
Thèse de doctorat de l'université Paris XI-Sud

Spécialité :
INFORMATIQUE

Présentée par :
Cédric JACQUIOT

Thèse présentée pour l'obtention du grade de
Docteur de l'université de Paris XI

# Création d'un système d'hypermédia adaptatif générique

Version 1.0

# Table des matières

# Chapitre 1

# Introduction

## 1.1 Prendre en compte l'utilisateur

Aujourd'hui, les sources de contenu hypertexte et hypermédia sont de plus en plus nombreuses : Internet, les intranet d'entreprise ou d'université sont autant d'univers totalement ou partiellement accessibles qui regorgent d'information. Pourtant, ces informations restent bien souvent inexploitées ou très peu exploitées. La raison première de ce problème est le manque fréquent de méta-données, facilitant la recherche et l'extraction de documents pertinents. La seconde raison, certes moins importante dans les systèmes les plus ouverts, mais d'une importance moins secondaire dans les systèmes fermés (cours en ligne, intranet d'entreprise etc.), est que les documents sont souvent peu adaptés à l'utilisateur.

Quel intérêt peut avoir un étudiant à consulter une fiche de cours, s'il n'a pas les connaissance snécessaires pour comprendre les notions qui y sont abordées ? Quel intérêt a un ingénieur qui cherche des informations techniques sur un projet à consulter le budget de ce projet, qu'il aura trouvé en tapant la référence dudit projet ? Quel intérêt aura un technicien à connaître la référence d'une pièce s'il cherche simplement à comprendre comment il doit la monter ? C'est afin d'éviter que des situations indésirables ne se produisent que les systèmes d'hypermédia adaptatifs ont été introduit. Le but de ces systèmes est de proposer des contenus qui reflètent certains aspects de l'utilisateur, modélisé à travers son profil.

Ces systèmes, apparus dès la fin des années 1980, n'ont cessé d'évoluer depuis, et ont pris une importance toute particulière avec l'arrivée d'Internet, et le développement de technologies standards particulièrement adaptées à la conception d'hypermédias (HTML, XML, RDF etc.). De systèmes développés au cas par cas, pour des besoins très particuliers, souvent liés à l'éducation assistée par ordinateur (e-learning), nous sommes parvenus aujourd'hui à des systèmes beaucoup plus génériques, fondés sur des modèles et proposant des outils-auteur pour assister leur utilisation sur un domaine donné.

On retrouve de manière quasi-systématique trois composants essentiels dans les systèmes ac-

tuels :

- Le modèle du domaine : ce modèle est chargé de représenter les connaissances qui sont mises à disposition des utilisateurs dans le système. Il s'agit des concepts abordés, des ressources disponibles (documents XML, images, vidéos etc.), des liens entre ces différents éléments. Le modèle du domaine permet de structurer les différents éléments du domaine afin de faciliter leur adaptation à l'utilisateur.
- Le modèle de l'utilisateur : ce modèle est chargé de prendre en compte les caractéristiques de l'utilisateur. Ces caractéristiques peuvent être très générales : préférences de l'utilisateur en matière d'affichage, parcours professionnel, formes d'apprentissage préférées. Une partie de ces caractéristiques est également liée au domaine : quelles connaissances l'utilisateur a-t-il du domaine traité par le système .
- Le modèle de l'adaptation : ce modèle est chargé de modéliser la façon dont le domaine va être présenté à l'utilisateur en fonction de ses caractéristiques personnelles. Il existe différentes méthodes d'adaptation, que nous verrons en détail au chapitre 2. Classiquement, il est possible d'adapter la sélection de liens proposés à l'utilisateur, de composer des documents à partir de diverses ressources du domaine, ordonnées en fonction de l'utilisateur.

Ces modèles permettent de créer des hypermédias adaptatifs pour des domaines d'application plus ou moins variés, en proposant des formes d'adaptation elles-mêmes plus ou moins variées, selon les modèles. Des règles permettent le plus souvent de décrire l'adaptation souhaitée. Il existe divers formalismes de règles, plus ou moins complexes, que nous étudierons dans le chapitre 2.

## 1.2 Approche proposée

Bien qu'un certain nombre d'approches apportent déjà une certaine généricité dans les hypermédias adaptatifs, nous avons constaté qu'un certain nombre de problèmes ne pouvaient pas être résolus en se fondant sur les modèles actuels. Nous avons mis en avant deux problèmes principaux concernant la conception d'hypermédias adaptatifs :

- Les modèles de données "statiques", c'est-à-dire les modèles de l'utilisateur et du domaine, ne sont pas toujours aussi réutilisables qu'on le souhaiterait. Certains modèles sont très spécifiques à une catégories particulière d'applications. D'autres ont des visées plus génériques. Néanmoins, malgré leur généricité, de nouveaux modèles proches sont conçus, proposant quelques perspectives différentes. Il manque toujours un niveau véritablement générique permettant de concevoir des modèles de l'utilisateur et du domaine spécifiques.
- Les modèles pour l'adaptation fournissent des jeux de règles de plus en plus complexes, ne prenant quasiment pas en compte l'orthogonalité qui existe entre certains axes pris en compte dans l'adaptation : modèle du domaine, caractéristiques spécifiques de l'utilisateur, connaissances du domaine. De plus, la façon dont les règles sont prises en compte pour fournir effectivement de l'adaptation sont des procédés codés directement dans les systèmes, et présentés

de manière informelle.

Aussi proposons nous dans cette thèse des solutions à ces deux grands problèmes.

Concernant les modèles de données statiques, notre approche est la suivante. Constatant qu'il est très difficile de fournir un modèle suffisamment générique pour toutes les sortes de domaines d'application, pour tous les modèles d'adaptation ; constatant également qu'il peut-être souhaitable, notamment pour des raisons de réutilisation de données, de disposer d'un modèle très spécifique, i.e. très peu générique ; nous proposons des modèles génériques, ou méta-modèles, dont le but n'est pas d'être utilisés directement pour la création d'hypermédias adaptatifs, mais de servir de base pour la création de modèles spécifiques. Ces modèles générique sont modélisés en UML, par soucis de respect de standards. Afin de montrer l'utilisation de ces modèles génériques, nous proposons des modèles spécifiques pour le e-learning, qui nous servirons dans le chapitre 6 pour construire une application réelle. Nous montrons également que des modèles très variés, parmi lesquels les plus connus d'entre eux, sont bien des spécialisations de nos modèles génériques.

Concernant le modèle de l'adaptation, notre approche a consisté à fournir un formalisme bien fondé, basé sur le calcul des prédicats du premier ordre, capable de décrire à la fois les règles d'adaptation, la façon dont on les utilise pour déduire des faits, et la façon dont ces faits sont utilisés pour fournir une adaptation à l'utilisateur et opérer les mises à jour de son profil. Ce modèle est fondé sur deux éléments principaux :

- Un modèle de l'adaptation à base de logique situationnelle : il s'agit d'une logique permettant de représenter des éléments concrets d'une situation, et la façon dont on peut évoluer d'une situation à une autre. Utilisée initialement dans le domaine de la robotique, elle a également fait l'objet d'applications dans les domaines de la planification et des web services. Elle repose entièrement sur le calcul des prédicats du premier ordre.
- Un langage de règles, qui fonctionne de paire avec la logique situationnelle. Ce langage prend en compte l'orthogonalité des différents éléments pris en compte pour fournir de l'adaptation. Il simplifie de ce fait la description de l'adaptation, en évitant l'écriture de règles trop complexes, prenant en compte des éléments trop diverses. Il permet de fournir une forme de méta-adaptation, en fonction de critères concernant l'utilisateur. Un système de déduction formel permet d'expliciter de manière formelle la façon dont il faut raisonner à partir de nos règles.

Cette thèse présente ces différents aspects de la façon suivante : dans un premier temps, nous présentons un état de l'art sur les systèmes d'hypermédias adaptatifs, qui met en avant les différents points présentés succinctement dans cette introduction. Ensuite, nous présentons successivement nos travaux sur les modèles génériques pour l'utilisateur, puis pour le domaine. Nous abordons dans le chapitre 5 notre modèle d'adaptation, au coeur de cette thèse, et réutilisant nos modèles génériques. Enfin, nous proposons un exemple d'implémentation du modèle proposé, basé sur un cours d'informatique. Cet exemple met notamment en avant la puissance de la logique situationnelle

pour fournir diverses formes d'adaptation.

# Chapitre 2

# Modélisation des hypermédias adaptatifs

## 2.1 Introduction

Dans ce chapitre, nous allons tenter de passer en revue les différentes techniques de conception des hypermédias adaptatifs. Au fil des années, plusieurs approches ont été développées. Après que de nombreux systèmes aient été construits de manière ad hoc [36, 2, 51, 13, 67, 6] c'est-à-dire pour fournir un contenu adapté dans un seul domaine d'utilisation, au début des années 1990, une méthode de conception [11], détaillant notamment les différentes formes possibles d'adaptation, a été introduite par Peter Brusilovsky. Cette méthode permet de guider un concepteur d'hypermédia adaptatif dans la création d'un tel système, en détaillant l'ensemble des choix possibles.

Vers la fin des années 1990, les premières modélisations génériques des systèmes d'hypermédias adaptatifs sont apparues. Selon l'approche envisagée, le niveau de granularité de la description des systèmes diffère largement. Ainsi, AHAM [8] décrit-il un modèle d'hypermédia adaptatif, reposant sur des modèles de l'utilisateur et du domaine très libres, mais un système d'adaptation utilisant des règles réutilisable très détaillé. Le Munich Model [52], qui repart de la même base que celle utilisée par AHAM, décrit plus en détail les modèles de l'utilisateur et du domaine, ainsi que l'architecture du moteur d'adaptation, sans toutefois revenir sur les mécanismes de l'adaptation à proprement parler. De nombreux modèles, tels le très récent ASHDM [20], décrivent plutôt les éléments à inclure dans un système que la manière précise de fournir de l'adaptation dans ces systèmes.

Quelques tentatives pour capturer l'essence des hypermédias adaptatifs ont été mises en oeuvre, notamment à travers différents travaux de Nicola Henze [33], qui propose notamment une caractérisation logique des hypermédias adaptatifs.

Certains travaux récents ont porté sur l'amélioration des systèmes de règles, comme ceux d'Alexandra Cristea, qui, avec son langage de règles LAG-XLS [65], introduit des notions de méta-adaptation

dans les règles. D'autres approches ont été introduites, comme celles utilisant des composants pédagogiques [23], véritables morceaux de logiciels, plutôt que des ressources statiques (type document XML).

En dehors du domaine des hypermédias adaptatifs, de nombreux travaux ont porté sur la modélisation des domaines et des utilisateurs, notamment en matière de pédagogie à distance (e-learning), qui est un des domaines majeurs d'application des technologies d'adaptation à l'utilisateur [23, 2, 36, 64, 49, 71, 66, 35].

Nous nous proposons d'étudier ces différentes approches dans ce chapitre. Après avoir clairement défini le cadre général des hypermédias adaptatifs, nous présenterons quelques systèmes "ad hoc", qui sont à la base des systèmes actuels de plus en plus génériques. Nous présenterons ensuite différents modèles et systèmes génériques d'hypermédias adaptatifs, avant de nous intéresser en détail aux trois grands axes le plus souvent employés pour décrire ces systèmes : le modèle du domaine, le modèle de l'utilisateur de le modèle de l'adaptation. Enfin, nous étudierons la façon dont sont gérées les règles dans ces systèmes.

## 2.2 Cadre général des hypermédias adaptatifs

Les systèmes d'hypermédias adaptatifs sont devenus particulièrement populaires dès le début des années 1990, où ils servaient d'outils d'accès à l'information conditionné par l'utilisateur. Ces systèmes peuvent être utilisés dans n'importe quel domaine où les applications doivent être utilisées par des personnes aux buts et aux connaissances différents. Peter Brusilowsky [11] définit les hypermédias adaptatifs en ces termes :

> Par systèmes d'hypermédias adaptatifs, nous entendons tout systèmes d'hypertexte ou d'hypermédia qui reflète certains aspects de l'utilisateur dans le modèle de l'utilisateur, et utilise ce modèle pour adapter à l'utilisateur différents aspects visibles du système.

Cette définition très générale permet d'englober de très nombreux systèmes : systèmes de e-learning, systèmes d'information en ligne, système d'aide en ligne, système de recherche d'information, systèmes d'information institutionnels, système de gestion des connaissances, systèmes de recommandations commerciaux etc.

Afin d'encadrer la création de tels systèmes, potentiellement très variés, une méthodologie, toujours utilisée de nos jours, a été proposée en 1995. Nous nous proposons ici d'en décrire les principaux aspects.

Pour concevoir un système d'hypermédia adaptatif, il est nécessaire de répondre à un certains nombres de questions, que nous allons détailler à présent.

*À quels éléments le système peut-il s'adapter ?*

Il existe 4 catégories d'éléments auquel on peut s'adapter dans les systèmes d'hypermédias adaptatifs :

- On peut s'adapter aux connaissances de l'utilisateur, c'est à dire aux connaissances qu'il a des constituants (concepts, documents) du domaine.
- On peut également s'adapter aux buts de l'utilisateur : que doit-il apprendre, quelle(s) tâche(s) souhaite-t-il réaliser ?
- Un système d'hypermédia adaptatif peut également s'adapter à l'expérience et aux compétences de l'utilisateur. Ces éléments diffèrent des connaissances de l'utilisateur de par le fait qu'il s'agit de compétences extérieures au domaine de l'hypermédia adaptatif.
- Enfin, les systèmes d'hypermédias adaptatifs peuvent s'adapter aux préférences de l'utilisateur, comme la façon de présenter les éléments des documents, la taille des caractère, les couleurs etc.

*Que peut-on adapter dans les hypermédias adaptatifs ?*

Les techniques d'adaptation dans les hypermédias adaptatifs sont séparées en deux grandes catégories : adaptation de présentation et de composition des documents, et adaptation de navigation entre les documents.

L'adaptation de présentation est constituée d'adaptation de texte et d'adaptation des médias. Elles consistent, pour le texte, à masquer des explications, à utiliser des variantes des textes, à rajouter des explications concernant les pré-requis etc. Les adaptations de média sont beaucoup moins développées.

*Quelles méthodes d'adaptation peut-on employer ?*

Pour fournir de l'adaptation de navigation, il existe différentes méthodes que nous détaillons ici :

- Le tri des liens, qui consiste à fournir une liste ordonnée de liens en fonction des buts, des connaissances et du document courant ;
- Le masquage de liens, qui permet de limiter les possibilités de navigations en masquant les liens inutiles pour l'utilisateur, en fonction de ses connaissances et de son but ;
- L'annotation de lien, qui consiste à fournir de manière textuelle ou graphique une indication sur le lien, ou sur la pertinence d'utiliser ce lien pour l'utilisateur, en fonction de son profil. On trouve, par exemple, dans certains systèmes, des icônes colorées dont la couleur guide l'utilisateur : *vert* pour les liens les plus souhaitables, *rouges* pour les liens indésirables etc. ;
- L'adaptation du plan proposé à l'utilisateur, qui consiste à proposer des plans de l'hypermédia différents en fonction des connaissances de l'utilisateur ;
- Le guidage du parcours de l'utilisateur, qui consiste à guider le choix du document que l'utilisateur doit lire ensuite. On peut soit fournir un bouton "suivant" à l'utilisateur, soit lui fournir un document formé d'une séquence de différents documents.

Pour fournir de l'adaptation de contenu, voici les méthodes disponibles :

- L'ajout d'explications, qui consiste à fournir des explications supplémentaires à certaines catégories d'utilisateurs ;
- L'ajout d'explications introductives, qui consiste à insérer des explications en début d'une

page présentant un sujet donné. Ces explications sont relatives aux pré-requis nécessaires pour aborder le sujet en question ;

- Les explications comparatives, qui consistent à faire un parallèle avec un concept similaire connu de l'utilisateur. L'explication compare alors les deux concepts ;
- Les variantes d'explications, qui consistent à proposer des variantes des explications sur un même sujet à l'utilisateur ;
- Le tri, qui consiste à trier les fragments composant une page en fonction de leur intérêt pour l'utilisateur.

Pour chacun de ces types d'adaptation, il existe différentes techniques pour les implémenter, que l'on peut mettre en place quand on conçoit un système d'hypermédia adaptatif. Nous étudierons un certain nombre de ces méthodes dans la section 2.3

*Quelles données le système doit-il utiliser ?*

Les mécanismes d'adaptation que l'on peut envisager reposent sur une modélisation de l'utilisateur. Ces modèles, que nous verrons en détail dans la section 2.5.1, peuvent être renseignés manuellement, par conversion de données, ou de manière partiellement ou totalement automatisée. L'utilisation des données concernant l'utilisateur est toujours automatique : elle est l'essence même des hypermédias adaptatifs. La mise à jour des données de l'utilisateur est très souvent automatique, mais peut être effectuée manuellement si nécessaire. La collecte initiale de donnée peut être fournie par l'importation de données existantes, et/ou par l'interrogation directe de l'utilisateur par le système.

Dans les systèmes d'hypermédias adaptatifs «ad hoc», les données concernant le domaine peuvent-être codées en dur, puisqu'ils ne servent qu'à un unique domaine. Dès lors, donner un modèle général pour les domaines est inutile. Néanmoins, de nombreux modèles ou systèmes plus ou moins génériques sont apparus depuis quelques années. Ces modèles sont censés être utilisables sur plusieurs domaines d'application, et fournissent donc des modèles réutilisables pour une ou plusieurs catégories de systèmes adaptatifs.

Nous détaillerons les possibilités de modélisation pour les données dans les sections 2.5.1 et 2.5.2.

Dans cette section, nous avons présenté les éléments essentiels nécessaires à la conception d'hypermédias adaptatifs. Dans les sections suivantes, nous allons détailler les différents aspects présentés ici, à travers l'étude des différents systèmes et modèles existant.

## 2.3 Hypermédias adaptatifs ad hoc

Les premiers systèmes d'hypermédias adaptatifs [51, 2, 6, 13, 36, 67] étaient le plus souvent conçus pour une application particulière : un cours d'anatomie [2], un manuel en ligne [6] etc. Dans cette section, nous allons présenter une sélection de ces systèmes, plus anciens que les modèles

génériques, mais qui permettent d'aborder certains aspects de la création de systèmes d'hypermédias adaptatifs.

### 2.3.1 MetaDoc [6]

MetaDoc est un système d'hypermédia adaptatif qui repose sur la méthode stretchtext, adjointe à un profil de l'utilisateur, pour fournir du texte adapté à l'utilisateur. Ce système a été utilisé pour fournir une version adaptative du manuel technique "Managing the AIX operating system".

Stretchtext est un formalisme qui permet de fournir différents niveaux de détail dans le texte, en fonction de ce que demande l'utilisateur (plus de détail, moins de détail). Ainsi, par rapport au texte d'origine, on peut rajouter du texte avant, pendant ou après, et on peut également définir du texte de remplacement.

Dans la version originale de Stretchtext, c'est l'utilisateur qui demande à avoir plus ou moins de détails. Dans MetaDoc, un profil de l'utilisateur est créé, prenant en compte son niveau de connaissance et choisit automatiquement le niveau de détail fourni à l'utilisateur. L'utilisateur peut ensuite demander plus ou moins de détails sur chaque document qui lui est présenté. Le niveau de connaissance du concept abordé par le document est alors modifié : si l'utilisateur a demandé plus de détail, son niveau de connaissance est diminué, et inversement.

Lors de la première session de l'utilisateur, des stéréotypes lui sont associés pour définir des niveaux de base pour l'ensemble des concepts : l'utilisateur est interrogé sur ses connaissances en informatique en général, et sur sa connaissance du système d'exploitation AIX. Selon ses réponses, son profil initial est établi. Il sera ensuite modifié lors de son parcours de l'hypermédia adaptatif.

Le manuel proposé au format MetaDoc est toujours au moins aussi efficace que le manuel de base, et souvent beaucoup plus efficace, selon les critères pris en compte.

Ce système fournit donc une forme d'adaptation, basée sur la modification du contenu présenté. L'utilisateur n'est pas guidé dans son parcours, et les modifications de texte sont assez limitées : les pages ne peuvent être composées à partir de différentes ressources. La façon de prendre en compte les connaissances de l'utilisateur est statique : on ne peut pas la paramétrer en utilisant, par exemple, des règles. De plus, cette technique nécessite de modifier les ressources utilisées avant de les utiliser dans le système.

### 2.3.2 Anatom-Tutor [2]

Anatom-Tutor est un système d'apprentissage de l'anatomie humaine. Il est spécifiquement orienté vers l'apprentissage. Il est constitué d'un mode d'accès à la base de connaissances. Ce mode est non adaptatif. Il possède également un mode hypertexte qui permet de présenter les éléments du domaine d'une manière adaptée à l'utilisateur. Enfin il dispose d'un mode de questions, qui permet d'interroger l'étudiant et de vérifier son niveau.

Au niveau du domaine, les objets sont représentés de manière hiérarchisée. Ainsi, des déductions peuvent être effectuées sur des données concernant des concepts proches les uns des autres, d'une manière aussi «humaine» que possible. Le système de déduction est codé en dur.

Comme dans MetaDoc, une interrogation de l'utilisateur permet de créer un profil initial. Cette étape est répétée de manière semestrielle, pour mettre à jour les résultats de l'utilisateur. Comme dans MetaDoc, le système utilise des stéréotypes pour inférer les connaissances de l'utilisateur. Ces connaissances sont ensuite affinées au fur et à mesure de l'utilisation du système.

