteristicOf(r, $property_i$, op, val)*

  According to E_1: selected resources must have the property $property_i$ and their value must satisfy the comparison test.

Constituents:

- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- property$_i$: a variable representing a property of the class *Resource*.
- val: a variable representing a possible value for the property $property_i$.

Table B.1 – Elementary adaptation patterns using the simple selection mode

2 Elementary adaptation patterns using ordered selection mode

In Table B.2, we describe the elementary adaptation patterns using the odrdered selection mode. It includes the elementary adaptation patterns P2.1.1.1, P2.1.1.2, P2.1.2.1, P2.1.2.2, P2.2 and P2.3 (cf. Chapter 4, Section 4.3, Figure 4.11).

Pattern 2.1.1.1

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(currentR, 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 ones selected by E_2.

Constituents:

- concept: a variable describing an instance of the class *Concept*.
- currentR: 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.

Pattern 2.1.1.2

Name: Ordered Selection - Relation - Concept - breadth first

Intent: This pattern proposes resources that are linked to concepts by $abstraction$, and where each concept can reach the concept named *goal* directly or indirectly using $relation_i$ according to a depth first navigational path.
Solution:

- Expression
  - E_1: *linked-transitive(concept2, goal,* $relation_i$*)* $\wedge$ *linked(r, concept2,* $abstraction$*)* $\wedge$ *distance(concept2, origin,* $relation_i$*)* $\wedge$ *distance(concept, origin,* $relation_i$*)* $\wedge$ *linked(currentR, concept,* $abstraction$*)*
  - 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 are linked directly or indirectly to the $goal$ using $relation_i$ and they have the same distance of the concept which is an abstraction of the current resource from the first resource proposed to the user. According to E_2: selected resources linked to concepts using $abstraction$, and these concepts are linked directly or indirectly to the $goal$ using $relation_i$.
- Meta-expressions
  - $E_1 \prec E_2$

  According to this meta-expression, the set of resources selected by the criterion specified by E_1 is proposed before the ones selected by the criterion specified by E_2.

Constituents:

- concept: a variable representing an instance of the class *Concept*.
- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- origin: a variable representing the first resources proposed to the user.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- currentR: a variable representing an instance of the current resource proposed to the user.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.
- abstraction: a variable representing 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.

Pattern 2.1.2.1

Name: Ordered Selection - Relation - Resource - Depth-first
Intent: This pattern proposes resources that can reach the resource named *goal* directly or indirectly using $relation_i$ according to a depth first navigational path.

Solution:

- Expression
  - E_1: *linked-transitive(resource, goal,* $relation_i$*)* $\wedge$ *linked(currentR, resource,* $relation_i$*)*
  - E_2: *linked-transitive(r, goal,* $relation_i$*)*

  According to E_1: selected resources are linked directly or indirectly to the $goal$ using $relation_i$ and they are linked directly to the current resource using $relation_i$. According to E_2: selected resources are linked directly or indirectly to the $goal$ using $relation_i$.
- Meta-expressions
  - $E_1 \prec E_2$

  According to this meta-expression, the set of resources selected by the criterion specified by E_1 is proposed before the ones selected by the criterion specified by E_2.

Constituents:

- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- currentR: a variable representing an instance of the current resource proposed to the user.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.

Pattern 2.1.2.2

Name: Ordered Selection - Relation - Resource - Breadth-first
Intent: This pattern proposes resources that can reach the resource named *goal* directly or indirectly using $relation_i$ according to a breadth first navigational path.
Solution:

- Expression
  - E_1: *linked-transitive(resource, goal,* $relation_i$*)* $\wedge$ *distance(resource, origin,* $relation_i$*)* $\wedge$ *distance(currentR, origin,* $relation_i$*)*
  - E_2: *linked-transitive(r, goal,* $relation_i$*)*

  According to E_1: selected resources linked directly or indirectly to the $goal$ using $relation_i$ and they have the same distance of the current resource from the first resource proposed to the user. According to E_2: selected resources linked directly or indirectly to the $goal$ using $relation_i$.
- Meta-expressions
  - $E_1 \prec E_2$

  According to this meta-expression, the set of resources selected by the criterion specified by E_1 is proposed before the ones selected by the criterion specified by E_2.

Constituents:

- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- currentR: a variable representing an instance of the current resource proposed to the user.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.

Pattern 2.2

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 the criterion specified by E_i is proposed before the ones selected by the criterion specified by E_j (i < j).