Le mode de questions permet de mettre à jour le profil de l'utilisateur. Il offre également une aide à l'utilisateur en fonction de son profil. Si l'utilisateur réussit à une question, un approfondissement lui est proposé, par exemple. S'il échoue, le processus de déduction qu'aurait dû utiliser l'étudiant est expliqué. S'il était censé connaître la réponse, le système lui propose de lui expliquer pourquoi il aurait dû réussir.

Dans le mode hypertexte, le système adapte le niveau de détail proposé à l'utilisateur en fonction de ses connaissances. Si l'utilisateur demande plus de détail, son niveau de connaissance est mis à jour. Les documents proposés à l'utilisateur sont compilées en utilisant différents fragments de texte, correspondant à ce que souhaite apprendre l'utilisateur. Des questions relatives au texte proposé sont ajoutées au document.

Par rapport à MetaDoc [6], ce système propose donc un mode de questions capable d'expliquer les raisonnements à tenir pour trouver les bonnes réponses. Il est également capable de composer des documents à partir de différentes ressources, plutôt que d'utiliser simplement du stretchtext.

### 2.3.3 Hypadapter [36]

Hypadapter est un système adaptatif donnant accès à un manuel d'apprentissage du Common-LISP. Il est basé sur une sélection de contenu proposé à l'utilisateur.

Dans Hypadapter, chaque document contient un certain nombre de sections potentielles. Ces sections sont représentées par des paires attribut-valeur. Un sujet peut avoir par exemple des attributs descriptifs (notes, exemples, etc.), et des attributs de lien, qui le mettent en relation avec d'autres sujets.

À chaque fois que l'utilisateur demande l'affichage d'un document correspondant à un sujet, un certain nombre d'attributs sont sélectionnés pour être présentés à l'utilisateur. Ces attributs sont sélectionnées par des règles explicites, données en LISP, qui permettent de moduler le "score" de chaque attribut du sujet courant en fonction des connaissances et des stéréotypes de l'utilisateur. Seuls les attributs de meilleur score sont proposés à l'utilisateur.

Hypadapter propose, comme les autres systèmes vus précédemment, une modélisation mixte, implicite et explicite, de l'utilisateur. La modélisation implicite se fait en observant les actions de l'utilisateur. La modélisation explicite se fait en utilisant un questionnaire très détaillé permettant à l'utilisateur de donner son niveau de connaissances dans un certain nombre de sous-domaines de

l'informatique.

Ce système met également l'accent sur les possibilités de navigation de l'utilisateur : il propose plusieurs vues de l'hypermédia pour guider l'utilisateur : index alphabétique, présence d'un historique, de favoris, et carte représentant le système adaptatif. Néanmoins, les cartes ne sont pas adaptatives, et le parcours de l'utilisateur est entièrement libre.

La différence la plus notable entre ce système et ceux que nous avons vu précédemment réside dans le fait qu'il introduit un système de règles explicite qui peut potentiellement être modifié pour améliorer ou modifier le comportement adaptatif du système. De plus, ces règles ne portent pas sur chaque sujet ou chaque attribut traité individuellement, mais sur les catégories d'attributs à présenter à l'utilisateur. Là où, dans un système basé sur StretchText, il faut entièrement modifier les textes pour qu'ils puissent être utilisés dans le système d'hypermédia adaptatif, la méthode d'Hypadapter nécessite beaucoup moins de pré-traitement documentaire : tout au plus une catégorisation des sections relatives à chaque sujet.

### 2.3.4 ELM-ART [13]

ELM-ART et son successeur ELM-ART II proposent un cours adaptatif sur internet pour l'apprentissage du langage LISP. Ce système fournit des informations à l'utilisateur en fonction du parcours que celui-ci a choisi. Il utilise des informations sur les pré-requis des notions à aborder afin de chosir les meilleurs documents à proposer à l'utilisateur.

Le cours est organisé en chapitre, sections et sous-sections. Chaque unité est présenté à l'utilisateur sous la forme d'une page web. Un réseau de concept est greffé au-dessus du niveau des documents : chaque concept du réseau correspond à une page du manuel que représente ELM-ART.

Afin de mettre à jour le profil de l'utilisateur, le système considère que si la page correspondant à un concept a été visitée, le concept est acquis. L'adaptation qui est fournie consiste à trier les liens et à les annoter avec des icônes de couleur. Des tests terminant les sections d'apprentissage permettent de vérifier les connaissances de l'utilisateur.

Ce système ce distingue de ceux que nous venons d'étudier de deux façons : il se focalise sur l'adaptation de navigation, là où les autres systèmes se focalisaient plutôt sur l'adaptation de contenu. Il propose une hiérarchie de documents plus formelle. La nécessité de cette hiérarchie est liée directement au type d'adaptation fourni ici.

### 2.3.5 Conclusions

Dans cette section, nous avons présenté une sélection de systèmes d'hypermédias adaptatifs parmi les premiers systèmes mis en oeuvre. Nous avons ainsi pu dégager à travers des exemples très concrets, puisqu'appliqués le plus souvent à un domaine particulier ou une catégorie de domaines, les enjeux des hypermédias adaptatifs : faciliter l'acquisition des connaissances par la présentation

de documents adaptés aux connaissances de chaque utilisateur, et faciliter l'accès aux différents documents disponibles.

Néanmoins, ces systèmes ont assez rapidement montré leurs limites : beaucoup d'éléments y sont "codés en dur", les données relatives au domaine de l'hypermédia adaptatif font souvent l'objet d'un long pré-traitement. Les modèles de l'utilisateur sont le plus souvent particuliers pour les besoins de chaque système de déduction. Quant aux mécanismes de déduction, ils sont souvent implicites, ou décrits dans des langages de programmation particuliers.

Après que de nombreux systèmes "ad hoc", comme ceux que nous venons de présenter, aient été créés, des systèmes beaucoup plus génériques, et des modèles de construction de systèmes ont été proposés pour faciliter la création d'hypermédias adaptatifs, dans un ou plusieurs domaines d'application. Nous allons présenter quelques-uns de ces systèmes dans la section suivante.

## 2.4 Modèles et systèmes génériques

Dans cette section, nous allons mettre l'accent sur la généricité dans les hypermédias adaptatifs. Comme nous venons de le voir dans la section précédente, la généricité a cruellement manqué aux premiers systèmes d'hypermédias adaptatifs : ils employaient une ou plusieurs techniques très particulières, pour répondre à un besoin particulier, et créer ainsi un système donné.

Afin d'éviter d'avoir à concevoir chaque système d'hypermédia adaptatif en repartant «à zéro», des systèmes plus génériques [34, 12, 68, 22, 28, 10], d'une part, et des modèles pour la création de tels systèmes [8, 52, 20, 15, 72, 32, 24, 18], d'autre part, ont été mis aux point. Dans cette section, nous nous proposons d'étudier les systèmes et les modèles les plus connus dans le domaine des hypermédias adaptatifs.

### 2.4.1 Interbook [12]

Interbook est un système complet de création d'hypermédias adaptatifs pour l'apprentissage, basé sur une version améliorée des principes d'ELM-ART.

L'hypermédia obtenu est composé d'un glossaire et d'un manuel électronique. Le manuel est structuré en sections, sous-section et niveau de base. Le modèle du domaine sert à structurer le contenu du manuel. Chaque élément de base du domaine est lié à un ensemble de concepts. Les concepts sont organisées par la relation de pré-requis, qui permet de savoir quels concepts il est nécessaire d'apprendre avant quels autres. Le glossaire permet de visualiser le plan du domaine. Chaque noeud du plan correspond à un concept différent, et est lié à tous les documents du domaine utilisant ce concept.

Ce système construit également un modèle de l'utilisateur. Ce modèle contient la connaissance qu'a l'utilisateur des différents concepts. Le niveau de connaissance de chaque concept est modifié en fonction des actions de l'utilisateur. Les buts d'apprentissage peuvent être définis pour chaque

utilisateur.

L'adaptation fournie porte sur l'agencement des composants que contient un document, l'annotation de lien par icônes colorées, le guidage direct de l'utilisateur et l'aide basée sur la révision des pré-requis.

Interbook fournit également un outil-auteur qui permet la création assistée d'un système d'hypermédia adaptatif utilisant l'architecture que nous venons de décrire. Pour ce faire, le créateur d'hypermédia adaptatif qui réutilise InterBook doit donner la liste des concepts du domaine, structurer le manuel qui va servir de base à l'hypermédia adaptatif, lier les concepts et les éléments structurés du domaine, et enfin lier entre eux les concepts, en utilisant notamment la relation de pré-requis.

Interbook propose donc une version améliorée d'ELM-ART, et beaucoup plus ouverte sur la réutilisation pour la création d'autres hypermédias adaptatifs à but pédagogique. Bien que disponible en ligne, Interbook n'est pas spécifiquement conçu pour gérer différentes sources de données hétérogènes.

### 2.4.2 NetCoach [68]

NetCoach est le successeur officiel d'ELM-ART II. Il fournit un cadre pour la création de systèmes d'hypermédias adaptatifs. Il utilise une base de connaissance constituée de concepts, qui sont des représentations des pages hypertextes que pourra proposer le système à l'utilisateur. Chaque concept correspond à un élément du domaine, comme les chapitres, sections ou sous-sections. La base de connaissances permet de définir et d'utiliser des relations entre concepts.

Dans NetCoach, le modèle de l'utilisateur est séparé en plusieurs couches, relatives aux connaissances que l'utilisateur a du domaine. La première couche décrit si l'utilisateur a déjà visité une page correspondant à un concept. La deuxième couche contient des informations sur les réussites et les échecs de l'utilisateur aux exercices. La troisième couche décrit pour quels concepts on peut inférer qu'ils sont connus, en fonction de concepts plus avancés que l'utilisateur semble maîtriser. La quatrième couche contient les concepts que l'utilisateur prétend connaître. Ces couches sont indépendantes, la modification d'une couche n'entraîne pas l'écrasement de données dans les autres couches.

NetCoach fournit de l'adaptation en annotant les liens, en guidant l'utilisateur. L'adaptation prend en compte les buts d'apprentissage de l'utilisateur. Le système propose des options pour que l'utilisateur puisse modifier quelques caractéristiques de l'adaptation.

NetCoach utilise un outil auteur complet qui permet la création d'un système éducatif. Celui-ci dispose notamment d'un éditeur de concept, qui permet de définir les relations entre les différents concepts du cours, d'un éditeur de tests, qui permet de fournir des exercices corrigés automatiquement à l'utilisateur. Il est possible de définir quels concepts doivent être acquis pour atteindre un but donné.

Dans NetCoach comme dans Interbook, les mécanismes d'adaptation sont implicites et ne peuvent

pas être modifiés. Ces systèmes sont spécifiquement conçus pour l'éducation, et n'ont pas spécifiquement de visées plus générales.

Nous allons maintenant présenter quelques modèles plus génériques.

### 2.4.3 Personal Reader [22]

Le Personal Reader présente un approche, basée sur les systèmes d'hypermédias adaptatifs pour les corpus de documents fermés, tels que ceux que nous venons de présenter, et qui tente de réconcilier cette approche avec l'ouverture vers le web, qui est intrinsèquement non-adaptatif. Pour ce faire, en plus de proposer une forme d'adaptation classique sur le domaine de documents fourni avec l'hypermédia adaptatif, le Personal Reader propose de faire des suggestion de lien vers des ressources trouvées sur le web.

Les données du Personal Reader sont décrites en RDF. Le langage de règles utilisé est TRIPLE [44], qui permet de faire des requêtes directement sur les éléments décrits en RDF. Ces règles permettent de décrire le niveau d'intérêt d'un composant, le niveau de détail souhaitable etc.

Du point de vue des éléments extérieurs au système, le Personal Reader utilise des requêtes TRIPLE pour faire correspondre les schémas des données provenant de l'extérieur avec celui utilisé pour les données du système. Il utilise des heuristiques pour tenter de préciser les méta-données souvent incomplètes des ressources provenant de l'extérieur du système.

Le Personal Reader présente donc une approche très orientée vers le web sémantique, l'utilisation de méta-données et, de ce fait, l'utilisation de ressources provenant de différentes sources, contrairement à la plupart des systèmes d'hypermédias adaptatifs.

### 2.4.4 AHAM [8]

AHAM (Adaptive Hypermedia Application Model) est un modèle pour la création d'hypermédias adaptatifs, quel que soit le domaine d'utilisation de cet hypermédia. Il est basé sur le Dexter Hypermedia Reference Model [29]. Il a été implémenté dans le système générique AHA ! (Adaptive Hypermedia Architecture) [10]. Il est basé sur un découpage explicite en trois modèles, repris dans de nombreux autres modèles depuis : le modèle du domaine, le modèle de l'utilisateur et le modèle de l'adaptation.

Le modèle du domaine décrit de quelle manière les constituants du domaine traités sont structurés, en utilisant des concepts et des relations. Chaque concept est la représentation abstraite d'un fragment ou d'une agglomération de fragments de ressources physique. Ainsi, une page à présenter à l'utilisateur est un type de concept particulier dans le modèle du domaine.

Le modèle de l'utilisateur représente les connaissances, les préférences, les buts et l'historique de la navigation de l'utilisateur par le biais de tables. Néanmoins, l'accent est surtout mis sur la représentation des connaissances relatives au domaine dans AHAM.

Au niveau de l'adaptation, AHAM propose un système de règles qui permet de décrire de manière complètement personnalisée le comportement adaptatif du système. AHAM supporte l'adaptation de contenu, de lien. Il met à jour le modèle de l'utilisateur à chaque fois que ce dernier visite une page.

Pour utiliser les règles d'adaptation, AHAM requiert l'utilisation d'un moteur d'adaptation. Un sélecteur de page choisit les fragments à présenter à l'utilisateur et un constructeur de page détermine la présentation à appliquer à ces fragments. Les stratégies de ces deux éléments peuvent être redéfinies.

AHAM propose donc un cadre beaucoup plus générique que ce que nous avons présenté précédemment pour la création de systèmes d'hypermédias adaptatifs. La généricité des composants d'AHAM le rend utilisable dans de nombreux cas de figure. La présence de règles indépendantes d'un langage de programmation le rend plus facilement réutilisable. Nous détaillerons ce système de règle dans la section 2.6.1.

AHAM a notamment été réutilisé pour être combiné avec Auld Linky [54], un système d'hypermédia adaptatif ouvert, au contenu susceptible de varier avec le temps, les problèmes de connexion, et à la présentation moins homogène.

### 2.4.5 SCARCE [28]

SCARCE (SemantiC and Adaptive Retrieval and Composition Environment) est un environnement pour la création d'hypermédias adaptatifs, basé sur le web sémantique. Il permet de réaliser de l'annotation de liens, du guidage direct, du masquage partiel ou total de liens et du tri de liens.

L'architecture de l'application est composée de trois couches, similaires à celles proposées dans HERA [37]. La couche sémantique permet d'organiser les relations entre fragments de documents du domaine manipulé. La couche logique permet de gérer la composition syntaxique des documents. La couche de présentation permet de gérer la façon dont les documents sont affichés.

Le domaine est donc composé de fragments de documents, qui forment les documents virtuels après adaptation. L'organisation de ces fragments est donné dans la couche sémantique de SCARCE.

Le modèle de l'utilisateur regroupe les connaissances de l'utilisateur et des stéréotypes de l'utilisateur. Les stéréotypes sont abondamment utilisés pour faciliter la sélection de fragments appropriés.

Afin de fournir l'adaptation, le moteur utilise des règles. Bien que la syntaxe utilisée soit toujours la même, les règles sont utilisées à différents niveaux. Certains règles permettent de déterminer quels d'utilisateur doivent utiliser quels types d'adapatation. Par exemple, on peut écrire une règle qui décrive le fait que le guidage direct est conçu pour les moins de 18 ans. D'autres règles permettent de cibler la pertinence d'un fragment pour différentes catégories d'utilisateur. Ainsi, pour chaque fragment, et pour chaque classe de pertinence (de deux à cinq classes), une règle donne les catégories d'utilisateur qui rentrent dans la classe de pertinence pour le document en question.

En utilisant des techniques issues des documents virtuels, SCARCE présente donc une approche

d'annotation individualisé des fragments à l'aide de règles, là où de nombreux systèmes privilégient des règles génériques pour l'ensemble des fragments.

### 2.4.6 Le Munich Reference Model [52]

Contrairement aux architectures que nous venons de présenter, le Munich Reference Model n'offre pas une architecture implémentée capable de prendre en compte des données de différentes natures. Le Munich Reference Model propose un modèle abstrait pour la conception de systèmes d'hypermédias adaptatifs. Ce modèle est décrit par le biais de deux formalismes : UML (Unified Modeling Language), qui permet de représenter graphiquement les éléments nécessaires aux systèmes adaptatifs, et OCL (Object Constraint Language) qui permet de définir formellement les contraintes posées sur les différents éléments du modèle.

Le Munich Reference Model est basé sur le Dexter Hypertext Reference Model [29], tout comme AHAM [8]. De ce fait, l'architecture décrite dans le Munich Reference Model est assez proche de celle de AHAM. Elle présente néanmoins l'avantage d'être formalisée de manière plus standard et plus précise, notamment en ce qui concerne le modèle de l'utilisateur.

**Dans certains systèmes d'hypermédia adaptatif, les données concernant un utilisateur particulier sont appelées "modèle de l'utilisateur". Il en va de même pour les domaines. De ce fait le Munich Model appelle méta-modèle sa représentation en UML de l'utilisateur et du domaine. D'un point de vue objet, pourtant, les données sur l'utilisateur sont des instances du modèle UML représentant les utilisateurs ou les domaines dans leurs généralités. De ce fait, et dans toute la suite du document, nous parlerons de données utilisateur (resp. domaine), ou d'instance du modèle de l'utilisateur (resp.du domaine) pour un utilisateur (resp. domaine) particulier, et de modèle pour parler des structures permettant de définir des données utilisateur ou domaine.**

Le modèle de l'utilisateur dans le Munich Model est assez simple : il sépare les attributs en deux catégories selon qu'ils dépendent du domaine (connaissances) ou non (préférences, background etc.).

Le modèle du domaine reprend la hiérarchie conceptuelle de AHAM : les concepts représentent dans le modèle des éléments concrets du domaine, ou des agrégations de ces éléments.

Le modèle de l'adaptation est basé sur des règles de type ccondition/action, catégorisée selon leur utilité : construction d'une page, adaptation à l'utilisateur, mise-à-jour du profil de l'utilisateur.

Le Munich Reference Model fournit donc un modèle pour la construction de nouveaux systèmes d'hypermédias adaptatifs. Il reprend un certain nombre d'éléments déjà présentés d'AHAM, mais d'une manière plus formelle et plus adéquate pour l'implémentation de système. Néanmoins, les modèles proposés sont des modèles techniques, et reflètent des éléments qui vont au-delà de la simple analyse du problème, imposant ainsi des choix d'implémentation pas toujours nécessaires.

Un modèle similaire, formalisé en UML, basé sur OOHDM [62], a été proposée dans [18].

### 2.4.7 ASHDM [20]

ASHDM est un modèle récemment proposé pour faciliter la construction de systèmes d'hypermédias adaptatifs. Il présente plusieurs particularités. En plus de réutiliser les trois composants principaux proposés dans AHAM, il réutiliser le découpage proposé dans OOHDM (Object Oriented Hypermedia Design Method) [62], à savoir un découpage en modèle conceptuel, modèle de navigation et modèle de présentation. De plus ASHDM aborde le problème de la méta-adaptation.

Le modèle conceptuel est une ontologie définie pour le web sémantique, qui utilise des concepts et des relations entre ces concepts. Il n'est pas adapté lui-même, puisqu'il est une représentation d'une réalité.

Le modèle de navigation est constitué d'objets de navigations. Ces objets sont considérés comme des vues sur objets du modèle conceptuel. Le modèle de navigation est constitué de classes, de liens, de contextes, de structures d'accès etc. Ces différents éléments sont ensuite mis en relation avec les éléments du modèle conceptuel. Il est possible de fournir de l'adaptation quant à la façon d'associer les éléments du modèle de navigation à ceux du modèle conceptuel.

Le modèle de présentation représente les intéractions possibles entre l'utilisateur et le système.

Le modèle de l'utilisateur est fondé autour d'un noyau de données, telles que son adresse e-mail, son état physiologique, ses préférences etc. Chacun est libre d'augmenter le contenu de ce noyau, ou d'ignorer certains de ses éléments, ce qui, en pratique, n'en fait qu'un guide pour la conception qui ne saurait être utilisé pour faciliter les inférences.

L'adaptation est basée sur des règles de type évènement/condition/action et évènement/action.

La méta-adaptation introduite dans ASHDM repose sur la sélection d'un modèle d'adaptation, en fonction de certaines données de l'utilisateur, par exemple son rôle, son but, les tâches qu'il doit réaliser etc. On peut ainsi choisir d'adapter le contenu des pages uniquement la première fois qu'elles sont visionnées en fonction du type d'utilisateur. La méta-adaptation peut également consister à adapter les modèles, c'est-à-dire à ne garder que certains triplets RDF du modèle, ou changer la valeur de certaines données. Néanmoins, la manière de définir la méta-adaptation, et les techniques qui pourraient y parvenir, ne sont pas imposées.

ASHDM présente donc une architecture très orientée vers les modèles, qui subdivise le système en de nombreuses catégories, permettant de définir un grand nombre d'éléments de très petite granularité. Une implémentation d'ASHDM, basée sur HyperDE [43], a également été proposée.

### 2.4.8 Conclusions

Dans cette section, nous avons montré qu'il existe un certain nombre de systèmes et de modèles génériques. Certains permettent la création d'un hypermédia dans le domaine éducatif, fournissant un certain nombre d'éléments figés et permettant d'importer d'autres éléments. Des modèles de plus en plus génériques sont apparus ces dernières années, comparativement aux modèles les plus «ad hoc» qui ont bercé le début des années 1990. Si certains donnent un cadre général pour assister la

création complète d'une application, sans donner trop de contraintes, d'autres fournissent jusqu'au langage de règle détaillé permettant de décrire différents aspects de l'adaptation. Certains systèmes s'intéressent même à l'ouverture des systèmes d'hypermédias adaptatifs sur des corpus de documents aussi ouverts que possibles, introduisant de nouvelles problématiques.

Par l'étude de ces différents modèles et de ces différents systèmes, on voit apparaître la problématique de la réutilisation des systèmes : chacun possède ses spécificités, une approche plus ou moins complexe, mais aucun ne tente de proposer, même de manière théorique, une réponse globale au problème de la modélisation de ces sytèmes, c'est-à-dire une approche qui ne se contenterait pas de proposer une nouvelle alternative, mais qui proposerait des éléments unifiant les différents modèles, avec un niveau de détail suffisant pour créer de nouveaux modèles de base tout aussi (in)satisfaisant.

Dans les trois sections suivantes, nous nous proposons de détailler les différents modèles proposés pour les trois aspects que l'on retrouve dans chacun des modèles et systèmes décrit dans cette section : le modèle de l'utilisateur, le modèle du domaine, et le modèle de l'adaptation, et plus précisément les formalismes logiques sous-jacent, puisque c'est à eux que nous nous sommes particulièrement intéressés dans cette thèse. Nous mettrons notamment l'accent sur les limites de ces modèles.

## 2.5 Données utilisées pour l'adaptation

Dans cette section, nous nous intéressons aux différents modèles de l'utilisateur et du domaine proposés dans des hypermédias adaptatifs et des domaines proches des hypermédias adaptatifs.

Il existe un certain nombre de modèles pour représenter des utilisateurs et des domaines de documents/concepts. Néanmoins, les hypermédias adaptatifs ont une structure intrinsèquement plus dynamique qu'un certain nombre de systèmes utilisant des modèles de domaine et d'utilisateur : les modèles associés aux hypermédias adaptatifs sont donc structurés différemment, afin de prendre en compte cette caractéristique particulière.

Cette section présente une synthèse des recherches les plus abouties concernant la modélisation de l'utilisateur, d'une part, et celle du domaine d'autre part.

### 2.5.1 Modélisation des utilisateurs

Dans cette partie, nous allons étudier en détail les modèles d'utilisateur qui nous ont paru les plus intéressants. Nous avons divisé notre étude en deux parties. La première passe en revu les standards qui permettent de modéliser les utilisateurs dans des systèmes non adaptatifs, plus particulièrement dans des systèmes pour le e-learning. La seconde partie étudie la façon dont l'utilisateur est modélisé dans les hypermédias adaptatifs.

**Modèles standards**

Dans cette partie, nous présentons deux modèles standards de l'apprenant - utilisateur en contexte pédagogique - non spécifiques aux hypermédias adaptatifs. Il s'agit du PAPI [40] (Public And Private Information) et du modèle de l'utilisateur de l'IMS [47]. Ces deux modèles organisent un certain nombre de données sur l'utilisateur selon différentes structures. Il nous a paru intéressant de présenter ces données, qui constituent un exemple complet et normalisé d'informations que l'on peut vouloir stocker dans un système d'hypermédia adaptatif.

Le modèle du PAPI est composé de six rubriques :

- les relations de l'apprenant (avec d'autres personnes) ;
- les informations de contact de l'apprenant (nom, adresse etc.) ;
- les préférences de l'apprenant (type de média préféré pour les entrées/sorties) ;
- le portfolio de l'apprenant (les médias stockés par lui) ;
- les performances de l'apprenant ;
- les informations concernant la sécurité de l'apprenant (identification en particulier) ;

Dans chacune de ces rubriques, on trouve la définition d'une ou plusieurs structure(s) utiles pour représenter les éléments de ladite rubrique. Par exemple, dans la rubrique "contact de l'apprenant", on trouve une structure `papi_learner_contact_ info_type` qui regroupe le nom, le numéro de téléphone, l'adresse e-mail et l'adresse postale de l'apprenant.

Afin de pouvoir prendre en compte des champs non prévus, certaines structures contiennent un élément de type `papi_learner_bucket_type`, qui permet d'ajouter des paires propriété/valeur dans la structure.

IMS regroupe différents modèles, parmi lesquels un modèle de l'apprenant. Là encore, les propriétés sont placées dans des structures, classées par rubrique :

- identification de l'apprenant ;
- accessibilité pour l'apprenant (langues, handicap, préférences etc.) ;
- buts de l'apprenant ;
- information "qcl" (Qualification, Certificate, License) de l'apprenant (diplômes, certificats etc.) ;
- activités de l'apprenant (activités pour lesquelles les informations sur l'apprenant sont pertinentes) ;
- compétences de l'apprenant (compétences d'apprentissage acquises) ;
- intérêts de l'apprenant (loisirs) ;
- affiliation de l'apprenant (organisations auxquelles il appartient) ;
- livrets de l'apprenant (relevés des notes académiques) ;
- clés de sécurité de l'apprenant, utilisées lors des interactions avec l'apprenant ;
- relations de l'apprenant avec d'autres ressources.

On note que, comme dans le modèle PAPI, un champs `extension` est prévu dans chaque

structure, afin de pouvoir rajouter des propriétés non prévues explicitement par le modèle.

On note également que, dans les deux modèles, la totalité des champs sont optionnels : les personnes réutilisant l'un ou l'autre de ces modèles doivent simplement veiller à satisfaire les dépendances entre champs quand ils en choisissent une sélection.

Pour concevoir un modèle d'utilisateur pour un domaine particulier, il semble donc utile de donner un certain nombre d'attributs prédéfinis, optionnels, et de faciliter la création d'attributs non-prédéfinis.

**Modèles utilisés dans les HA**

Dans les systèmes d'HA, le modèle de l'utilisateur reste en général assez succinct. On y trouve généralement les connaissances de l'utilisateur sur le domaine, qui est l'aspect fondamental des modèles d'utilisateur dans ces systèmes. On trouve aussi des données générales concernant l'utilisateur et non relatives au domaine.

Dans [8], une table contient l'état des connaissances que l'utilisateur a du domaine, ainsi que des préférences ci-nécessaire. L'accent est surtout mis, comme souvent, sur les connaissances liées au domaine de l'application.

Dans [23], le modèle de l'utilisateur est composé explicitement de deux parties : une partie contenant les préférences de l'utilisateur, dans un format attribut-valeur, et une partie contenant les connaissances du domaine. Le plus souvent, il n'est fait aucune distinction explicite entre ces deux types de données, comme dans [28, 8, 12, 68].

Dans certains systèmes, des stéréotypes, calculés en fonction des autres données (préférences et connaissances du domaine) sur l'utilisateur, permettent de donner une vision simplifiée de l'utilisateur (cf. [28, 2, 36]). De plus, dans la plupart des systèmes, on conserve également l'historique des actions de l'utilisateur.

Le *Munich Reference Model* [52] présente un modèle formalisé de l'utilisateur. Ce modèle est représenté sur la figure 2.1. Il est composé d'une classe gestionnaire d'utilisateurs, qui contient un ou plusieurs utilisateurs. Un utilisateur possède des attributs "nom d'utilisateur" et "email". Il contient également d'autres attributs, potentiellement quelconques, et définissables spécifiquement pour chaque système à concevoir. Les attributs sont divisés en deux catégories, tout comme dans [19] : ceux qui dépendent du domaine, et ceux qui n'en dépendent pas. Un attribut est constitué d'un nom et d'une valeur. L'identifiant est unique pour un utilisateur dans toute l'application. Les attributs dépendant du domaine sont liés aux composants du domaine.

**Conclusions sur les modèles de l'utilisateur**

La modélisation de l'utilisateur dans les hypermédias adaptatifs est en général assez succincte : il est possible de définir les paires attribut-valeur que l'on souhaite. De ce fait, les modèles de l'utilisateur des hypermédias adaptatifs sont en général compatibles avec les standards IMS et PAPI :

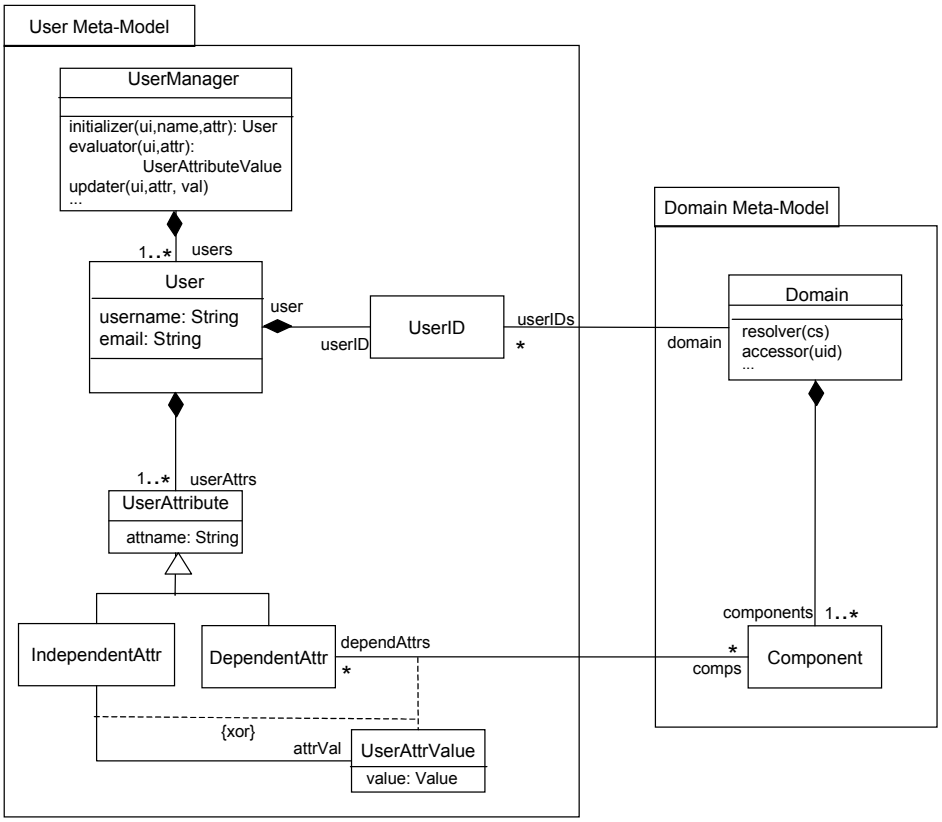


FIG. 2.1 – Le modèle de l'utilisateur dans le *Munich Reference Model*

il suffit de choisir d'utiliser leurs structures et leur vocabulaire pour forme les paries attribut-valeur nécessaires à la constitution d'un modèle pour l'utilisateur.

On note également que dans les modèles, il n'y a pas toujours un fond commun d'attributs absolument indispensables pour tous les modèles de l'utilisateur, ou alors ces attributs sont très succints. Cependant, il apparaît que les notions de but, d'historique, d'identifiant etc. sont très utiles dans les HA. D'autre part, les aspects dynamiques des modèles de l'utilisateur ne sont pas toujours pris en compte dans les modèles, notamment le calcul des stéréotypes n'est pas toujours modélisés.

### 2.5.2 Modélisations des domaines

Dans cette section, nous allons présenter les différents modèles de domaine qui nous ont le plus intéressé. Nous avons choisi d'aborder les données utiles pour le e-learning à titre d'exemple sur les données à représenter dans une première partie, puis différents modèles de domaine pour le e-learning, et enfin différents modèles spécifiquement conçus pour les hypermédias adaptatifs.

#### Description des ressources pédagogiques

Il existe un certain nombre de standards de méta-données pour la descriptions de ressources. Parmi eux, les plus connus sont le Dublin Core [38] (ressources générales), le LOM (Learning Ob-

ject Metadata) [42] et le DC-ED (Dublin Core Education) [39] (ressources pédagogiques). IMS [47] et SCORM (Sharable Content Object Reference Model) [45], standards très utilisés dans la modélisation de ressources liées à l'enseignement, réutilisent le LOM.

Néanmoins, ces standards ne correspondent généralement pas aux besoins des applications souhaitant les réutiliser : des "profils d'application" sont alors créés (e.g. : CanCore [46]), qui désignent les champs à renseigner obligatoirement parmi ceux du standard réutilisé. Ces profils rajoutent également des méta-données, détruisant de fait l'intéropérabilité des standards.

Dans tous ces standards, on retrouve des éléments tels que l'auteur d'un document, la date de publication, la langue du document etc. Ces ressources n'étant pas spécifiques aux hypermédias adaptatifs, elles incluent peu de relations entre ressources essentielles aux HA, comme la notion de pré-requis.

Le Dublin Core n'est pas spécifiquement conçu pour les ressources pédagogiques, mais est souvent appliqué dans ce domaine. Son but est de fournir un ensemble minimal d'éléments qui facilitent la description de documents. Le Dublin Core contient les éléments suivants :

- Title : le nom donné à la ressource ;
- Creator : l'entité responsable du contenu de la ressource ;
- Subject : le sujet du contenu de la ressource ;
- Description : un descriptif du contenu de la ressource ;
- Publisher : une entité responsable de la mise à disposition de la ressource ;
- Contributor : une entité responsable d'avoir contribué au contenu de la ressource ;
- Date : une date associée avec un évènement dans le cycle de vie de la ressource ;
- Type : la nature ou le genre du contenu de la ressource ;
- Format : le type physique ou numérique de la ressource ;
- Identifier : une référence non-ambigüe à la ressource dans un contexte donné ;
- Source : une reference à une ressource dont la ressource décrite est dérivée ;
- Language : la langue du contenu de la ressource ;
- Relation : une référence à une ressource liée ;
- Coverage : l'étendu ou la visée du contenu de la ressource ;
- Rights : de l'information sur les droits contenu dans et sur la ressource.

Le LOM définit une ressource pédagogique comme étant une entité, numérique ou physique, qui peut être utilisée, réutilisé ou référence dans des applications de e-learning. Il fournit un schéma de données conceptuel qui définit la structure d'une instance de méta-données pour une ressource pédagogique. Les éléments permettant la description des ressources pédagogiques sont groupés dans les catégories suivantes :

- General : Catégorie qui regroupe toutes les informations générales pour la description de la ressource pédagogique ;
- Lifecycle : Catégorie qui décrit l'historique et l'état courant de la ressource pédagogique ;

- Meta-metadata : Catégorie qui décrit des informations à propos des méta-données elles-mêmes ;
- Technical : Catégorie qui décrit les caractéristiques techniques de la ressource ;
- Educational : Catégorie qui décrit les caractéristiques pédagogiques de la ressource ;
- Rights : Catégorie qui décrit les droits de propriété intellectuelle et les conditions d'utilisation de la ressource ;
- Relation : Catégorie qui décrit les relations entre la ressource pédagogique et d'autres ressources pédagogiques ;
- Annotation : Catégorie qui permet l'utilisation de commentaires sur l'utilisation pédagogique de la ressource, ainsi que sur le(s) créateur(s) de la ressource ;
- Classification : Catégorie qui décrit si la ressource pédagogique fait partie d'un système de classification particulier.

ADL (Advanced Distributed Learning) est une initiative du ministère de la défense américain, dont le but est d'aider la modernisation de l'apprentissage. SCORM (Sharable Content Object Reference Model) est un ensemble de standards techniques, faisant partie d'ADL, qui facilite l'importation, la recherche, la réutilisation et l'exportation de données pédagogiques numériques. SCORM étend ARIADNE, IMS et le LOM.

#### Modèles de domaines pédagogiques

Il existe plusieurs modèles de ressources pédagogiques. De manière générale, on distingue deux organisations orthogonales de ces ressources : une organisation dite horizontale (nature des ressources), et une organisation dite verticale (organisation hiérarchisée des ressources). L'organisation horizontale permet de représenter les différentes natures de ressources possibles : introduction, conclusion, exercice, exemple etc. L'organisation verticale représente la hiérarchie des entités représentées.

Dans le LMML (Learning Material Markup Language Framework) [70], ces deux organisations sont explicitement séparées. Ce modèle dispose d'une hiérarchie pour décrire des modules d'apprentissage, qui sont subdivisés en sections, sous-sections, etc. Le plus petit type de section est l'unité d'apprentissage, qui correspond à tout ce qui est nécessaire pour un cours donné. Les unités d'apprentissage regroupent différents types d'éléments (exercice, introduction, résumé ...) qui sont représentés dans la hiérarchie horizontale. Cette hiérarchie est présentée figure 2.2.

Un certain nombre de modèles sont présentés dans [30]. Parmi eux, on trouve le modèle UML de e-mi@ge [48]. Ici, les deux types d'organisation sont représentées ensemble : l'unité de base est la séquence, qui peut être de différentes natures (exposé, illustration etc.). Les séquences sont organisées en chapitres, eux-mêmes organisés en modules. Contrairement au LMML, les concepts traités sont pris en compte dans ce modèle.

EML (Educational Modelling Language) [41] propose une hiérarchie verticale plus souple que les précédentes. En effet, Les unités d'apprentissage peuvent être soit simples - elles correspondent

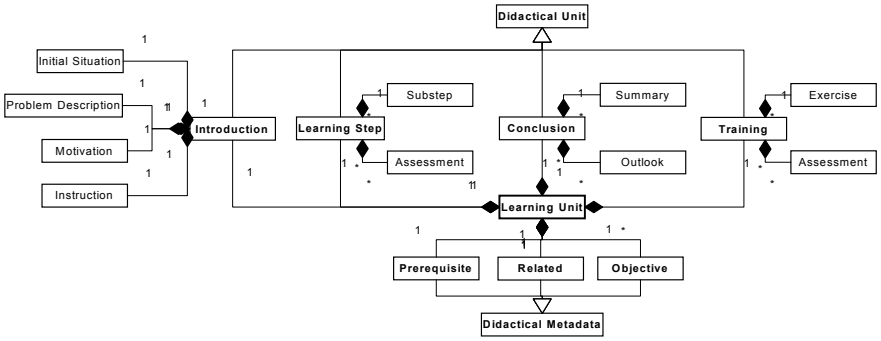


FIG. 2.2 – Les unités d'apprentissage dans le LMML

alors aux unités d'apprentissage vues dans le LMML -, soit composées. Une unité composée peut elle-même être composée d'autres unités composées.

**Modèles du domaine dans les hypermédias adaptatifs**

Dans cette section, nous présentons plus en détail quelques modèles du domaine utilisés dans des modèles connus d'hypermédias adaptatifs.

Dans AHAM [8], le domaine est organisé autour de la notion de "composant concept". Un "composant concept" est une représentation abstraite d'un objet du domaine. Il existe des concepts atomiques, qui correspondent à des fragments d'information. Ces concepts atomiques constituent la base du modèle. Il existe également des concepts composites, regroupant plusieurs concepts (composites ou atomiques), des composites abstraits, qui ne regroupent que des concepts composites, et des "concepts de page", qui ne sont composés que de concepts atomiques. Ainsi, les concepts atomiques correspondent à des ressources indivisibles, les pages à des agrégations de ressources que l'on peut présenter à l'utilisateur, et les composites abstraits à des entités de plus haut niveau, tels que chapitre, section etc.

AHAM utilise également des ancres pour définir des sections particulières dans les concepts. Elles correspondent aux ancres concrètes que l'on peut mettre, par exemple, dans les documents HTML.

Il est également possible de définir différents types de relations entre concept. Selon [72], les relations les plus utilisées dans AHAM sont le type "lien hypertexte", qui indique qu'il existe un lien hypertexte d'un concept vers un autre, le type pré-requis, qui indique que la connaissance d'un concept est nécessaire pour aborder un autre concept, et le type "inhibition" qui indique que la connaissance d'un concept rend inutile la connaissance d'un autre concept.

Le *Munich Reference Model* [52] et les travaux présentés dans [19] ont des éléments communs avec AHAM, et les formalisent en UML. La représentation du domaine donnée dans [52] est reproduite sur la figure 2.3.