Constituents:

- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- Class$_i$: a variable representing a subclass of the class *Resource*.

Pattern 2.3

Name: Ordered Selection - Properties
Intent: This pattern proposes ordered resources that satisfy some values of the property $property_i$.
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 must have the property $property_i$ and their value must satisfy the comparison test.
- 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 the criterion specified by E_i is proposed before the ones selected by the criterion specified by E_j (i < j).

Constituents:

- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- property$_i$: a variable representing a property of the class *Resource*.
- val: a variable representing a possible value for the property $property_i$.

Table B.2 – Elementary adaptation patterns using the ordered selection mode

3 Elementary adaptation patterns using recommended selection mode

In Table B.3, we describe the elementary adaptation patterns using the recommended selection mode. It includes the elementary adaptation patterns P3.1.1.1, P3.1.1.2, P3.1.2.1, P3.1.2.2, P3.2 and P3.3 (cf. Chapter 4, Section 4.3, Figure 4.11).

Pattern 3.1.1.1

Name: Recommended Selection - Relation - Concept- Depth first

Intent: This pattern proposes recommended resources that are linked to concepts by $abstraction$, and where each concept can reach the concept named *goal* directly or indirectly using $relation_i$ according to a depth-first navigational.
Solution:

- Expression
  - E_1: *linked-transitive(concept2, goal,* $relation_i$*)* $\wedge$ *linked(r, concept2,* $abstraction$*)* $\wedge$ *linked(concept, concept2,* $relation_i$*)* $\wedge$ *linked(currentResource, concept,* $abstraction$*)*
  - E_2: *linked-transitive(concept, goal,* $relation_i$*)* $\wedge$ *linked(r, concept,* $abstraction$*)*

  According to E_1: the selected resources are linked to concepts using $abstraction$, these concepts are linked directly to a concept that is an abstraction of the current resource and they are linked directly or indirectly to the $goal$ using $relation_i$. According to E_2: the selected resources are linked to concepts using $abstraction$, and these concepts are linked directly or indirectly to the $goal$ using $relation_i$.
- Meta-expressions
  - $E_1 \uplus E_2$

  According to this meta-expression, the set of resources selected by the criterion specified by E_1 is recommended rather than the ones selected by the criterion specified by E_2.

Constituents:

- concept: a variable representing an instance of the class *Concept*.
- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- currentResource: a variable representing an instance of the current resource proposed to the user.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.
- abstraction: a variable representing 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.

Pattern 3.1.1.2

Name: Recommended Selection - Relation - Concept - breadth first
Intent: This pattern proposes recommended resources that are linked to concepts by $abstraction$, and where each concept can reach the concept named *goal* directly or indirectly using $relation_i$ according to a depth first navigational path.

Solution:

- Expression
  - E_1: *linked-transitive(concept2, goal,* $relation_i$*)* $\wedge$ *linked(r, concept2,* $abstraction$*)* $\wedge$ *distance(concept2, origin,* $relation_i$*)* $\wedge$ *distance(concept, origin,* $relation_i$*)* $\wedge$ *linked(currentResource, concept,* $abstraction$*)*
  - E_2: *linked-transitive(concept, goal,* $relation_i$*)* $\wedge$ *linked(r, concept,* $abstraction$*)*

  According to E_1: the selected resources are linked to concepts using $abstraction$, these are linked directly or indirectly to the *goal* using $relation_i$ and they have the same distance of the concept which is an abstraction of the current resource from the first resource proposed to the user. According to E_2: the selected resources are linked to concepts using $abstraction$, and these concepts are linked directly or indirectly to the *goal* using $relation_i$.
- Meta-expressions
  - $E_1 \uplus E_2$

  According to this meta-expression, the set of resources selected by the criterion specified by E_1 is recommended rather than the ones selected by the criterion specified by E_2.

Constituents:

- concept: a variable representing an instance of the class *Concept*.
- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- origin: a variable representing the first resources proposed to the user.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- currentResource: a variable representing an instance of the current resource proposed to the user.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.
- abstraction: a variable representing 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.

Pattern 3.1.2.1

Name: Recommended Selection - Relation - Resource - Depth-first
Intent: This pattern proposes recommended resources that can reach the resource named *goal* directly or indirectly using $relation_i$ according to a depth first navigational path.

Solution:

- Expression
  - E_1: *linked-transitive(resource, goal,* $relation_i$*)* $\wedge$ *linked(currentResource, resource,* $relation_i$*)*
  - E_2: *linked-transitive(r, goal,* $relation_i$*)*

  According to E_1: the selected resources are linked directly or indirectly to the $goal$ using $relation_i$ and they are linked directly to the current resource using $relation_i$. According to E_2: the selected resources are linked directly or indirectly to the $goal$ using $relation_i$.
- Meta-expressions
  - $E_1 \uplus E_2$

  According to this meta-expression, the set of resources selected by the criterion specified by E_1 is recommended rather than the ones selected by the criterion specified by E_2.

Constituents:

- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- currentResource: a variable representing an instance of the current resource proposed to the user.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.

Pattern 3.1.2.2

Name: Recommended Selection - Relation - Resource - Breadth-first
Intent: This pattern proposes recommended resources that can reach the resource named *goal* directly or indirectly using $relation_i$ according to a breadth first navigational path.
Solution:

- Expression
  - E_1: *linked-transitive(resource, goal,* $relation_i$*)* $\wedge$ *distance(resource, origin,* $relation_i$*)* $\wedge$ *distance(currentResource, origin,* $relation_i$*)*
  - E_2: *linked-transitive(r, goal,* $relation_i$*)*

  According to E_1: the selected resources are linked directly or indirectly to the $goal$ using $relation_i$ and they have the same distance of the current resource from the first resource proposed to the user. According to E_2: the selected resources are linked directly or indirectly to the $goal$ using $relation_i$.
- Meta-expressions
  - $E_1 \uplus E_2$

  According to this meta-expression, the set of resources selected by the criterion specified by E_1 is recommended rather than the ones selected by the criterion specified by E_2.

Constituents:

- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- origin: a variable representing the first resources proposed to the user.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- currentResource: a variable representing an instance of the current resource proposed to the user.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.

Pattern 3.2

Name: Recommended Selection - Classes
Intent: This patterns proposes recommended resources according to their type.

Solution:

- Expressions
  - E_1: *instanceOf (r, $Class_1$)*
  - ...
  - E_n: *instanceOf (r, $Class_n$)*

  According to Ei: the selected resources are instances of the class $Class_i$
- Meta-expressions
  - $E_i \uplus E_j$, i < j, i = 1..n and j = 1..n.

  According to this meta-expression, the set of resources selected by the criterion specified by E_i is recommended rather than the ones selected by the criterion specified by E_j (i < j).

Constituents:

- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- $Class_i$: a variable representing a subclass of the class *Resource*.

Pattern 3.3

Name: Recommended Selection - Properties
Intent: This pattern proposes resources that satisfy some values of the property $property_i$.
Solution:

- Expressions
  - E_1: *characteristicOf(r, $property_i$, op, val_1)*
  - ....
  - E_n: *characteristicOf(r, $property_i$, op, val_n)*

  According to Ei: The selected resources must have the property $property_i$ and their value must satisfy the comparison test.
- Meta-expressions
  - $E_i \uplus E_j$, i < j, i = 1..n and j = 1..n.

  According to this meta-expression, the set of resources selected by the criterion specified by E_i is recommended rather than the ones selected by the criterion specified by E_j (i < j).

Constituents:

- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- $property_i$: a variable representing a property of the class *Resource*.
- val: a variable representing a possible value for the property $property_i$.

Table B.3 – Elementary adaptation patterns using the recommended selection mode

4 Elementary adaptation patterns using alternate selection mode

In Table B.4, we describe the elementary adaptation patterns using the alternate selection mode. It includes the elementary adaptation patterns P4.1.1.1, P4.1.1.2, P4.1.2.1, P4.1.2.2, P4.2 and P4.3 (cf. Chapter 4, Section 4.3, Figure 4.11).

Pattern 4.1.1.2

Name: Alternate Selection - Relation - Concept- Depth first

Intent: This pattern proposes alternate resources that are linked to concepts by $abstraction$, and where each concept can reach the concept named *goal* directly or indirectly using $relation_i$ according to a depth-first navigational.
Solution:

- Expression
  - E_1: *linked-transitive(concept2, goal,* $relation_i$*)* $\wedge$ *linked(r, concept2,* $abstraction$*)* $\wedge$ *linked(concept, concept2,* $relation_i$*)* $\wedge$ *linked(currentResource, concept,* $abstraction$*)*
  - 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 are linked directly to a concept that is an abstraction of the current resource and they are linked directly or indirectly to the *goal* using $relation_i$. According to E_2: selected resources are linked to concepts using $abstraction$, and these concepts are linked directly or indirectly to the *goal* using $relation_i$.
- Meta-expressions
  - $E_1 \mid E_2$

  According to this meta-expression, the set of resources selected by the criterion specified by E_1 is alternate of the ones selected by the criterion specified by E_2.