On note que ces structures restent très générales : elles permettent de représenter n'importe

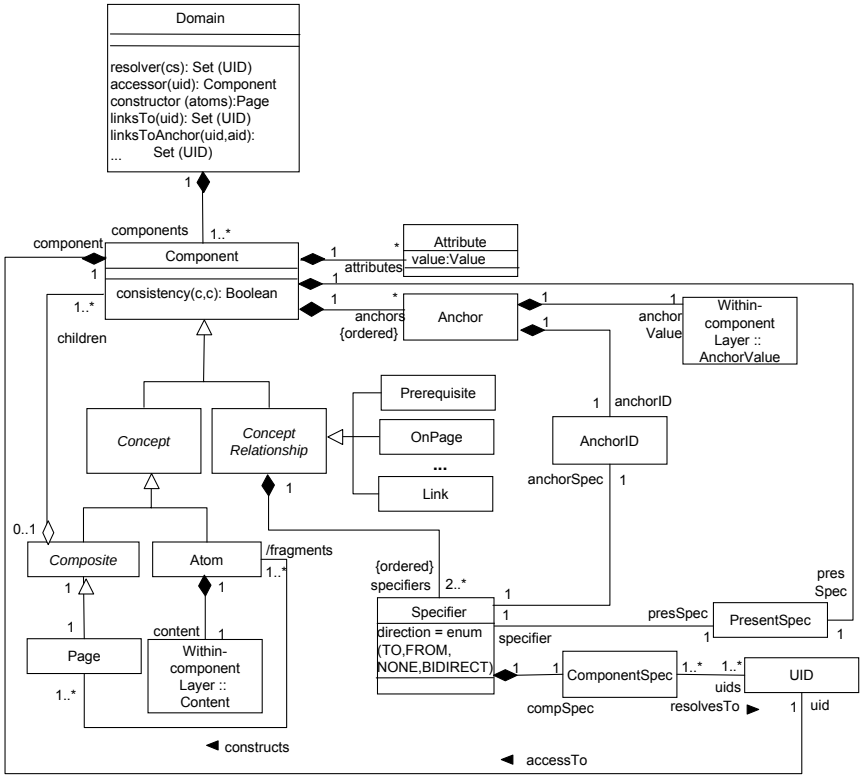


FIG. 2.3 – Le modèle du domaine dans le *Munich Reference Model*

quelle relation entre concept et n'importe quel type de fragments. Néanmoins, la nature (transitivité, réflexivité etc.) des relations ou des fragments ne peut-être précisé outre mesure.

Dans [31], Nicola Henze fait une synthèse des relations utilisées dans divers systèmes d'hypermédias adaptatifs éducatifs. Ces systèmes sont :

- ELM-ART [13], système d'apprentissage du LISP qui fit partie des tous premiers cours adaptatifs présents sur Internet. Dans ELM-ART, les documents sont organisés selon une hiérarchie de chapitres, sections, sous-sections et pages.
- NetCoach [68], qui est le successeur de ELM-ART. Dans NetCoach, les documents sont organisés d'une manière similaire à celle de ELM-ART.
- Interbook [12], qui est un système qui permet la création de livres de cours électroniques basés sur des fichiers Word organisés selon une hiérarchie. Les éléments au niveau atomique sont séparés en différentes catégories : exemples, problèmes, tests etc.
- KBS Hyperbook [34], qui est un système d'HA pour l'apprentissage de JAVA. Dans KBS Hyperbook, les documents sont regroupés en fonction de leur rôle dans le processus d'apprentissage (exemples, descritptions théoriques etc.). Il n'y a pas d'autre structure que ce regroupement dans KBS Hyperbook.

Dans ces différents systèmes, les relations entre documents les plus courantes sont les suivantes :

- la relation de pré-requis ;

- la relation de sortie, qui permet de décrire le fait qu'un document peut-être lu après que le document courant a été lu, indépendamment de ces pré-requis ;
- la relation de successeur, qui permet de décrire le successeur d'un document dans une hiérarchie donnée ;
- la relation "partie de", qui permet de dire si un document est une partie d'un autre document ;
- la relation de test lié, qui permet de décrire qu'un test est nécessaire après la lecture d'un document ;
- la relation de critère, permettant de savoir quelle quantité d'entraînement est nécessaire à la lecture d'un document - cette relation ne lie pas deux documents, mais un document à un critère ;

Les auteurs [23] proposent une distinction entre concepts et ressources. Ici, le terme de concept n'a pas le même sens que dans AHAM. Un concept dans [23] est un élément du domaine d'étude, qui peut-être présenté dans diverses ressources. Les concepts sont organisés selon une hiérarchie de type généralisation/spécialisation. Ceci signifie qu'un concept peut-être plus général qu'un autre concept (le concept de base de données est plus général que le concept de base de données relationnelle) ou plus spécialisé (relation réciproque). La hiérarchie généralisation/spécialisation est complétée par des relations entre concepts comme "étend", "est une antithèse de", "facilite la compréhension de" etc.

Les ressources manipulées dans [23] sont des composants pédagogiques, c'est-à-dire des morceaux de logiciels munis d'interfaces standards. Ils peuvent regrouper un ou plusieurs documents. Ils correspondent chacun à un concept. La notion de pré-requis, fondamentale dans les HA, est utilisée ici entre un composant pédagogique et un concept. Un composant pédagogique peut être étudié, si, et seulement si, le niveau de connaissance d'un ou plusieurs concepts dépasse un seuil établi par le créateur du composant pédagogique. Le niveau de connaissance d'un concept est mis à jour en sortie d'un composant pédagogique traitant de ce concept, grâce à une fonction d'acquisition, présente dans tous les composants pédagogiques.

#### Conclusions sur la modélisation du domaine

Les modèles de représentation du domaine utilisent généralement des structures beaucoup plus complexes que les modèles de l'utilisateur. Les modèles du domaine sont généralement organisés selon un ou plusieurs axes : nature des documents (exercice, illustration, etc.), hiérarchie des documents (chapitre, section etc.), concepts associés aux documents etc. Ces hiérarchies ne sont pas toujours détaillées dans les hypermédias adaptatifs : la plupart se contentent de permettre le découpage des documents à l'aide d'ancres et la mise en place d'une hiérarchie verticale informelle. Ils permettent également parfois d'annoter les documents à l'aide de méta-données.

On note que la grande diversité des domaines d'application potentiels rend très difficile la description d'éléments précis et clairement définis dans les modèles d'hypermédias adaptatifs. Nous

montrerons dans le chapitre 4 la solution que nous apportons à ce problème particulier.

## 2.6 Règles, logique, et adaptation

Il existe de nombreuses approches pour modéliser l'adaptation à l'aide de logique [33, 8, 72, 65, 16, 21, 14]. Dans cette section, nous allons tenter de passer en revue ces mécanismes logiques sous-jacents à la création d'adaptation dans les systèmes d'hypermédias adaptatifs. Dans un premier temps, nous allons détailler le formalisme de règle le plus utilisé actuellement, puis une extension de ce formalisme pour le rendre plus puissant.

Nous présenterons ensuite une étude de Nicola Henze sur la caractérisation en logique des prédicats du premier ordre des systèmes d'hypermédias adaptatifs [33].

### 2.6.1 Règles condition-action

Dans cette section, nous allons détailler la syntaxe et l'utilisation des règles de type "condition-action" utilisables notamment dans AHAM [8], mais aussi de nombreux autres systèmes.

A l'origine, les règles dans AHAM étaient de la forme $< Rule,\ Phase,\ Propagate >$ où $Rule$ est une règle de type clause de Horn, $Phase$ spécifie le moment du déclenchement de la règle, et $Propagate$ le fait que la règle puisse, ou non, entraîner le déclenchement d'autres règles. Les règles sont déclenchées si l'un de leur prémice est modifié.

Ce langage de règle a été revu [72] pour proposer un formalisme plus standard, réutilisant le principe de condition-action utilisé dans les bases de données notamment. Les règles prennent alors la forme $< rule >::= C \rightarrow A. < rule >$ est l'identifiant de la règle, $C$ la condition de déclenchement de la règle et $A$ son action.

Les conditions des règles sont des requêtes qui permettent de sélectionner des données du domaine ou de l'utilisateur. Par exemple, on peut demander la sélections des pages auxquelles l'utilisateur a accès à un instant donné. Ces requêtes sont de la forme $Select < list > where < condition >$.

Les actions sont des mises à jour de certains éléments concernant l'utilisateur. Ainsi, il est possible de définir sous quelles conditions il devient intéressant de proposer un document à l'utilisateur. Les actions sont de la forme $Update < liste\_de\_mise\_\_jour > where < condition >$

L'utilisation de ce système de règle soulève un certain nombre de problèmes : l'exécution des règles se finit-elle dans tous les cas (terminaison), et obtient-t-on toujours la même adaptation dans une même situation (confluence). Le problème de terminaison est lié à l'appel potentiellement récursif de règles en boucle : l'action d'une règle peut déclencher la condition d'une autre règle. Le problème de confluence est lié à l'ordre d'exécution des règles dont la condition devient vrai à un instant donné.

Des restrictions sont proposées pour garantir la terminaison et la confluence : ces deux aspects

ne peuvent pas être garantis dans le modèle général. Un algorithme de propagation des règles est notamment proposé pour assurer la confluence, et un nombre minimal de règles de création des règles condition-action sont posées pour garantir la terminaison.

### 2.6.2 LAG-XLS [65]

LAG-XLS est un langage de règle à visée aussi générique que possible, spécifiquement conçu pour les styles d'apprentissage dans AHA ! **??**. LAG-XLS est basé sur XML. Une DTD (Document Type Definition) permet d'en définir formellement la grammaire.

Le langage LAG-XLS est formé autour d'éléments de structure des règles, d'éléments descriptifs et d'éléments de contenu des règles.

Les éléments de structure sont $if$, $then$, $else$, $condition$ et $action$. On retrouve la structure programmatique classique "$if$ ... $then$ ... $else$", qui permet d'encadrer les conditions et leur conséquences. La balise $condition$ permet de définir des conditions pour la structure $if$ ... $then$ ... $else$. La balise $action$ permet de spécifier comment le modèle de l'utilsateur est mis à jour.

Les éléments descriptifs sont $strategy$ et $description$, qui permettent respectivement de nommer et de décrire la stratégie d'adaptation donnée dans la règle.

Les éléments de contenu sont $select$, $sort$, $showContent$, $showContentDefault$, $showLink$, $navigationType$ :

- showLink : permet de spécifier un lien à présenter ;
- showContent : permet de spécifier un contenu à présenter ;
- showDefaultContent : permet de spécifier qu'il faut présenter le contenu par défaut si aucune autre représentation d'un concept n'est disponible ;
- navigationType : permet de spécifier le type de structure de navigation à utiliser, comme la navigation en profondeur d'abord, ou en largeur d'abord ;
- select permet de sélectionner des éléments sur lesquels appliquer les formes d'adaptation données par les balises ci-dessus ;
- sort : permet de trier des éléments selon un critère donné, par exemple les médias selon leur nature.

Ce langage permet ainsi de décrire l'adaptation à un niveau plus élevé, en considérant des choix de stratégie d'adaptation (montrer un lien, une image, un texte) en fonction des préférences de l'utilisateur. Ce langage permet également de définir des méta-stratégies. Les méta-stratégies permettent de moduler, voire de redéfinir la perception que le système a de l'utilisateur, en fonctions de ses préférences et de ses actions. La perception ainsi modifiée de l'utilisateur entraîne en conséquence une modification du comportement des règles : les prémisses ne sont plus réalisées dans les même circonstances.

LAG-XLS réutilise les avantages du langage d'adaptation LAG, qui repose sur une architecture basée sur trois couches. La première est constituée de règles de type "$if$...$then$", la seconde est

constitué d'un langage d'adaptation sémantique. La troisième gère des procédures d'adaptation. Le langage LAG propose de ce fait une approche assez programmatique des règles, avec notamment la possibilité d'utiliser des boucles `while`, `for`, etc.

### 2.6.3 Caractérisation logique des hypermédias adaptatifs

Dans cette section, nous allons présenter une étude [33] sur la logique et les hypermédias adaptatifs. Le but de cette démarche est de pouvoir analyser et comparer les systèmes d'hypermédias adaptatifs à but pédagogique. La solution proposée est basée sur le calcul des prédicats du premier ordre, et permet de caractériser certains aspects des hypermédias adaptatifs pédagogiques.

Le modèle proposé suggère de découper les systèmes adaptatifs pédagogiques en quatre composants : l'espace des documents, le modèle de l'utilisateur, les observateurs des actions de l'utilisateur, et le composant d'adaptation.

Les domaines peuvent être représentés très simplement à l'aide de prédicat qui définissent les relations entre constituants du domaine. Par exemple, on peut décrire que le document $doc_1$ est un prérequis du document $doc_1$ en écrivant : $preq(doc_1, doc_2)$. Il en va de même pour les concepts manipulés.

Le modèle de l'utilisateur peut-être représenté à l'aide de propriété. Par exemple, pour exprimer que Paul a un attribut $a$ avec la valeur $v$, il suffit de définir le prédicat $has\_property(Paul,\ a,\ v)$.

Les observateurs permettent de capturer les interactions de l'utilisateur avec le système d'hypermédia adaptatif. Ainsi, on peut définir un prédicat $obs$ qui permet de décrire ces interactions. Il suffit d'écrire $obs(doc_1, Paul, visited)$ pour exprimer le fait que Paul a vu $doc_1$.

Le composant d'adaptation est composé de règles de la forme :
$\forall user\ \forall doc\ (premices \implies conclusion(doc, user, ...))$

Les prémisses de la règles peuvent être n'importe quelle formule de la logique des prédicats du premier ordre avec quantificateurs existentiels et universels, négation, conjonction et disjonction (et par conséquent l'implication et l'équivalence). La conclusion peut porter sur l'état d'apprentissage d'un document, ses annotations, etc.

Ainsi, si l'on veut définir qu'un document est recommandé à partir du moment où les documents pré-requis sont connus, il faut écrire :
$\forall user\ \forall doc_1$
$((\forall doc_2\ preq(doc_2, doc_1) \implies obs(doc_2, user, visited))$
$\implies learning\_state(doc_1, user, recommended)$

Cette définition logique des hypermédias adaptatifs pédagogiques a été utilisée pour caractériser un certain nombre de systèmes existant. Ainsi, NetCoach [68], ELM-ART II [13], Interbook [12] et KBS Hyperbook [34] ont été caractérisés en logique. Il apparaît que certaines caractéristiques de ces systèmes ne peuvent être capturées par le modèle proposé, comme par exemple certains calculs qui permettent de connaître le degré de connaissance de l'utilisateur sur un document. Néanmoins, cette

approche permet de comparer aisément les fonctionnalités des hypermédias adaptatifs pédagoriques, en utilisant un formalisme déclaratif et bien fondé. Cette caractérisation permet également de décrire la taxonomie des différents éléments des systèmes adaptatifs pédagogiques. Ce modèle n'a pas permis d'offrir une solution pour un langage de règle commun qui capturerait les fonctionnalités des différents langages de règles utilisés dans les systèmes. Enfin, cette caractérisation permet de mettre en exergue les méta-données nécessaires pour chaque système d'hypermédia adaptatif étudié.

## 2.7 Conclusions et problèmes posés

Dans ce chapitre, nous avons passé en revue un certain nombre de systèmes et de modèles d'hypermédias adaptatifs, et nous avons fait la synthèse de leur principaux composants.

Nous avons vu qu'il existait différentes approches pour la conception d'hypermédias adaptatifs, des plus génériques aux plus particulières. Ainsi, nous avons vu qu'historiquement, les premiers systèmes proposaient des démarches au cas par cas. Pour chaque système, les méthodes et techniques d'adaptation étaient choisies, puis mises en place. Bien que dans certains cas, ces systèmes auraient pu être réutilisés dans des domaines d'application très proches, ils l'ont rarement été, du fait notamment du manque de généricité des méthodes choisies, mais aussi de l'impossibilité de redéfinir des stratégies d'adaptation différentes, puisque celles-ci étaient codées "en dur" dans les systèmes.

Nous avons montré qu'un certain nombre de plate-formes ont réutilisé les idées développées dans ces premiers systèmes pour permettre la création de nouveaux hypermédias adaptatifs, en important ses propres données. Néanmoins, les stratégies d'adaptation restaient souvent inaccessibles.

Nous avons vu que, par la suite, des modèles beaucoup plus génériques ont été conçus pour faciliter la création de systèmes adaptatifs. Ces modèles laissent quelques libertés pour le domaine et l'utilisateur, données statiques des systèmes d'hypermédias adaptatifs, et permettent surtout de redéfinir les stratégies d'adaptation.

Nous avons constaté que la plupart des modèles proposés utilisent un découpage du système selon trois parties : le modèle de l'utilisateur, le modèle du domaine, et le modèle de l'adaptation. Nous avons étudié quelques systèmes qui utilisent une version affinée de ce découpage, en séparant, par exemple, le modèle du domaine en plusieurs couches.

Les modèles de l'utilisateur dans les hypermédias adaptatifs restent assez succincts : ils permettent de représenter n'importe quels attributs de l'utilisateur. Les modèles les plus récents assurent une distinction explicite entre connaissances de l'utilisateurs et attributs non-relatifs au domaine de l'application. On constate que, pour obtenir un modèle plus fin, destiné à un domaine d'application particulier, il faut accepter de perdre en généricité, et de fait, en facilité de réutilisation du modèle.

Les modèles de domaine dans les systèmes d'hypermédias adaptatifs offrent des façons très variées de représenter le domaine d'une application donnée. Ainsi, la notion de concept a-t-elle un sens très différent selon les systèmes. Certains l'utilisent comme une représentation abstraite d'éléments concrets (ou de conglomérats d'éléments concrets), d'autres séparent le niveau conceptuel

(sujets d'un domaine donné) du niveau matériel (ressources manipulées), et une ressource peut alors correspondre à un ou plusieurs concepts. Tout comme pour les modèle de l'utilisateur, un gain en précision sur les éléments représentés entraîne une perte de généricite. Par exemple, les modèles de domaines pédagogiques que nous avons présentés ne sauraient être utilisés pour un système adaptatif de gestion des connaissances !

Afin de répondre à ces problèmes de perte de généricité, nous avons proposé une approche, que nous utiliserons dans les chapitres 3 et 4, pour améliorer respectivement la conception de modèles de l'utilisateur et de modèles de domaine. Cette approche par du constat qu'il est très difficile de proposer un modèle suffisamment générique pour concevoir correctement n'importe quel système d'une manière acceptable, c'est-à-dire où les données ne sont pas répertoriées dans des catégories "fourre-tout", les rendant difficilement réutilisables par la suite. Il est également impossible de concevoir un modèle précis qui conviendrait pour tous les domaines d'application. Aussi proposons-nous un modèle générique pour l'utilisateur et un autre pour le domaine. Ces modèles génériques ont pour but de capturer les éléments communs aux modèles que nous avons pu rencontrer, ainsi que les relations qui les lient les uns aux autres. Ces modèles génériques n'ont pas pour but de permettre directement la création de systèmes implémentés. Ils doivent permettre de concevoir des modèles d'utilisateur et de domaine dans un contexte particulier, répondant à des besoins particuliers. Ainsi avons-nous déplacé le problème de perte de généricité contre perte de précision à un autre niveau : nos modèles génériques capturent la généricité, et permettent la création assistée de modèles précis.

Dans ce chapitre, nous avons également étudié la façon de mettre en place l'adaptation dans les systèmes d'hypermédias adaptatifs, et notamment l'utilisation de langages de règles pour parvenir à cette finalité. Nous avons également étudié certains travaux qui mettent en avant le caractère intrinsèquement logique des systèmes d'hypermédias adaptatifs, et notamment de l'adaptation. Ainsi existe-t-il différents langages de règles, avec des syntaxes plus ou moins complexes, et qui permettent de décrire l'adaptation à des niveaux plus ou moins précis. Certains systèmes donnent des règles pour calculer la désirabilité des documents au cas par cas, d'autres proposent de décrire des règles plus génériques, qui sont ensuite appliquées à un ensemble de documents répondant à un ou plusieurs critères donnés.

Nous avons pu remarquer que l'écriture de ces règles peut devenir assez complexe : les prémisses des règles permettent d'interroger différents éléments du système, comme les connaissances de l'utilisateur, ses préférences, les liens entre différents éléments du domaine. Ceci conduit à l'écriture de règles très complexes, ou à la limitation de la taille des règles, qui limite par conséquent les possibilités d'adaptation.

Nous avons également remarqué que les moteurs d'inférence utilisant ces langages de règles sont souvent décrits de manière informelle, et font l'objet d'une implémentation qui fixe le comportement adaptatif induit par les règles. La compréhension des mécanismes d'inférence, souvent complexes, n'en est rendu que plus obscur.

Enfin, nous avons constaté que la façon de faire des déductions sur les règles dans les langages étudiés est également implicite. Il est donc plus difficile de savoir précisément le sens des règles que l'on décrit, mais surtout, il est impossible de prouver que l'implémentation du langage de règle effectue bien les preuves correctement, puisque la façon de les effectuer n'est pas formellement décrite. En outre, il n'est pas envisagé de modifier les possibilités de raisonnement sur les langages de règle.

Afin de tenter de résoudre les trois problèmes concernant l'adaptation que nous venons de mettre en exergue, nous fournissons un modèle de l'adaptation basé sur le calcul des prédicats du premier ordre. Ce modèle va bien au-delà d'une simple caractérisation logique de l'adaptation, puisqu'il permet la création de systèmes adaptatifs, dans un cadre formel. Les formalismes proposés sont tous basés sur des formalismes standards. Ainsi le moteur d'adaptation est-il basé sur la logique situationnelle, un formalisme logique basé sur le calcul des prédicats du premier ordre, et dont la sémantique est connue.

Nous avons également séparé les différents niveaux d'adaptation, liés au domaine, aux préférences de l'utilisateur et à ses connaissances, en donnant trois couches distinctes à notre langage de règle. La première couche est constituée de règles de type "prémisses impliquent conclusion", qui cherchent un parcours dans le domaine, sans tenir compte d'autres éléments que les relations du domaine. Afin de tenir compte des préférences de l'utilisateur, nous avons introduit un système de méta-règles, qui permet de sélectionner un ensemble de règle de parcours du domaine, correspondant au mieux à l'utilisateur. Comme nous le verrons dans le chapitre 5, ces méta-règles sont très déclaratives, facilitant ainsi leur conception. Enfin, nous avons proposé un système de déduction formel, qui permet à la fois de définir de quelle façon prendre en compte les connaissances de l'utilisateur, d'une part, et, d'autre part, de donner une méthode formelle pour savoir comment raisonner sur les règles.

Afin d'étayer notre propos, nous proposerons un exemple d'application, construite à partir de nos modèles génériques du domaine et de l'utilisateur, et de notre formalisme logique d'adaptation. Cet exemple est proposé dans le chapitre 6.

# Chapitre 3

# Modélisation générique de l'utilisateur

Dans ce chapitre, nous présentons un modèle de l'utilisateur pour les hypermédias adaptatifs destinés au e-learning, ainsi qu'un modèle générique de l'utilisateur pour les hypermédias adaptatifs.

Le modèle pour le e-learning apporte, par rapport aux modèles existant, une catégorisation plus détaillée des attributs de l'utilisateur. Bien que restant très simple, il permet d'affiner la classification des données qui pourront être utilisées lors de l'adaptation. Ce modèle est donné en UML, par soucis de standardisation. Ce modèle sera utilisé dans le chapitre 6, pour la construction de notre exemple.

Le modèle générique est construit à partir de modèles de l'utilisateur existant. Il représente l'ensemble des éléments communs à la plupart des modèles de l'utilisateur dans les hypermédias adaptatifs. Du point de vue de l'UML, ce modèle est une généralisation des modèles existant, comme nous le montrerons à travers quelques exemples. Le but de ce modèle est, d'une part, de montrer qu'il existe un fond commun entre tous les modèles existant, et, d'autre part, de donner une base à respecter pour la création de modèles spécifiques de l'utilisateur.

Ce chapitre s'articule donc selon trois axes : tout d'abord nous présentons notre modèle de l'utilisateur, puis notre modèle générique et enfin nous montrons la pertinence de ce modèle à travers des exemples.

## 3.1 Un modèle de l'utilisateur pour le e-learning

Dans cette section, nous allons détailler notre modèle de l'utilisateur. Dans ce modèle, nous voulons être capables de :

- Représenter un certain nombre de caractéristiques pré-définies, indispensables et communes à tous les utilisateurs, comme par exemple son nom et son prénom ;
- Catégoriser des attributs non-prédéfinis selon différents axes, afin de faciliter l'utilisation des

données.

– Retenir les documents parcourus de manière générale et spécifiquement pour l'acquisition de chaque concept.

La représentation en UML de notre modèle de l'utilisateur est donné sur la figure 3.1.

### 3.1.1 Responsabilité des classes

Voici le détail des responsabilités des classes UML présentées sur la figure 3.1.

– La classe GestionnaireUtilisateur est chargée de servir d'interface avec les autres composants du système d'hypermédia adaptatif. A cet effet, elle est munie des méthodes Ask et Tell qui permettent de poser des questions et d'apporter des réponses aux composants externes (domaine, adaptation). Elle est composée de l'ensemble des utilisateurs, qu'elle est chargée de gérer.

– La classe Utilisateur est chargée de représenter les informations concernant un utilisateur particulier. Elle est composée d'attributs prédéfinis, comme le nom, l'identifiant ou le mot de passe de l'utilisateur.

– La classe AttributPréférence est chargée de représenter différentes préférences de l'utilisateur. Les préférences de l'utilisateur sont ses préférences d'affichage : taille des caractères, problèmes de couleurs, de contraste etc. et ses préférences de présentation : préfère les éléments textuels ou graphiques, ne veux pas d'élément audio etc.

– La classe AttributBackground est chargée de représenter les attributs de l'utilisateur relatif à son parcours scolaire/professionel.

– La classe Stéréotype est chargée de représenter les différentes catégories de stéréotypes auxquels appartient l'utilisateur. Un stéréotype est constitué d'un nom et d'un valeur. La valeur peut-être choisie parmi un nombre très réduit de possibilité, afin de créer des catégories d'utilisateur regroupant de nombreux utilisateurs. La valeur d'un stéréotype est calculée, en fonction d'attributs de l'utilisateur.

– La classe AutresAttributs est chargée de représenter les attributs de l'utilisateur qui ne sont pas relatifs à son parcours et qui ne sont pas des préférences. Par exemple, une clé de cryptage de données. Le but de cette classe est notamment d'assurer la compatibilité de notre modèle avec des modèles standards type IMS Learner ou PAPI Learner (par conversion).

– La classe d'association Degré est chargée de donner une valeur à la connaissance d'un concept, parmi : très bas, bas, moyen, bon, excellent. Cette échelle de valeur est un choix, elle permet d'avoir une bonne précision par rapport à une classification binaire, et évite un degré de précision trop élevé et donc peu utile à l'adaptation. Nous verrons que le modèle générique permet de définir n'importe quelle échelle de degrés de connaissances.

– La classe Historique est chargée de représenter un historique de documents parcourus, de permettre de donner la date de parcours du document, de parcourir l'historique dans l'ordre de

parcours (méthodes DocumentSuivant() et DocumentPrécédent()). La classe historique peut-être utilisée pour représenter l'historique de tous les documents parcourus par un utilisateur, ou pour représenter l'historique des documents parcourus pour parvenir à un degré de connaissance pour un concept donné.

– Les classes Document et Concept sont détaillées dans le modèle du domaine.

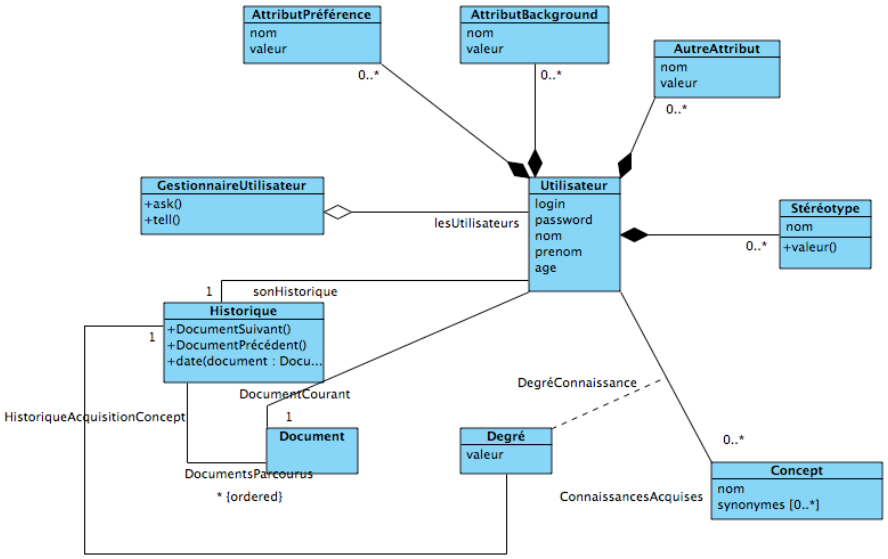


FIG. 3.1 – Modèle, en UML, de l'utilisateur

### 3.1.2 Justification des choix

La plupart des modèles utilisent une seule classe pour représenter les attributs de l'utilisateur. Nous avons pourtant choisi de faire intervenir les classes AttributPréférence, AttributBackground, AutreAttribut et stéréotype. Les préférences de l'utilisateur, son parcours et les stéréotypes auxquels il appartient sont trois catégories d'attributs à la sémantique très différente. Ce sont ces catégories qui reviennent le plus souvent dans les hypermédias adaptatifs à but pédagogique. Ne pas séparer ces catégories dans le modèle rend la conception d'un hypermédias adaptatifs plus difficile : les données ne sont alors pas triées ; l'adaptation est plus difficile : les préférences, liées à la présentation, sont au même niveau que d'autres attributs, qui ont quant à eux une importance pour la sélection de documents, et les stéréotypes sont au même niveau que les détails du profil. Dans notre modèle, la classe AutreAttribut permet en outre de fournir d'autres attributs indépendants du domaine qui n'appartiennent à aucune des catégories mentionnées ci-dessus si nécessaire.

Contrairement aux autres classes d'attributs, la valeur des stéréotypes est calculée. Ces valeurs sont calculées en fonction d'autres attributs. Les stéréotypes permettent de donner des catégories

simples et regroupant de nombreux utilisateurs dans une même catégorie. En fonction des mises-à-jour des autres attributs, les stéréotypes doivent être recalculés : il faut donc donner la méthode pour les calculs en question.

Contrairement à certains modèles nous avons choisi de faire intervenir la notion d'historique. De plus, nous avons rajouté la notion d'historique lié à un concept. La notion d'historique nous paraît porteuse d'intérêt : elle doit permettre à l'utilisateur de parcourir à nouveau des documents déjà parcourus, composés des mêmes ressources atomiques que lors du premier visionnage. C'est d'ailleurs pourquoi notre classe historique stocke les documents parcourus et non pas les concepts. Concernant les historiques relatifs à un concept, ils contiennent les documents parcourus ayant permis l'acquisition d'un degré de connaissance pour un concept donné. Le but de ces historiques est de permettre à un utilisateur de parcourir à nouveau les documents lui ayant permis d'acquérir un concept qu'il souhaite revoir.

## 3.2 Modèle générique de l'utilisateur

En nous appuyant sur les différents modèles de l'utilisateur pour les hypermédias adaptatifs que nous avons rencontré, nous avons cherché à généraliser ces modèles afin de fournir un fond commun pour tous les modèles, permettant à la fois d'extraire les éléments essentiels des modèles et de créer de nouveaux modèles à partir du modèle générique.

Le diagramme du modèle générique de l'utilisateur que nous avons réalisé est présenté sur la figure 3.2. Ce modèle générique permet de représenter et gérer des utilisateurs, de créer des attributs optionnels, et prendre en compte les connaissances que l'utilisateur a du domaine et de gérer les historiques. Ainsi, les éléments les plus courants des modèles sont pris en compte dans notre modèle générique. Toutes les classes de ce modèle sont abstraites.

### 3.2.1 Responsabilité des classes

Nous détaillons ici les différentes classes de notre modèle générique, ainsi que leurs responsabilités respectives.

- La classe Interface est chargée d'administrer les données sur les utilisateurs et de servir d'interface avec les autres éléments du système.
- La classe RepresentationUtilisateur est chargée de représenter un utilisateur.
- La classe AttributPrédéfini est chargée de représenter les attributs prédéfinis pour les utilisateurs, i.e. ceux qui doivent toujours être renseignés.
- La classe AttributNonPrédéfini est chargée de représenter les différents attributs indépendants du domaine non prédéfinis pour les utilisateurs. Ces attributs sont distingués les uns des autres par leur nom.
- La classe AttributConnaissance est chargée de représenter la valeur associée à la connaissance

d'un concept.

– La classe RepresentationHistorique est chargée de représenter un historique des actions de l'utilisateur ou des documents parcourus etc.
– La classe Abstraction est détaillée dans le modèle générique du domaine

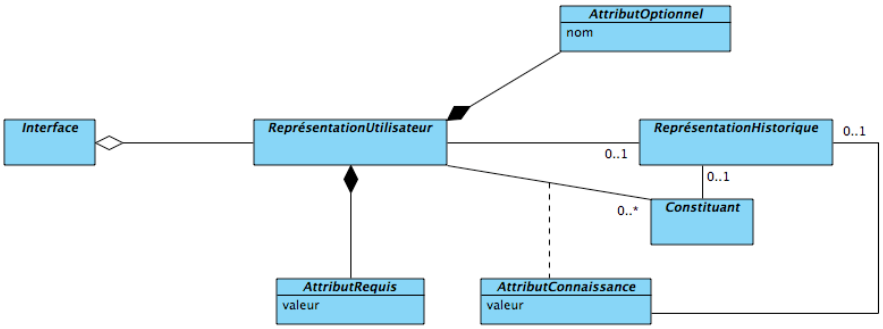


FIG. 3.2 – Modèle générique de l'utilisateur, en UML

### 3.2.2 Justification des choix

Le modèle générique généralise les différentes catégories d'éléments que l'on trouve non seulement dans notre modèle, mais également dans d'autres modèles, comme nous le montrerons dans la section 3.4.

L'interface est un composant chargé de gérer les utilisateurs et de communiquer avec les autres modules : contrairement à la classe GestionnaireUtilisateur du modèle de base, la classe interface ne détaille pas la façon de communiquer avec les autres agents.

La classe RepresentationUtilisateur est une abstraction de la classe - toujours présente - chargée de représenter les utilisateurs. Elle contient des attributs prédéfinis, à préciser.

Les attributs non prédéfinis appartiennent, dans le modèle générique, à la classe AttributNonPrédéfini, qui peut avoir autant de sous-classes concrètes que nécessaire pour catégoriser les différentes sortes d'attributs non prédéfinis.

### 3.2.3 Contraintes sur la spécialisation du modèle générique

Nous détaillons ici les différentes contraintes que nous imposons pour la spécialisation de notre modèle générique. Ainsi, certaines classes peuvent être spécialisées plusieurs fois, d'autres doivent l'être au moins une fois, certaines sont facultatives etc. :

– **Interface** : Doit posséder au moins une sous-classe dans le modèle spécialisé. Il n'y a pas de limite du nombre de sous-classe car on peut vouloir utiliser différentes interfaces pour communiquer avec des composants de différentes natures.
– **RepresentationUtilisateur** : Doit posséder au moins une sous-classe. Il n'y a pas d'autre contrainte car il faut pouvoir représenter des utilisateurs de sortes potentiellement différentes.

- **AttributPrédéfini** : Doit posséder au moins une sous classe, la profondeur des sous-classesest limitée à 1. En effet, il faut au moins un attribut prédéfini (identifiant de l'utilisateur), et on ne peut pas définir des sous-catégories d'un type d'attribut prédéfini, puisqu'ils ont le même rôle que des attributs (au sens UML) de la classe RepresentationUtilisateur.
- **AttributNonPrédéfini** : Doit posséder au moins une sous-classe. Il faut au moins une classe pour représenter les attributs indépendants du domaine, mais il est possible d'en avoir plusieurs, de manière hiérarchisée.
- **RepresentationHistorique** : pas de contrainte, puisque la notion d'historique n'est pas utilisée par tous. Il peut y avoir plusieurs sortes d'historiques, donc on ne contraint pas le nombre de sous-classes.
- **AttributConnaissance** : Doit posséder au moins une sous-classe. Il faut au moins une façon d'annoter les connaissances. Il peut y en avoir plusieurs.

Nous reviendrons dans le chapitre 4 sur la classe constituant, qui appartient au domaine.

## 3.3 Lien entre modèle pour le e-learning et modèle générique

Dans cette section, nous montrons que notre modèle d'utilisateur pour le e-learning est une spécialisation de notre modèle générique. Le but de cette démarche est de donner un exemple de spécialisation du modèle générique, et de montrer que notre modèle pour le e-learning suit bien la structure de notre modèle générique.

Afin de faire le lien entre notre modèle générique et notre modèle pour le e-learning, nous modifions légèrement le diagramme représentant notre modèle, afin d'extraire les attributs UML - prédéfinis - de la classe Utilisateur. Ils sont désormais composés dans la classe utilisateur. Le but de cette opération est de pouvoir utiliser le concept abstrait d'**attribut prédéfini**. Le modèle ainsi modifié est présenté sur la figure 3.3. La sémantique de ce modèle n'est pas modifiée, la transformation est purement formelle.

La figure 3.4 montre comment les éléments du modèle donné figure 3.3 sont des spécialisations des éléments du modèle générique donné figure 3.2.

Les classes AttributPréférence, AttributBackground, Stéréotype et AutreAttribut héritent tous de la classe AttributNonPrédéfini. Ils s'agit bien de différentes catégories d'attributs non prédéfinis. Les attributs login, password, nom, prénom et âge sont des sous-classes de AttributPrédéfini. La seule interface est la classe GestionnaireUtilisateur. La classe Utilisateur hérite de la classe RepresentationUtilisateur. La classe Degré hérite de la classe AttributConnaissance. La classe historique hérite de la classe RepresentationHistorique.

Nous avons donc vérifié que notre modèle de l'utilisateur pour le e-learning est bien une spécialisation de notre modèle générique. Cela signifie que notre modèle pour le e-learning reprend bien les éléments caractéristiques des modélisations de l'utilisateur dans les hypermédias adaptatifs.

Nous allons montrer dans la section suivante que les éléments de notre modèle générique sont bien caractéristiques des modélisations de l'utilisateur dans les hypermédias adaptatifs.

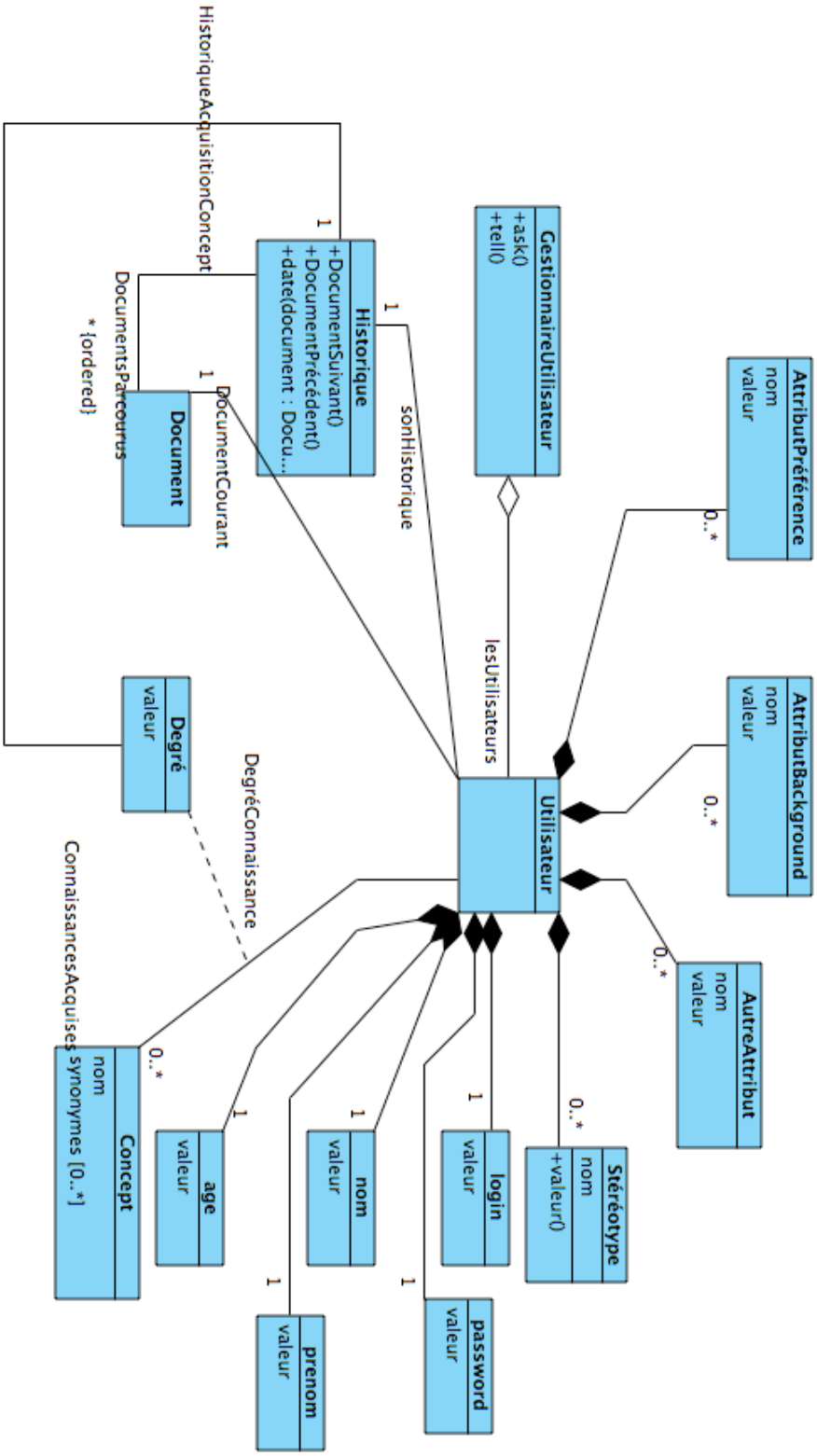


FIG. 3.3 – Modèle de l'utilisateur modifié, en UML

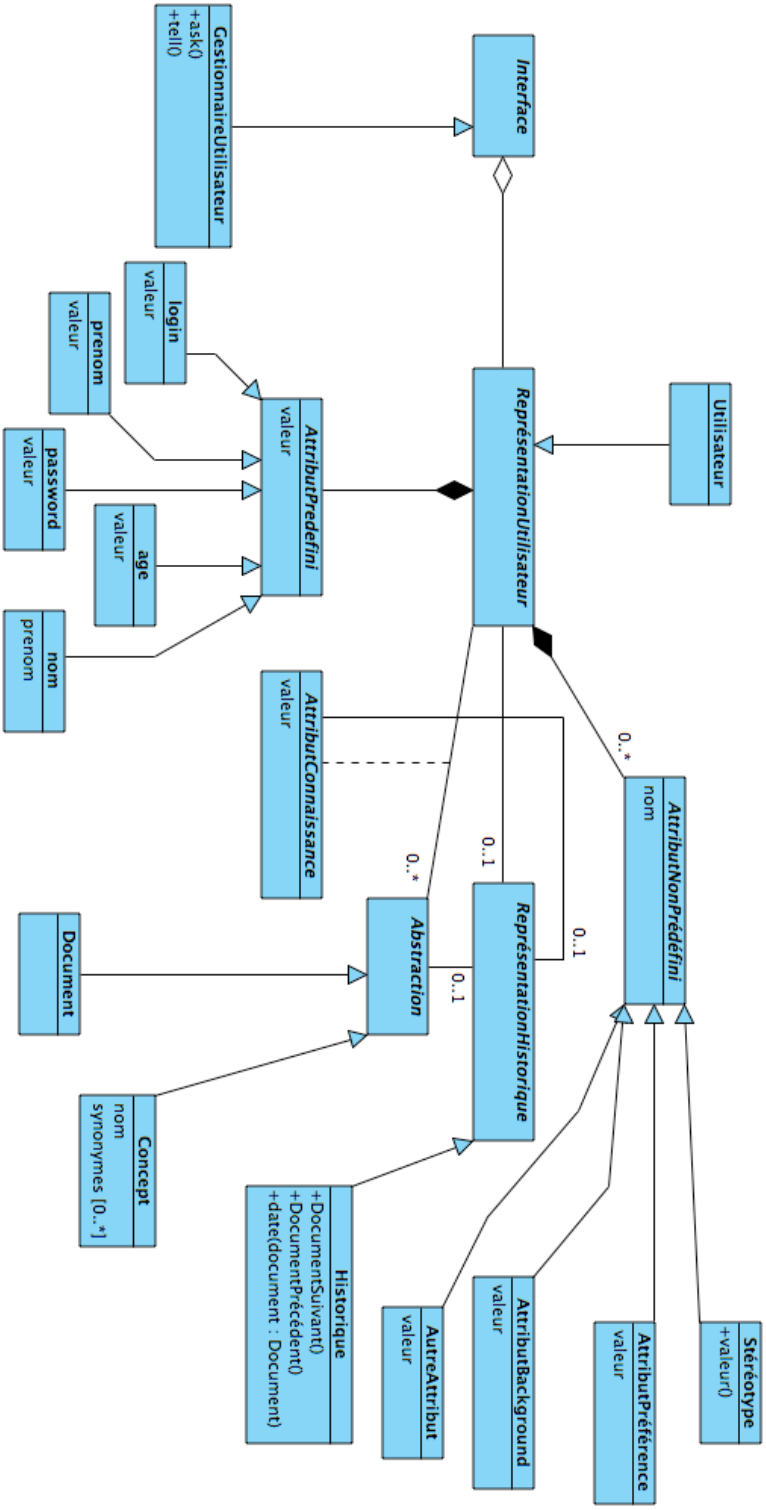


FIG. 3.4 – Spécialisation du modèle générique pour obtenir le modèle de l'utilisateur

## 3.4 Modèle générique et modèles existant

Dans cette section, nous montrons que notre modèle générique est bien une généralisation de modèles connus, notamment du Munich Model et du modèle de l'utilisateur de AHAM, justifiant ainsi de sa conception et du bien-fondé de son utilité pour concevoir de nouveaux modèles.