Constituents:

- concept: a variable representing an instance of the class *Concept*.
- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- currentResource: a variable representing an instance of the current resource.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.
- abstraction: a variable representing 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.

Pattern 4.1.1.2

Name: Alternate Selection - Relation - Concept - breadth first
Intent: This pattern proposes alternate resources that are linked to concepts by $abstraction$, and where each concept can reach the concept named *goal* directly or indirectly using $relation_i$ according to a depth first navigational path.

Solution:

- Expression
  - E_1: *linked-transitive(concept2, goal,* $relation_i$*)* $\wedge$ *linked(r, concept2,* $abstraction$*)* $\wedge$ *distance(concept2, origin,* $relation_i$*)* $\wedge$ *distance(concept, origin,* $relation_i$*)* $\wedge$ *linked(currentResource, concept,* $abstraction$*)*
  - 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 are linked directly or indirectly to the $goal$ using $relation_i$ and they have the same distance of the concept which is an abstraction of the current resource from the first resource proposed to the user. According to E_2: selected resources are linked to concepts using $abstraction$, and these concepts are linked directly or indirectly to the $goal$ using $relation_i$.
- Meta-expressions
  - $E_1 \mid E_2$

  According to this meta-expression, the set of resources selected by the criterion specified by E_1 is alternate of the ones selected by the criterion specified by E_2.

Constituents:

- concept: a variable representing an instance of the class *Concept*.
- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- origin: a variable representing the first resources proposed to the user.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- currentResource: a variable representing an instance of the current resource.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.
- abstraction: a variable representing 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.

Pattern 4.1.2.1

Name: Alternate Selection - Relation - Resource - Depth-first

Intent: This pattern proposes alternate resources that can reach the resource named *goal* directly or indirectly using $relation_i$ according to a depth first navigational path.

Solution:

- Expression
  - E_1: *linked-transitive(resource, goal,* $relation_i$*)* $\wedge$ *linked(currentResource, resource,* $relation_i$*)*
  - E_2: *linked-transitive(r, goal,* $relation_i$*)*

  According to E_1: selected resources have to be linked directly or indirectly to the $goal$ using $relation_i$ and they are linked directly to the current resource using $relation_i$. According to E_2: selected resources have to be linked directly or indirectly to the $goal$ using $relation_i$.
- Meta-expressions
  - $E_1 \mid E_2$

  According to this meta-expression, the set of resources selected by the criterion specified by E_1 is alternate of the ones selected by the criterion specified by E_2.

Constituents:

- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- currentResource: a variable representing an instance of the current resource.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.

Pattern 4.1.2.2

Name: Alternate Selection - Relation - Resource - Breadth-first
Intent: This pattern proposes alternate resources that can reach the resource named *goal* directly or indirectly using $relation_i$ according to a breadth first navigational path.
Solution:

- Expression
  - E_1: *linked-transitive(resource, goal,* $relation_i$*)* $\wedge$ *distance(resource, origin,* $relation_i$*)* $\wedge$ *distance(currentResource, origin,* $relation_i$*)*
  - E_2: *linked-transitive(r, goal,* $relation_i$*)*

  According to E_1: the selected resources are linked directly or indirectly to the $goal$ using $relation_i$ and they have the same distance of the current resource from the first resource proposed to the user. According to E_2: the selected resources are linked directly or indirectly to the $goal$ using $relation_i$.
- Meta-expressions
  - $E_1 \mid E_2$

  According to this meta-expression, the set of resources selected by the criterion specified by E_1 is alternate of the ones selected by the criterion specified by E_2.

Constituents:

- goal: a variable representing the goal to reach, which is an instance of the class *Concept*.
- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- currentResource: a variable representing an instance of the current resource.
- relation$_i$: a variable representing a relation defined between instances of the class *Concept*.

Pattern 4.2

Name: Alternate Selection - Classes
Intent: This patterns proposes alternative resources according to their type.

Solution:

- Expressions
  - E_1: *instanceOf (r, $Class_1$)*
  - ...
  - E_n: *instanceOf (r, $Class_n$)*

  According to Ei: selected resources must be instances of the class $Class_i$
- 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 the criterion specified by E_i is an alternative of the ones selected by the criterion specified by E_j (i < j).

Constituents:

- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- $Class_i$: a variable representing a subclass of the class *Resource*.