### 3.4.1 Mise en relation du modèle générique avec le modèle de l'utilisateur du Munich Model

Le modèle de l'utilisateur du Munich Model est présenté sur la figure 3.5. Dans un premier temps, nous avons transformé la représentation en UML du Munich Model en une autre représentation UML, équivalente du point de vue du sens, mais où les éléments sont organisés de manière compatible avec notre modèle générique.

Le résultat de cette transformation est donné sur la figure 3.6. Les attributs username et email de la classe User ont été extraits de cette classe, tout comme dans notre modèle pour le e-learning ; la notion de valeur d'un attribut a été intégré à la classe UserAttribute, et la classe DependantAttribute est désormais classe d'association de la relation entre les utilisateurs et les composants du domaine. De plus ce sont les éléments de la classe IndependantAttribute qui sont désormais agrégés dans la classe User, et non pas ceux de la classe UserAttribute.

Justifications :

- L'extraction des attributs de User est purement formelle, elle permet d'en faire une généralisation.
- Intégrer un attribut value à UserAttribute plutôt que de garder un lien vers UserAttrValue est également un changement formel :
  - Concernant les attributs indépendants du domaine la classe UserAttrValue est, dans le modèle d'origine, toujours rattachée à un attribut indépendant du domaine, il est donc équivalent de rajouter un attribut value à la classe IndependantAttr ;
  - Concernant les attributs dépendant du domaine, la valeur est classe d'association de la relation entre DependantAttr et Component. Le fait de factoriser value a pour conséquence, potentiellement, de multiplier le nombre d'attributs dépendants du domaine, mais ne modifie pas le sens de l'ensemble : il est toujours possible de fournir des attributs et leurs valeurs associées entre l'utilisateur et le domaine.
- La création d'une association entre User et Component avec DependentAttr comme classe d'association permet de représenter la même information que la relation de composition entre UserAttribute et User, il ne s'agit donc que d'une redondance d'information, sans préjudice ;
- Le changement d'aggrégation des attributs (on n'agrège plus que les attributs indépendants du domaine) est lié aux remarques du point précédent : l'association entre User et Component, munie de sa classe d'association, suffit à emmagasiner l'information de ce qui lie l'utilisateur

au domaine.

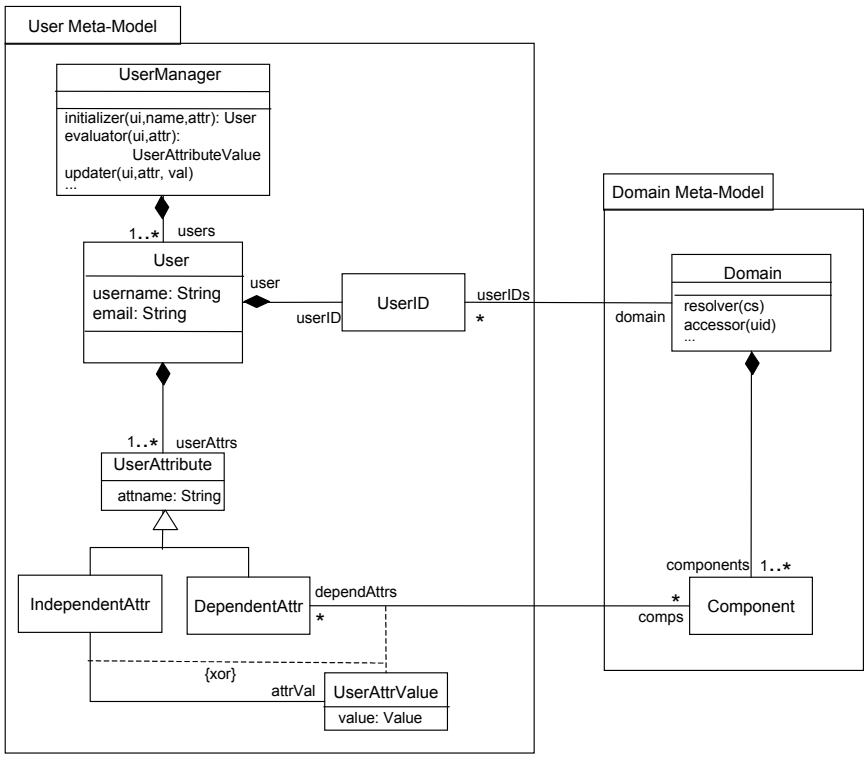


FIG. 3.5 – Modèle de l'utilisateur, en UML, dans le Munich Model

La figure 3.7 présente les liens de spécialisation entre notre modèle générique et la partie utilisateur du Munich Model. Cette figure montre que la partie utilisateur du Munich Model est bien une spécialisation de notre modèle générique.

La partie utilisateur du Munich Model présente quelques différences avec notre modèle pour le e-learning :

- Le Munich Model ne permet pas de conserver la trace des documents parcourus : il n'y a pas d'historique.
- Le Munich Modèle a une seule catégorie d'attributs indépendants du domaine, nous en avons plusieurs, afin de pouvoir raisonner différemment sur différentes sortes d'attributs.
- Les attributs liés à la connaissance d'un concept dans notre domaine sont simplement valués, dans le Munich Model ils sont nommés, et il peut donc y avoir plusieurs sortes d'attribut liant l'utilisateur à un concept. Ces différentes possibilités ne sont pas détaillées dans le Munich Model.

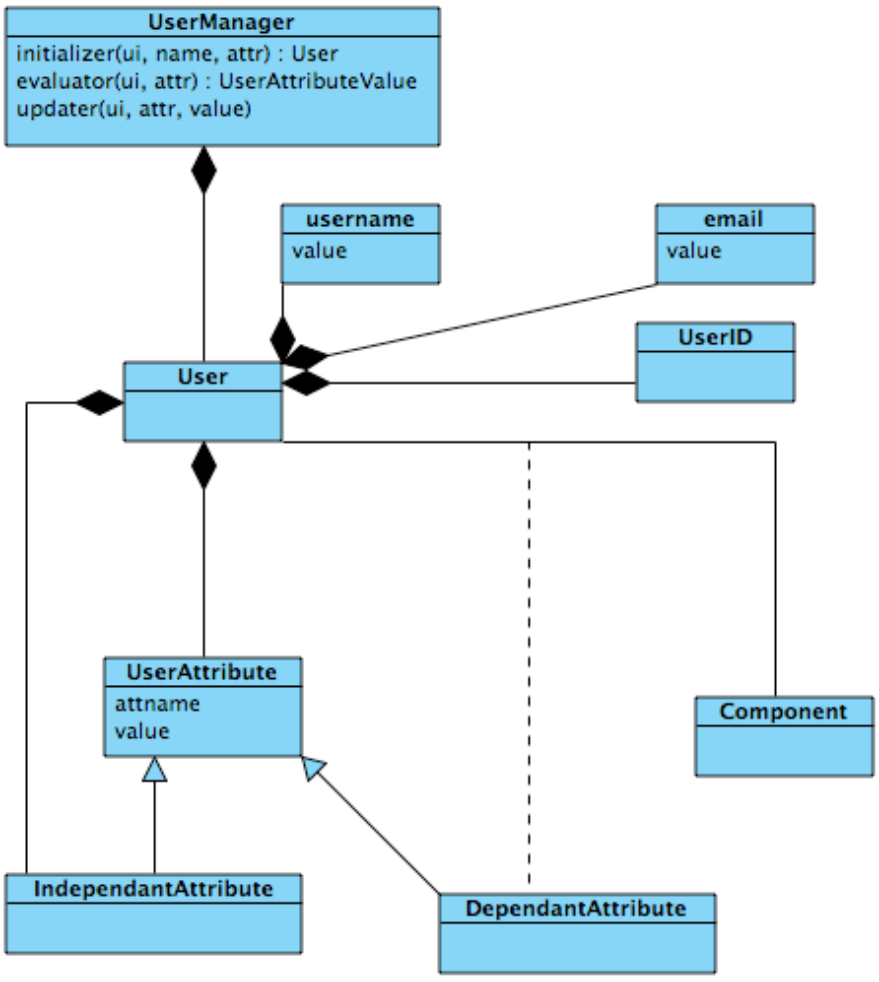


FIG. 3.6 – Modèle de l'utilisateur, en UML, dans le Munich Model, reformulé

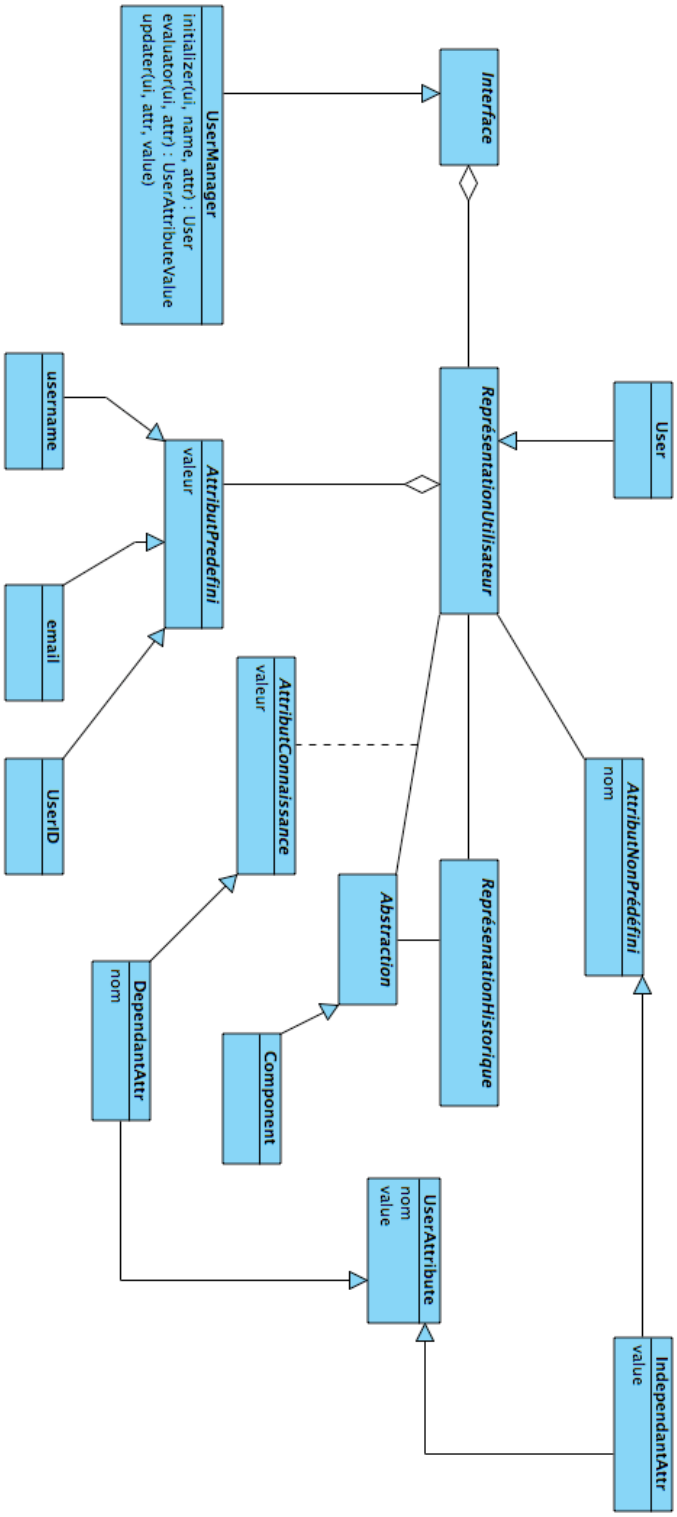


FIG. 3.7 – Modèle de l'utilisateur du Munich Model et modèle générique de l'utilisateur, en UML

### 3.4.2 Mise en relation du modèle générique avec le modèle de l'utilisateur de AHAM

Le modèle de l'utilisateur de AHAM est relativement peu formalisé, et il reste très général. Dans AHAM, le modèle de l'utilisateur est constitué d'entités, qui correspondent le plus souvent à des concepts (au sens de AHAM). Pour chaque entité un certain nombre de paires attribut-valeur sont stockées, comme, par exemple, le fait qu'un concept a été lu ou le niveau de connaissance d'un concept. Les données relatives aux utilisateurs sont stockées dans une structure de table. La façon de gérer les utilisateurs est implicite. Il n'y a pas d'historique. Il n'y a pas d'autre attribut prédéfini que l'identifiant de l'utilisateur (implicite).

Nous retrouvons donc, dans le modèle de l'utilisateur de AHAM :

- Une représentation de l'utilisateur, qui est identifié ;
- Un certain nombre d'attributs non prédéfinis, correspondant dans AHAM à des entités. Certains dépendent du domaine, d'autres pas. Chaque entité peut contenir un certain nombre de paire attribut-valeur, là où nos attributs sont valués simplement.
- Une interface de communication avec les éléments extérieurs, qui prend la forme d'une table dans une base de données.

On peut donc formaliser facilement ce modèle comme une spécialisation de notre modèle générique. L'utilisateur aura des entités indépendantes du domaine spécialisant AttributNonPrédéfini, des entités liées au domaine (connaissances) spécialisant la classe AttributConnaissance, un seul attribut prédéfini (identifiant), et une interface - implicite dans AHAM - gérera les utilisateurs.

Nous représentons le modèle de l'utilisateur de AHAM sur la figure 3.8. Sa forme, très similaire à celle du modèle de l'utilisateur du Munich Model [52] - ils dérivent tout deux de dexter [29] - permet facilement de montrer qu'il est bien une spécialisation - très simple - de notre modèle générique.

### 3.4.3 Système de Duitama

Dans sa récente thèse, J.F. Duitama décrit le modèle de l'apprenant de la façon suivante :

Le modèle de l'utilisateur est un triplet :

$$UM =< learner, preference, knowledge >$$

$learner$ est l'identifiant de l'utilisateur. Les préférences de l'utilisateur correspondent à sa langue ou à sa présentation préférée. Chaque élément de l'ensemble des préférences est représenté par un couple $< attribute, value >$. Les connaissances de l'utilisateur sont représentées par des triplets de la forme : $< domainConcept, role, educationalState >$.

Là encore, les connaissances sont séparées des attributs indépendants du domaine. Les attributs indépendants du domaine sont donnés par un couple nom/valeur. Les attributs de connaissance possèdent une valeur (educational-state) et un rôle, correspondant à la nature de la connaissance acquise

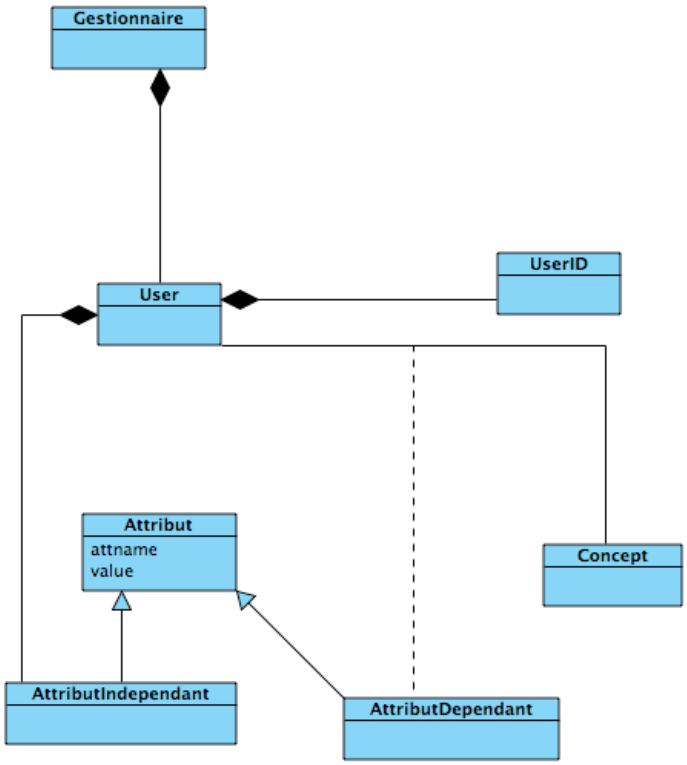


FIG. 3.8 – Modèle de l'utilisateur de AHAM, en UML

pour le concept du domaine (connaissances pratiques, théoriques, histoire du concept, introduction etc.)

On peut donc facilement construire une spécialisation de notre modèle générique qui représente le modèle de l'utilisateur de Duitama. Par exemple, la sous-classe de AttributConnaissance aura, en plus de l'attribut par défaut (valeur) un attribut rôle. La seule sous-classe de AttributPredefini sera un identifiant, correspondant au learner du triplet représentant l'utilisateur.

Nous représentons le modèle de l'utilisateur de ce système sur la figure 3.9. Ce modèle, qui est très simple, est bien une spécialisation de notre modèle générique.

### 3.4.4 Conclusions sur les liens avec les autres modèles

Dans cette section, nous avons vu comment nous pouvions montrer que notre modèle générique est une généralisation d'autres modèles, modulo, parfois, quelques modifications de forme. Ces modifications de forme sont acceptables dans la mesure où notre modèle générique n'est pas un modèle de conception, mais un modèle d'analyse : il n'a donc pas pour but de capturer les spécificités d'implémentation de chaque modèle existant.

Le Munich Model et AHAM sont également des modèles assez génériques. Ils sont donc des spécialisations assez peu spécifiques de notre modèle générique. Notre but ici n'était pas de montrer que nous avons un modèle beaucoup plus générique que ces modèles, mais de montrer la compa-

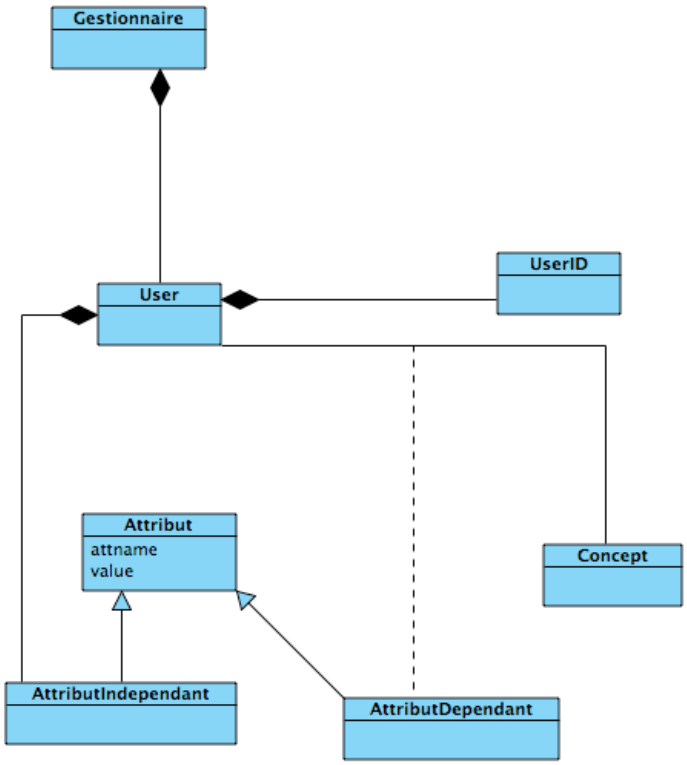


FIG. 3.9 – Modèle de l'utilisateur de Duitama, en UML

tibilité des modèles. Ainsi, il est légitime de proposer d'utiliser notre modèle générique pour créer des modèles spécifiques à un domaine d'application particulier, puisque ce modèle générique est compatible avec les modèles les plus connus.

## 3.5 Conclusions

Dans ce chapitre, nous avons présenté notre modèle de l'utilisateur pour le e-learning. Tout comme la plupart des modèles de l'utilisateur conçus pour les hypermédias adaptatifs, ce modèle reste assez simple. Il offre néanmoins une catégorisation beaucoup plus fine des différents attributs que peut posséder un utilisateur. Cette catégorisation doit permettre de faciliter la création et surtout la maintenance des attributs en question dans le système. Ce modèle donne également une importance assez grande à la notion d'historique, que nous estimons importante surtout dans le cadre du e-learning : pouvoir revoir les mêmes documents, composés de la même façon, lorsque l'on souhaite réviser un cours, offre une appréciable consistance au système. De plus, le fait de pouvoir s'intéresser à la ré-acquisition d'un concept en particulier est un apport pour l'utilisateur.

Nous avons également présenté un modèle générique pour capturer les points communs entre la plupart des modèles de l'utilisateur dans les hypermédias adpatatifs et pour permettre de créer de nouveaux modèles spécifiques à un type d'hypermédia en particulier. À travers quelques exemples,

nous avons prouvé que ce modèle générique est bien une généralisation de modèles connus. Notre modèle de l'utilisateur pour le e-learning est également une spécialisation de ce modèle générique. Outre la conception de nouveaux modèles de l'utilisateur, ce modèle générique doit permettre de faciliter l'importation de données provenant d'autres systèmes. La forme assez simple, en général, des modèles de l'utilisateur dans les hypermédias adaptatifs doit simplifier l'opération d'importation, via notre modèle générique, vers un autre modèle dérivé de ce modèle générique, de données concernant des utilisateur.

# Chapitre 4

# Modélisation générique du domaine

Dans ce chapitre, nous présentons deux éléments concernant la modélisation de l'utilisateur pour les hypermédias adaptatifs.

Tout d'abord, nous avons construit un modèle de domaine pour le e-learning. Bien qu'il existe déjà des modèles de domaine pour les hypermédias adaptatifs d'une part, et des modèles de domaine pour le e-learning d'autre part, notre modèle, en plus d'être à la fois conçu pour les hypermédias adaptatifs et pour le e-learning, apporte une séparation entre concepts (thesaurus du domaine) et documents qui est rarement modélisée explicitement dans les hypermédias adaptatifs - quand elle existe. Ce modèle est donné en UML, dans un souci d'utilisation des standards. Ce modèle sera notamment utilisé dans notre exemple, chapitre 6.

D'autre part, nous avons étudié un certain nombre de modèles formels de domaine dans les hypermédias adaptatifs, et nous en avons extrait un modèle générique pour la représentation des domaines dans les hypermédias adaptatifs. Du point de vue de l'UML, ce modèle générique est une généralisation des modèles que nous avons rencontrés, comme nous le montrerons sur quelques exemples, et notamment sur le Munich Model (partie domaine). Le but de ce modèle générique est double. Tout d'abord montrer que malgré leurs apparentes différences, les différents modèles existant ont une partie commune non négligeable. D'autre part, étant donné qu'on ne saurait parvenir à un modèle unifié pour tous les hypermédias adaptatifs (différence de nature des domaines, types d'hypermédias différents etc.), notre modèle générique fournit une base pour la création d'un modèle spécialisé.

Dans ce chapitre, nous allons donc présenter, dans un premier temps, notre modèle pour le e-learning, détailler ses spécificités et la façon dont on peut raisonner avec. Puis nous présenterons notre modèle générique avant de montrer qu'il est bien une généralisation de modèles du domaine pour les hypermédias adaptatifs.

## 4.1 Modèle du domaine

Dans cette section, nous présentons notre modèle de domaine pour le e-learning. Ce modèle a pour but de posséder les caractéristiques suivantes :

- Représenter de manière clairement séparée, d'une part, le thésaurus du domaine, c'est-à-dire l'ensemble des concepts qui le composent, et, d'autre part, les ressources utilisées pour l'apprentissage sur le domaine en question.
- Donner des relations fixées entre les différents constituants (concepts, ressources) et clairement définies. Le problème de l'extension des relations, c'est-à-dire de l'ajout de nouvelles relations dans le modèle, sera traité dans la partie modèle générique du domaine.
- Définir des catégories de ressources permettant de mieux les sélectionner et les composer dans les documents présentés à l'utilisateur.

Notre modèle du domaine pour le e-learning est représenté en UML sur la figure 4.1

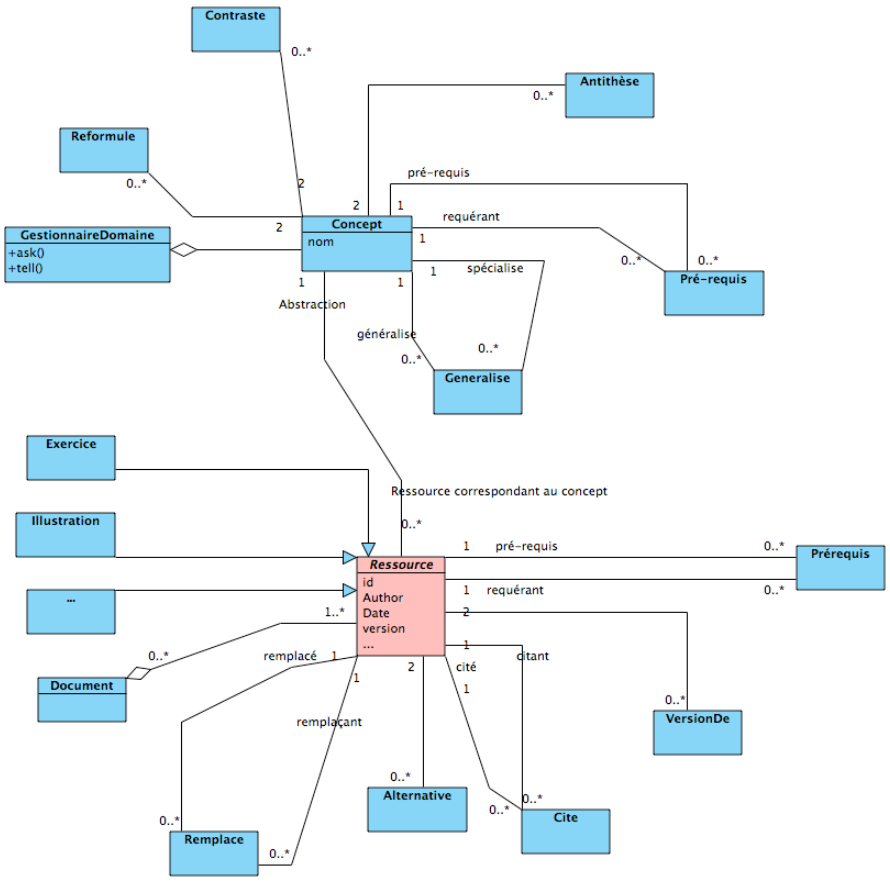


FIG. 4.1 – Modèle du domaine, en UML

### 4.1.1 Responsabilité des classes

Nous détaillons ici les différents éléments constituant notre modèle de domaine pour le e-learning, relations mises à part (cf. 4.1.2).

- Un *concept* représente une idée ou un ensemble d'idées abstraites. Un concept est représenté par son nom, qui décrit un thème particulier du domaine.
- La classe *Ressource* est chargée de représenter les différentes ressources disponibles sur le domaine. Les ressources sont atomiques, elles correspondent à un seul concept et sont d'une seule sorte (*Exercice, Définition, Illustration*...).
- La classe *document* est chargée de représenter les documents que l'on peut présenter à un utilisateur, c'est-à-dire les agrégats ordonnés de ressources. Les documents sont donc composées à partir de ressources ordonnées, ce qui doit faciliter une adaptation de présentation et de contenu.
- La classe *GestionnaireDomaine* est chargée de gérer le domaine et de servir d'interface entre le domaine et les autres composants de l'HA.

Nous avons bien séparé concepts et ressources en deux niveaux distincts. Ces deux niveaux sont liés par la relation d'abstraction, qui permet de lier une ou plusieurs ressources atomiques à un unique concept. Comme nous le verrons dans la sous-section suivante, les relations entre concepts diffèrent des relations entre ressources : par exemple, la notion de connaissance porte sur les concepts (cf. chapitre 3), ce qui entraîne nécessairement une relation de pré-requis forte entre les concepts.

### 4.1.2 Définition des relations

Dans cette sous-section, nous allons détailler les différentes relations que nous avons sélectionné pour notre modèle du domaine pour le e-learning. Nous souhaitons fournir deux types d'information pour chaque relation. Tout d'abord, nous souhaitons en donner une sémantique, sous forme d'une définition, ce qui est absolument indispensable à la compréhension précise de la nature de ces relations. Nous souhaitons également définir les propriétés mathématiques de ces relations : sont-elles symétriques, transitives etc. ?

Nous souhaitons décrire ces propriétés mathématiques afin que le gestionnaire de domaine soit lui même capable de calculer si une relation existe entre deux éléments quand elle n'a pas été donnée explicitement. Si on a une relation $A\Re B$ entre deux concepts $A$ et $B$ et que la relation $\Re$ est symétrique, le gestionnaire de domaine doit répondre "**oui**" à la question "**A-t-on** $B\Re A$ **?**"

#### Propriétés possibles des relations binaires

Nous rappelons ici succinctement les différentes sortes de relations binaires que l'on peut rencontrer :

- Propriétés liées à la réflexivité :
  - Relations réflexives (R) : tout élément est en relation avec lui même ;

  - Relations irréflexives (IR) : aucun élément n'est en relation avec lui même ;
  - Relations aréflexives (AR) : relations ni réflexives, ni irréflexives.
- Propriétés liées à la symétrie :
  - Relations symétriques (S) : si x est en relation avec y, alors y est en relation avec x ;
  - Relations antisymétriques (AnS) : si x est en relation avec y et si y est en relation avec x alors x=y ;
  - Relations asymétriques (AS) : si x est en relation avec y, alors y n'est pas en relation avec x ;
  - Relations isolantes (I) : tout élément ne peut être en relation qu'avec lui-même ;
  - Relations dissymétriques (DS) : relations ni symétrique ni antisymétrique.
- Propriétés liées à la transitivité :
  - Relations transitives (T) : si x est en relation avec y, et si y est en relation avec z, alors x est en relation avec z ;
  - Relations circulaires (C) : si x est en relation avec y, et si y est en relation avec z, alors z est en relation avec x ;
  - Relations antitransitives (AT) : si x est en relation avec y, et si y est en relation avec z, alors x n'est pas en relation avec z ;
  - Relations anticirculaires (AC) : si x est en relation avec y, et si y est en relation avec z, alors z n'est pas en relation avec x.
- Autres propriétés :
  - Relations connexes (Co) : pour tout x et y, soit x est en relation avec y, soit y est en relation avec x, soit x est égal à y ;
  - Relations totales (To) : pour tout x et y, soit x est en relation avec y, soit y est en relation avec x.
- Relations d'ordre (Or) : relations réflexives, transitives et antisymétriques ;
- Relations d'équivalence (Eq) : relation réflexives, transitives et symétriques (exemple : égalité, congruance...).
- Relation de semi-ordre (SOr) : Relation transitive et antisymétrique.

**Relations entre concepts**

Voici la définition des relations entre concepts que nous proposons dans notre modèle du domaine :

- Synonyme : A reformule B si A est un synonyme de B. Exemple : *concept* reformule *idée*.
- Contraste : A contraste avec B si A propose une théorie alternative de B. Exemple : *Mécanique classique* contraste avec *Mécanique quantique*.
- Antonyme : A est une anti-thèse de B si A est un concept opposé à B. Exemple : *Algèbre linéaire* est une anti-thèse de *Algèbre non-linéaire*.

– Pré-requis : A est un pré-requis de B si l'apprentissage de B nécessite la connaissance de A.
– Relation hiérarchique : A est en relation hiérarchique avec B si A est une spécialisation du concept B.

Dans la table ci-dessous, nous présentons les propriétés mathématiques des relations entre concepts que nous venons de définir :

| Relation | R | IR | AR | S | AnS | AS | DS | T | C | AT | AC | Or | Eq | SOr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| hiérarchique | X | | | | X | | | X | | | X | X | | X |
| pré-requis | | X | | | X | | | X | | | X | | | X |
| antonyme | | X | | X | | | | | | X | | | | |
| contraste avec | | X | | X | | | | X | | | | | | |
| synonyme | X | | | X | | | | X | X | | | | X | |

**Relations entre ressources**

Voici la définition des relations entre ressources que nous proposons dans notre modèle du domaine :

– Prérequis : A est un prérequis de B si la lecture de la ressource A est nécessaire à la lecture de la ressource B.
– VersionDe : A est une version de B si A correspond à une version différente de la même ressource que B.
– Cite : A cite B si A contient une référence ou un lien vers B.
– Alternative : A est une alternative à B si on peut substituer A à B.
– Remplace : A remplace B s'il faut toujours substituer A à B.

Dans la table ci-dessous, nous présentons les propriétés mathématiques des relations entre ressources que nous venons de définir :

| Relation | R | IR | AR | S | AnS | AS | DS | T | C | AT | AC | Or | Eq | SOr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| alternative | X | | | X | | | | X | X | | | | X | |
| cite | | | X | | X | | | | | X | | | | |
| version de | X | | | X | | | | X | X | | | | X | |
| prérequis | | X | | | X | | | X | | | X | | | X |
| remplace | | X | | | X | | | | | X | | | | |

### 4.1.3 Justification des choix

Nous avons choisi de séparer les concepts et les ressources de manière très distinctes, de façon à former deux groupes disposant chacun de ses propres relations, et liés entre eux par la seule relation d'abstraction. Les concepts représentent les idées abstraites faisant partie du domaine. Les ressources représentent des entités concrètes, textes, images, sons, etc., indivisibles, et associées à un unique

concept. On sépare ainsi le domaine en deux couches : une couche de thésaurus, qui comprend l'ensemble des concepts du domaine, que l'utilisateur d'HA peut acquérir, indépendamment des différents documents qui en permettent l'acquisition ; et une couche de gestion et d'annotation des ressources qui peuvent être utilisées pour l'acquisition des concepts.

De cette façon, on peut par exemple importer le fait qu'un utilisateur connaît un concept, sans avoir à dire qu'il connaît les documents associés à ce concept, ce qui constituerait simplement un artifice informatique dans le cas où la notion de concept n'est pas directement accessible.

On trouve dans les hypermédias adaptatifs courants deux autres approches : une approche où tout les éléments du domaine sont des ressources, ce qui a pour effet de rester à un niveau assez bas dans les raisonnements, les connaissances étant toutes liées à des documents ; et une approche où les ressources ont pour pré-requis des concepts, ce qui permet, d'un point de vue informatique, les mêmes calculs que le modèle que nous proposons, mais qui est sémantiquement plus faible que notre modèle. En effet, si des concepts sont pré-requis d'une ressource, c'est parce que la ressource traite d'un concept, ou de plusieurs concepts, qui ne peuvent être abordés sans en connaître les pré-requis. Il est donc plus correct que la relation de pré-requis porte entre les concepts.

Nous avons défini deux notions de pré-requis distinctes. Une de relations de pré-requis concerne les concepts et l'autre concerne pour les ressources. La relation de pré-requis entre les concepts est fondamentale : elle est l'élément le plus important dans le calcul des documents adaptés, puisqu'elle permet de savoir quels sont les concepts qui peuvent être abordés ou non par l'utilisateur.

La relation de pré-requis entre les ressources est moins centrale. Néanmoins, elle permet d'affiner l'adaptation. En effet, si un exercice nécessite qu'ait été acquise une définition en particulier, la relation de pré-requis entre les ressources permet de le préciser, alors que la relation de pré-requis entre les concepts ne l'aurait pas permis. Les deux relations sont donc complémentaires.

Nous avons choisi d'utiliser des ressources atomiques. Dans certains systèmes, les ressources sont une appellation générale pour des éléments aussi bien atomiques que composites. Nous avons souhaité que la notion de ressource dans notre modèle soit particulièrement claire : seul des éléments indivisibles, d'un seul type, correspondant à un et un seul concept, sont des ressources. Par exemple, les ressources peuvent être représentées sous forme de fragments XML. Les ressources sont ensuite composées les unes avec les autres pour former des documents, qui sont éventuellement composites.

## 4.2 Modèle générique du domaine

Dans cette section, nous présentons un modèle générique pour les domaines dans les hypermédias adaptatifs, construit en nous appuyant sur les différents modèles de domaine qui existent déjà. Tout comme le modèle générique de l'utilisateur, ce modèle a pour but de mettre en exergue

les éléments essentiels qui constituent les domaines dans les hypermédias adaptatifs, mais aussi de permettre de créer de nouveaux modèles spécifiques à une catégorie de domaine en particulier.

Le diagramme de classe UML représentant le modèle générique du domaine que nous avons réalisé est présenté sur la figure 4.2. Ce modèle permet de représenter différents niveaux d'abstraction des ressources et/ou concepts. Il permet de gérer différentes catégories de relations, et notamment les propriétés mathématiques de ces relations, décrites dans la section précédente.

### 4.2.1 Responsabilité des classes

Nous détaillons ici le rôle des différentes classes de notre modèle générique :

- Interface a pour rôle de gérer le domaine et les communications entre le domaine et les autres agents.
- Constituant a pour rôle de représenter l'ensemble des éléments de base du domaine (ressources, concepts, pages etc.). Les relations ne sont pas considérées comme des constituants, elles représentent des liens entre différents constituants. Les constituants du domaine peuvent être plus ou moins abstrait, et il est possible d'abstraire ou de concrétiser un constituant.
- RepresentationRessource a pour rôle de représenter des constituants concrets (ressources, par exemple) du domaine. Une représentation de ressource est nécessairement composée de métadonnées.
- Relation a pour rôle de représenter les relations (binaires) entre les constituants du domaine.
- RelationEntreReprésentationRessourceEtConstituant représente les relations dont au moins un des éléments mis en relation est une représentation d'une ressource.
- RelationEntreRessources représente les relations entre représentation de ressources.
- Irréflexive, Réflexive, Symétrique, etc. permettent de représenter des relations avec différentes propriétés mathématiques, définies dans la partie 4.1.2

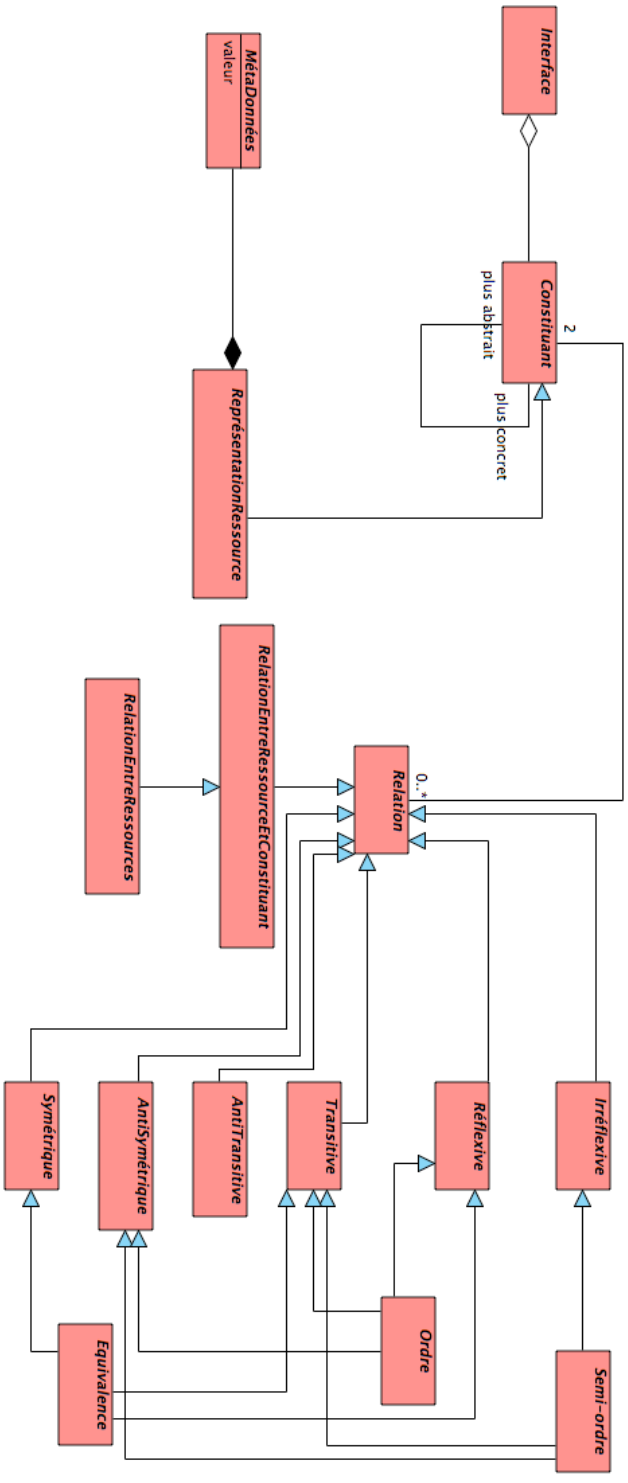


FIG. 4.2 – Modèle générique du domaine, en UML

### 4.2.2 Contraintes sur les spécialisations du modèle générique

Nous détaillons ici les différentes contraintes que nous imposons pour la spécialisation de notre modèle générique. Tout comme pour le modèle générique de l'utilisateur, certaines classes peuvent être spécialisées plusieurs fois, d'autres doivent l'être au moins une fois etc. :

- La classe Interface doit être spécialisée au moins une fois. Il n'y a pas de limite du nombre de sous-classe car on peut vouloir utiliser différentes interfaces pour communiquer avec différentes entités ;
- La classe Constituant doit être spécialisée au moins une fois. Cette spécialisation obligatoire peut éventuellement être une spécialisation de la classe RepresentationRessource ;
- La classe RepresentationRessource doit être spécialisée au moins une fois. Il faut au moins pouvoir représenter une sorte de ressources utilisées dans le système ;
- La classe Metadonnées doit être spécialisée au moins une fois. Il faut au moins un identifiant pour chaque ressource ;
- La classe Relation doit être spécialisé au moins une fois. Sans relation, il est impossible de faire de l'adaptation ;
- Toutes les autres classes sont sans contrainte de nombre de spécialisation.