Pattern 4.3

Name: Alternate Selection - Properties
Intent: This pattern proposes alternate resources that satisfy some values of the property $property_i$.
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 must 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 the criterion specified by E_i is alternate of the ones selected by the criterion specified by E_j (i < j).

Constituents:

- r: a variable representing an instance of the class *Resource* or of one of its specializations.
- $property_i$: a variable representing a property of the class *Resource*.
- val: a variable representing a possible value for the property $property_i$.

Table B.4 – Elementary adaptation patterns using the alternate selection mode

5 Summary

In this appendix, we have presented the different elementary adaptation patterns that we propose to define adaptation strategies for the adaptive navigation, accompanied by the the typology in which they are organized. The typology groups all our elementary adaptation patterns in order to make easier their use and their understanding. In fact, each elementary adaptation pattern is defined according to two criteria: a criteria selecting a set of resources and a criteria defining how the selected resources are going to be proposed.

Each elementary adaptation pattern can be used separately or with others in the definition

of an adaptation strategy. The process of combination is automatic and is described in [75]. Note that, the generated adaptation strategies are expressed at a high level, independent of any adaptation engine. This is why we have defined our framework at the top of the GLAM adaptation engine [38] and on the LAG adaptation language [17] which is considered as the first generic adaptation language and which is already interfaced with several existing adaptation engines.

APPENDIX C

Conversion of our elementary adaptation patterns to LAG

In this appendix, we detail the conversion of the defined elementary adaptation patterns (cf. Appendix B, Figure B.1) to LAG.

We have organized the presentation of this appendix per selection mode as the converted elementary adaptation are also organized per selection mode. Therefore, Section 1, Section 2, and Section 3 present successively the conversion of elementary adaptation patterns using the simple selection mode, using the ordered selection mode and the recommended selection mode. The elementary adaptation patterns using the alternate selection mode haven't been converted to LAG as this later does not support to check whether a resource is empty or not. Furthermore, the elementary adaptation patterns that are expressed on concepts directly cannot be converted to LAG as this later does not propose constructors to access concepts but only resources.

Remember that a concept, in LAG is also referred by concept, but a resource is referred by an attribute.

1 Conversion of elementary adaptation patterns using selection only mode to LAG

Table C.1 describes the conversion of P1.1.2 (cf. Table B.1) to LAG

Pattern P1.1.2

In the user model, we add the following attributes

- `goalnum` to count the number of concepts to be shown before the goal message concept. The attribute is independent from each concept.
- `beenthere` related to each concept in order to count the number of times a concept has been visited.

In the goal model, we define the following labels

- `MessageReachGoal` to indicate that the associated concept is the goal message concept.
- `reachGoal` to indicate that the associated concept have to be shown before the goal message concept.

LAG program

```
initialization(
    while true (
        // show all concept
        PM.GM.Concept.show = true
        UM.GM.Concept.beenthere = 0

        // hide the goal message concept
        if GM.Concept.label == MessageReachGoal)
            then ( PM.GM.Concept.show = false)
        )
    )

    UM.GM.Goalnum = 0

    while GM.Concept.label == reachGoal (
        PM.GM.Concept.show = true
        UM.GM.Goalnum += 1
    )

    UM.GM.reach = false
)
```

```
implementation (
      if UM.GM.Concept.access == true then (
            if (UM.GM.Concept.beenthere == 0)
               then ( if (GM.Concept.label == reachGoal)
                     then (UM.GM.Goalnum -= 1)
               )
            UM.GM.Concept.beenthere += 1
      )

      if UM.GM.Goalnum == 0
            then ( UM.GM.reach = true )

       /// show the goal
      if enough (UM.GM.reach == true
            GM.Concept.label == MessageReachGoal, 2)
            then ( PM.GM.Concept.show = true)
)
```

Table C.1 – Conversion of the elementary adaptation pattern *Selection Only - Relation - Resource* to LAG

Table C.2 describes the conversion of P1.2 (cf. Table B.1) to LAG

Pattern P1.2

In the EAP framework, the classification of concepts is done using 'is-a' relation. However, in LAG, the classification of concepts is expressed using attributes[1] of domain concepts (goal model concepts).

LAG program

```
initialization(
      while true(
            if GM.Concept.type == typeConcept1
               then (PM.GM.Concept.show = true )
               else PM.GM.Concept.show = false
            )
)
implementation (
      if GM.Concept.type == typeConcept1
            then (PM.GM.Concept.show = true )
)
```

Table C.2 – Conversion of the elementary adaptation pattern *Selection Only - Classes* to LAG

Table C.3 describes the conversion of P1.3 (cf. Table B.1) to LAG

Pattern P1.3

In LAG, we can only consider two goal model properties 'label', 'weight'. The weight can have only integer values. Additionally, one can create user model variables for each concept (for example, UM.GM.Concept.difficulty).

[1] instances of the class *Attribute*

1. if different for each concept: use instances
   - '\ EAP Course\ EAP\ Video'.difficulty = 35
   - '\ EAP Course \ EAP\ Text'.difficulty = 15
2. if domain specific: use types
   - If GM.Concept.type == Video
   - then UM.GM.Concept.difficulty = 35
3. if pedagogic: use weights (or labels)
   - While true (UM.GM.Concept.difficulty = GM.Concept.weight)

Program in LAG

```
initialization(
    while true (
        if (GM.Concept.weight == val)
            then (PM.GM.Concept.show = true)
            else (PM.GM.Concept.show = false)
    )
)
implementation (
    if ( GM.Concept.weight == val)
        then (PM.GM.Concept.show = true)
)
```

Table C.3 – Conversion of the elementary adaptation pattern *Selection Only - Property* to LAG

2 Conversion of elementary adaptation patterns using ordered selection mode to LAG

Table C.4 describes the conversion of P2.1.2.1 (cf. Table B.2) to LAG

Pattern P2.1.2.1

This pattern can be translated in three different ways, either

1. by exploiting the parent-child relation.
2. by exploiting weights.
3. By building the goal model according to the needed navigational path.

Solution by exploiting the parent-child relation

In the user model, we define the following attributes

- The attribute 'reach' to indicate whether the user has reached the goal or not yet.
- The attribute 'beenthere' is defined for each concept. It indicates how many times each concept has been visited.

In the goal model, we define the following labels

- The label 'start' is used to define the first concept to be proposed.
- The label 'goal' used to indicate the goal message concept.
- The label 'reach' used to indicate that the associated concept is on the path of the goal and should be proposed to the user.

Program in LAG

```
initialization(
    PM.next = true
    PM.ToDo = false
    PM.menu = false
    UM.GM.reach = false
    while true (
        // hide all concepts
        PM.GM.Concept.show = false
        UM.GM.Concept.beenthere = 0
    )
     if GM.Concept.label == start
         then (PM.GM.Concept.show = true)
    )
implementation (
    if enough (UM.GM.Concept.parent.access
        GM.Concept.label == reach
        UM.GM.Concept.beenthere == 0
        UM.GM.reach == false
        , 4)
        then (
            UM.GM.Concept.beenthere += 1
           GM.Concept.show = true
           GM.Concept.parent.show = false
        )
    if enough (UM.GM.Concept.parent.access
        GM.Concept.label == goal
        UM.GM.Concept.beenthere == 0
    , 3)
    then (
        UM.GM.Concept.beenthere += 1
        UM.GM.reach = true
        GM.Concept.show = true
    )
)
```

Table C.4 – Conversion of the elementary adaptation pattern *Ordered Selection - Relation - Resource - Depth first* to LAG

Table C.5 describes the conversion of P2.1.2.2 (cf. Table B.2) to LAG

Pattern P2.1.2.2
This pattern can be translated in three different ways, either 1. by exploiting the parent-child relation. 2. by exploiting weights. 3. By building the goal model according to the needed navigational path. Solution by exploiting the parent-child relation

In the user model, we define the following attributes

- The attribute 'reach' is independent from all concepts. It indicates whether the user has reached the goal or not yet.
- The attribute 'beenthere' is defined for each concept. It indicates how many times each concept has been visited.

In the goal model, we define the following labels

- The label 'start' is used to define the first concept to be proposed.
- The label 'MessageReachGoal' to indicate that the associated concept is the goal message concept.
- The label 'reachGoal' to indicate that the associated concept is on the path of the goal and should be proposed to the user.

Program in LAG

```
initialization(
    PM.next = true
    PM.ToDo = true
    PM.menu = true
    while true(
        if GM.Concept.label == first
        then (PM.GM.Concept.show = true)
    )
    UM.GM.reach = false
    UM.GM.level= 2
)
implementation (
    if enough (GM.Concept.level ¡= UM.GM.level
        GM.Concept.label == reachGoal
        UM.GM.reach == false
        , 3)
      then (PM.GM.Concept.show = true)
      else if (UM.GM.Concept.access == true )
        then (UM.GM.level +=1 )
    if enough (GM.Concept.level ¡= UM.GM.level
        GM.Concept.label == MessageReachGoal
        , 2)
        then (
            UM.GM.reach = true
            GM.Concept.show = true
        )
    )
```

Table C.5 – Conversion of the elementary adaptation pattern *Ordered Selection - Relation - Resource - Breadth first* to LAG

Table C.6 describes the conversion of P2.2 (cf. Table B.2) to LAG

Pattern P2.2

this pattern can be translated either

1. by using weights.
2. by using types of concepts

Solution 1 (solution using weights)

Concepts of type of $Class_1$ will have a weight = 1

```
    ...
    Concepts of type of Class_n will have a weight = n
Program in LAG
initialization(
    while true (
        UM.GM.Concept.beenthere = 0
        if GM.Concept.weight == 1
            then (PM.GM.Concept.show = true)
            else PM.GM.Concept.show = false        )
)
implementation (
    if UM.GM.Concept.access == true then (
        UM.GM.Concept.beenthere += 1
    )
    if UM.GM.Concept.beenthere ¿ GM.Concept.weight
        then (PM.GM.Concept.show = false)
    if UM.GM.Concept.beenthere == GM.Concept.weight
        then (PM.GM.Concept.show = true)
)
Solution 2 (solution using types of concepts)
initialization(
    while true (
        UM.GM.Concept.beenthere = 0
        if GM.Concept.type == typeConcept1
            then (PM.GM.Concept.show = true
            else PM.GM.Concept.show = false
    )
) implementation (
    if UM.GM.Concept.access == true then (
        UM.GM.Concept.beenthere += 1
    )
    if enough (UM.GM.Concept.beenthere == 2 GM.Concept.type == typeConcepti , 2 )
        then (PM.GM.Concept.show = true )
    if enough (UM.GM.Concept.beenthere == 2 GM.Concept.type != typeConcepti , 2 )
        then (PM.GM.Concept.show = false )
    ...
    if enough (UM.GM.Concept.beenthere == n GM.Concept.type == typeConcepti , 2 )
        then (PM.GM.Concept.show = true )
     if enough (UM.GM.Concept.beenthere == n GM.Concept.type != typeConcepti , 2 )
        then (PM.GM.Concept.show = false )
)
```

Table C.6 – Conversion of the elementary adaptation pattern *Ordered Selection - Classes* to LAG

Table C.7 describes the conversion of P2.3 (cf. Table B.2) to LAG

Pattern P2.3
Concepts will have either a weight = 1 to represent concepts with property = value1 ... a weight = n to represent concepts with property = valuen (Because weight can have only integer values.) Program in LAG initialization(while true (

```
            UM.GM.Concept.beenthere = 0
            if GM.Concept.weight == 1
                then (PM.GM.Concept.show = true)
               else PM.GM.Concept.show = false
    )
)
implementation (
    if UM.GM.Concept.access == true then (
        UM.GM.Concept.beenthere += 1
    )
    if enough(UM.GM.Concept.beenthere ¡ 2
        GM.Concept. weight == 2 ,2)
            then (PM.GM.Concept.show = false)
    if enough(UM.GM.Concept.beenthere == 2
        GM.Concept. weight == 3 ,2)
        then (PM.GM.Concept.show = true)
        ...
    if enough(UM.GM.Concept.beenthere ¡ n
        GM.Concept. weight == n ,2)
            then (PM.GM.Concept.show = false)
    if enough(UM.GM.Concept.beenthere == n
        GM.Concept. weight == n , 2)
            then (PM.GM.Concept.show = true)
)
```

Table C.7 – Conversion of the elementary adaptation pattern *Ordered Selection - Property* to LAG

3 Conversion of elementary adaptation patterns using recommended selection mode to LAG

Table C.8 describes the conversion of P3.1.2.1 (cf. Table B.3) to LAG

Pattern P3.1.2.1

This pattern can be translated in three different ways, either

1. by exploiting the parent-child relation.
2. by exploiting weights.
3. By building the goal model according to the needed navigational path.

Solution by exploiting the parent-child relation

In the user model, we define the following attributes

- The attribute 'reach' to indicate whether the user has reached the goal or not yet.
- The attribute 'beenthere' is defined for each concept. It indicates how many times each concept has been visited.

In the goal model, we define the following labels

- The label 'start' is used to define the first concept to be proposed.
- The label 'goal' used to indicate the goal message concept.
- The label 'reach' used to indicate that the associated concept is on the path of the goal and should be proposed to the user.

Program in LAG

```
initialization(
    PM.next = true
    PM.ToDo = false
    PM.menu = false
    UM.GM.reach = false
    while true (
        // hide all concepts
        PM.GM.Concept.show = false
        UM.GM.Concept.beenthere = 0
    )
     if GM.Concept.label == start
         then (PM.GM.Concept.show = true)
    )
implementation (
    if enough (UM.GM.Concept.parent.access
        GM.Concept.label == reach
        UM.GM.Concept.beenthere == 0
        UM.GM.reach == false
        , 4)
        then (
            UM.GM.Concept.beenthere += 1
           GM.Concept.show = true
        )
    if enough (UM.GM.Concept.parent.access
        GM.Concept.label == goal
        UM.GM.Concept.beenthere == 0
    , 3)
    then (
        UM.GM.Concept.beenthere += 1
        UM.GM.reach = true
        GM.Concept.show = true
    )
)
```

Table C.8 – Conversion of the elementary adaptation pattern *Recommended Selection - Relation - Resource - Depth first* to LAG

Table C.9 describes the conversion of P3.1.2.2 (cf. Table B.3) to LAG

Pattern P3.1.2.2
This pattern can be translated in three different ways, either 1. by exploiting the parent-child relation. 2. by exploiting weights. 3. By building the goal model according to the needed navigational path. Solution by exploiting the parent-child relation

In the user model, we define the following attributes

- The attribute 'reach' is independent from all concepts. It indicates whether the user has reached the goal or not yet.
- The attribute 'beenthere' is defined for each concept. It indicates how many times each concept has been visited.

In the goal model, we define the following labels

- The label 'start' is used to define the first concept to be proposed.
- The label 'MessageReachGoal' to indicate that the associated concept is the goal message concept.
- The label 'reachGoal' to indicate that the associated concept is on the path of the goal and should be proposed to the user.

Program in LAG

```
initialization(
    PM.next = true
    PM.ToDo = true
    PM.menu = true
    while true(
        if GM.Concept.label == first
        then (PM.GM.Concept.show = true)
    )
    UM.GM.reach = false
    UM.GM.level= 2
)
implementation (
    if enough (GM.Concept.level ¡= UM.GM.level
        GM.Concept.label == reachGoal
        UM.GM.reach == false
        , 3)
      then (PM.GM.Concept.show = true)
      else if (UM.GM.Concept.access == true )
        then (UM.GM.level +=1 )
    if enough (GM.Concept.level ¡= UM.GM.level
        GM.Concept.label == MessageReachGoal
        , 2)
        then (
            UM.GM.reach = true
            GM.Concept.show = true
        )
    )
```

Table C.9 – Conversion of the elementary adaptation pattern *Recommended Selection - Relation - Resource - Breadth first* to LAG

Table C.10 describes the conversion of P3.2 (cf. Table B.3) to LAG

Pattern P3.2

This pattern can be translated either

1. using weights.
2. using the type of concepts.

However, the order has to be first defined in the CAF file.

Program in LAG (solution using the type of concepts)

```
initialization(
```

```
        while true (
                if (GM.Concept.type == Class1) (
                    PM.GM.Concept.show = true
                ...
                if(GM.Concept.type == Classn) (
                    PM.GM.Concept.show = true
        )
)
implementation (
        if (GM.Concept.type == Class1) (
                PM.GM.Concept.show = true
        )
        ...
        if (GM.Concept.type == Classn) (
                PM.GM.Concept.show = true
        )
)
```

Table C.10 – Conversion of the elementary adaptation pattern *Recommended Selection - classes* to LAG

Table C.11 describes the conversion of P3.3 (cf. Table B.3) to LAG

Pattern P3.3

- The recommendation is expressed by defining an order between attributes in the CAF file.
- The order proposed to the user must match with the order defined in the CAF file.

Program in LAG

```
initialization(
        While true (
                if GM.Concept.weight == 1
                    then (PM.GM.Concept.show = true)
                ...
                if GM.Concept.weight == n
                    then (PM.GM.Concept.show = true)
        )
)
implementation (
        if GM.Concept.weight == 1
                then (PM.GM.Concept.show = true)
        ...
        if GM.Concept.weight == n
                then (PM.GM.Concept.show = true)
)
```

Table C.11 – Conversion of the elementary adaptation pattern *Recommended Selection - property* to LAG