### 4.2.3 Justification des choix

Contrairement au modèle de domaine que nous avons proposé ci-dessus, notre modèle générique ne reprend pas de manière aussi distincte la séparation entre un niveau abstrait (concepts) et un niveau concret (ressources). Il existe des modèles gérant de manières très différentes cette séparation, voire ne gérant qu'un seul niveau, le niveau concret. Notre modèle générique peut être spécialisé pour représenter aussi bien un système sans hiérarchie d'abstraction qu'un système avec une hiérarchie différente, éventuellement sur plusieurs niveaux d'abstraction.

Nous avons intégré au modèle générique un sous-ensemble des propriétés des relations binaires. Les propriétés mathématiques des relations ont des conséquences au niveau des inférences et des vérifications de données. En indiquant que la relation de pré-requis est transitive, un moteur d'adaptation prenant en compte les propriétés mathématiques peut aller chercher les pré-requis indirect d'un concept pour effectuer ses inférences. De plus la relation de pré-requis étant anti-symétrique et transitive, il ne peut y avoir de cycle dans les données entrées par un concepteur d'hypermédia adaptatif. Un programme adapté peut donc faire des vérifications sur les données de manière automatique, si les propriétés mathématiques des relations sont renseignées.

De plus, nous n'avons choisi qu'un sous-ensemble des propriétés permettant de catégoriser les relations binaires. Ce sous-ensemble correspond aux sortes de relations que l'on retrouve dans les hypermédias adaptatifs. Par exemple, la circularité d'une relation est rarement caractéristique des relations binaires dans les hypermédias adaptatifs, sauf si cette circularité découle de caractéristiques

plus fortes, par exemple la symétrie, que nous avons représenté dans notre modèle générique.

## 4.3 Lien entre notre modèle pour le e-learning et notre modèle générique

Dans cette section, nous montrons que notre modèle du domaine pour le e-learning est une spécialisation de notre modèle générique de domaine. Nous montrons ainsi, à travers notre propre exemple, qu'il est possible de spécialiser le modèle générique du domaine pour obtenir un modèle dans un domaine particulier. La section suivante montrera que notre modèle générique généralise également un certain nombre de modèles existant.

La figure 4.3 présente de quelle façon notre modèle est une spécialisation de notre modèle générique.

On voit ainsi que la classe Concept hérite de la classe Constituant, puisque les concepts sont abstraits. La classe Ressource hérite de la classe RepresentationRessource, puisqu'ils sont concrets. Ces deux classes du modèle pour le e-learning sont liées, tout comme leur super-classe respective, par la relation d'abstraction.

La classe GestionnaireDomaine hérite naturellement de la classe Interface du modèle générique.

Les propriétés mathématiques de chaque relation du modèle pour le e-learning sont données par des spécialisations de type "héritage multiple" des différentes classes de relation représentées dans le modèle générique. Ainsi, par exemple, la classe Pre-requis hérite de la classe semi-ordre, qui hérite elle-même des classe Irréflexive, Transitive et Antisymétrique.

## 4.4 Lien entre modèle générique et autres modèles

Dans cette section, nous montrons que notre modèle générique du domaine est bien une gnéralisation de modèles connus, notamment du Munich Model, justifiant ainsi de sa conception, et du bien-fondé de son utilité pour concevoir de nouveaux modèles. Cette section montre l'intérêt de laisser, comme nous l'avons fait, une certaine liberté au niveau de la gestion des niveaux d'abstraction des constituants des domaines.

### 4.4.1 Le Munich Model

La figure 4.4 représente le modèle du domaine du Munich Model. On remarque qu'un certain nombre d'identifiants de composants sont détachés des composants qu'ils identifient. Nous ne nous préoccuperons pas de cette spécificité, qui n'est pas prise en compte dans le modèle générique, mais qui ne pose pas de problème lors de la spécialisation, puisqu'il ne s'agit que de classes supplémentaires, et non pas de classes qui modifient le sens du modèle et empêchent d'en faire une

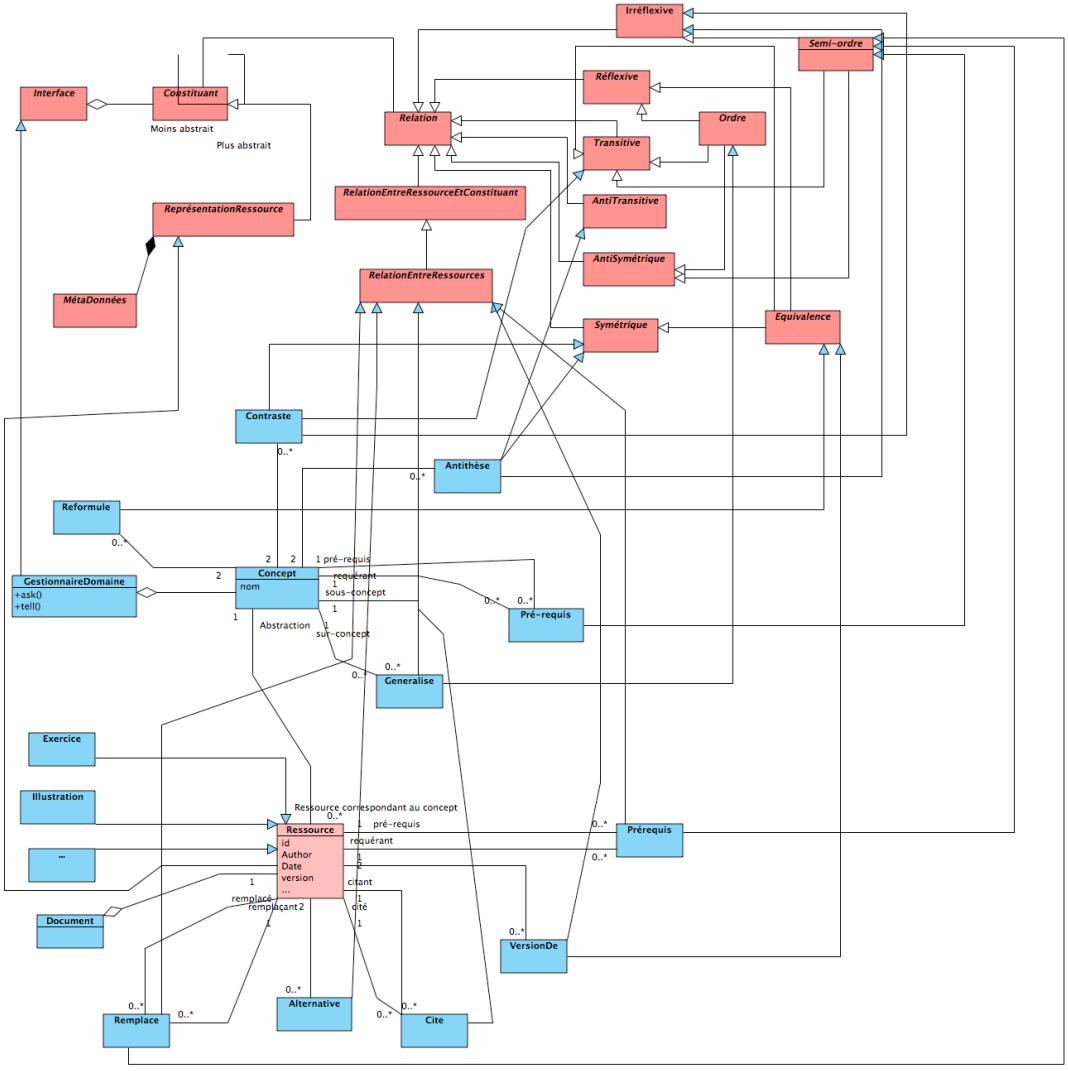


FIG. 4.3 – Lien entre modèle et modèle générique du domaine, en UML

spécialisation de notre modèle générique. De plus, ces classes pourraient être incorporées directement dans les classes identifiées : il ne s'agit, dans le modèle présenté sur la figure 4.4, que d'une façon de représenter les différents éléments. Il s'agit d'un modèle de conception, le modèle d'analyse correspondant n'a pas besoin de prendre en compte ce niveau de détail.

Nous présentons sur la figure 4.5 une version simplifiée du modèle de l'utilisateur du Munich Model, ne reprenant pas en détail les méthodes et attributs de ce modèle, mais montrant où placer les attributs si on ne veut pas en faire des classes à part. Les ressources concrètes, représentées par les classes du package "Within-Component Layer", ne sont pas représentées sur ce diagramme : notre modèle générique ne donne pas de façon spécifique de représenter les données matérielles.

La figure 4.6 présente de quelle façon le modèle du domaine du Munich Model est une spécialisation de notre modèle générique. On note que la classe Component, qui représente à la fois les constituants et les relations, n'est pas une spécialisation d'une classe de notre modèle générique :

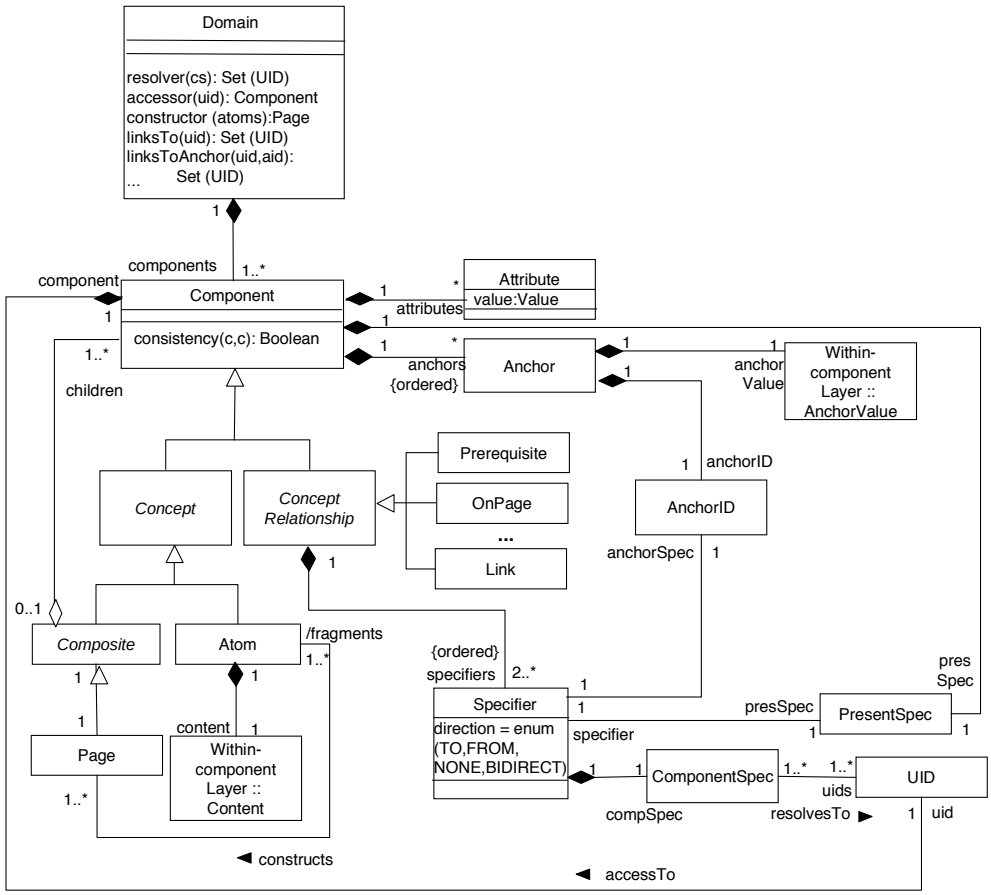


FIG. 4.4 – Modèle du domaine du Munich Model, en UML

le fait que les relations soient stockées et gérées directement dans l'interface est une spécificité du Munich Model, qu'il ne nous paraît pas indispensable de conserver au niveau générique : le seul accès aux constituants suffit à trouver les relations qui les lient entre eux. On note également que Anchor hérite de RepréntationRessource : une ancre est une subdivision d'un document, donc une ressource concrète.

On remarque que, dans le Munich Model, la notion de concept est associée à des ressources plus ou moins concrètes sur le domaine. Ces ressources sont subdivisées en catégories et sous-catégories plus ou moins atomiques. Ainsi, on peut considérer que les concepts atomiques, qui correspondent à des ressources atomiques, et les pages (composées uniquement de concepts atomiques) héritent de la classe RepréntationRessource, alors que les classes concept et composite, qui peuvent ne pas représenter d'éléments concrents du domaine, héritent de la classe Constituant. On note que la notion d'abstraction dans ce modèle est très différente de celle que nous utilisons dans notre modèle pour le e-learning.

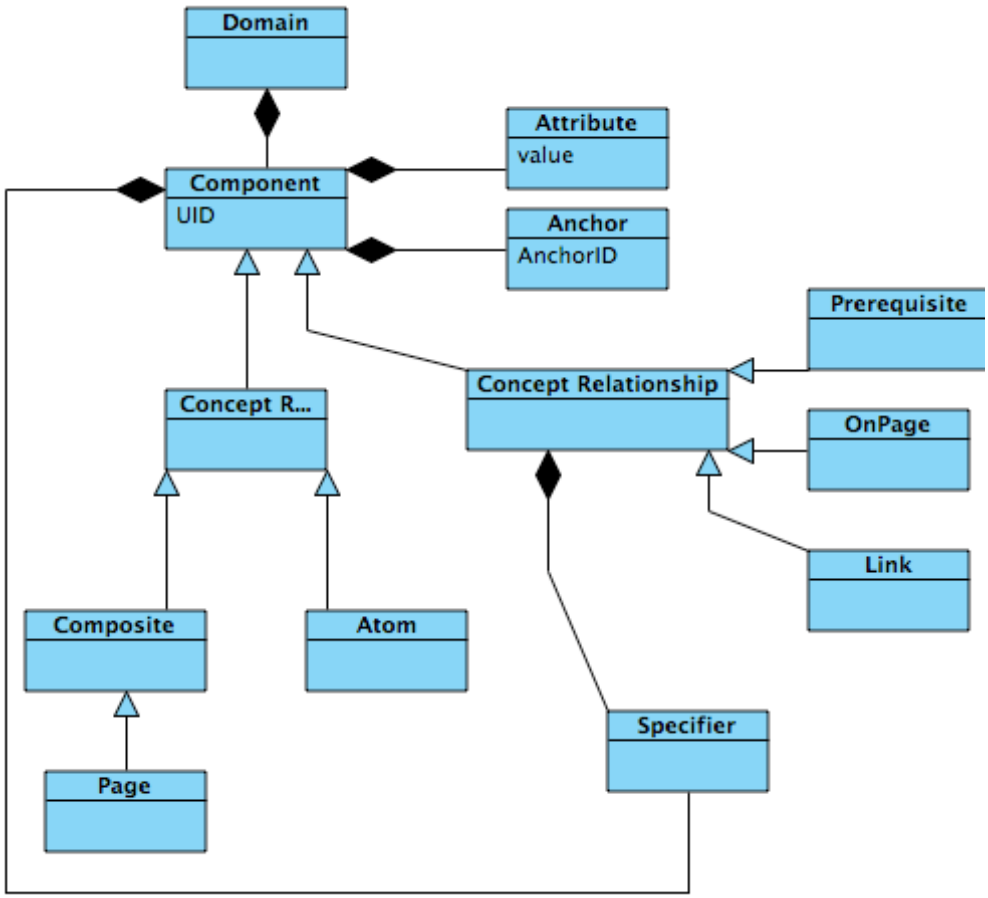


FIG. 4.5 – Modèle du domaine du Munich Model, simplifié, en UML

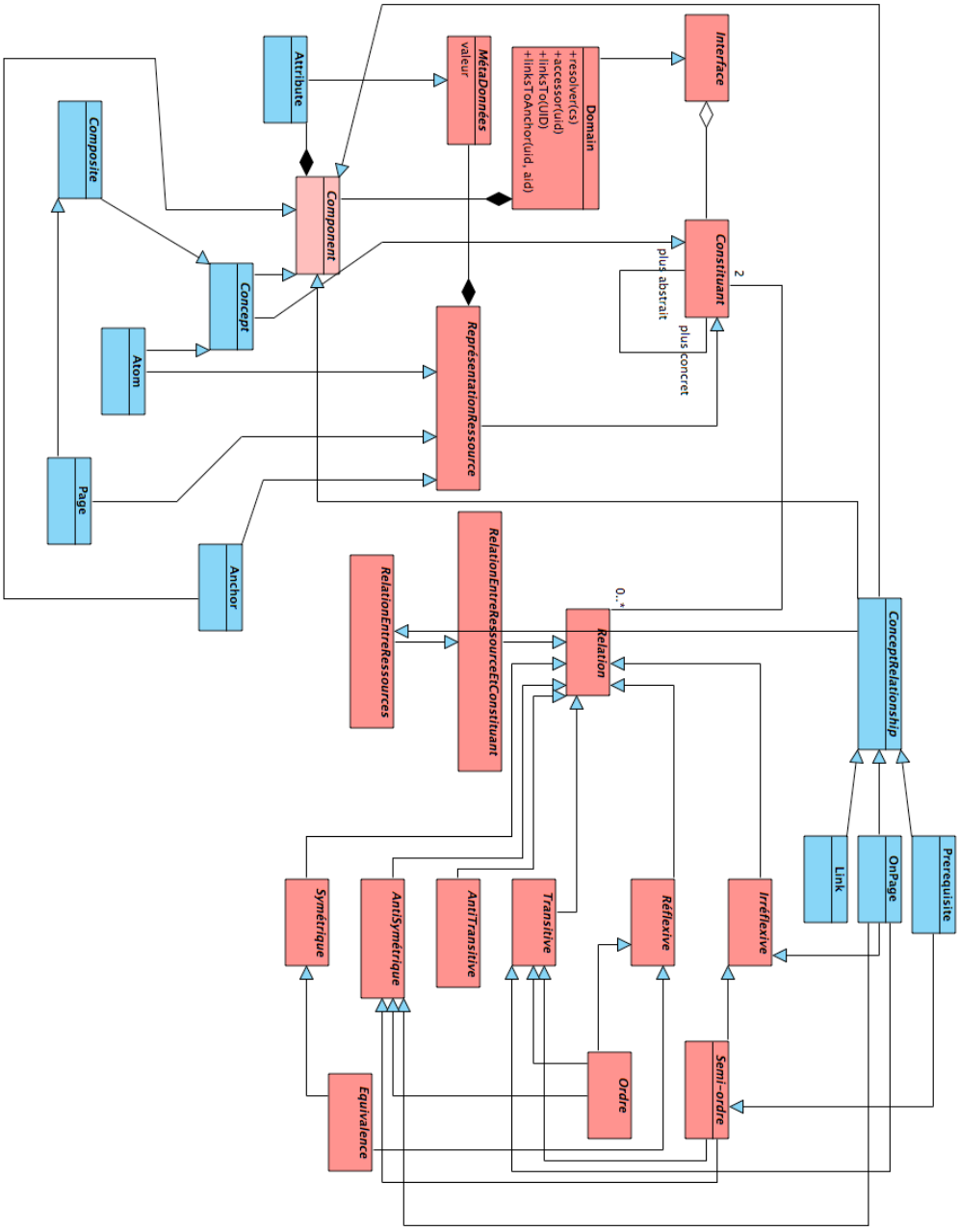


FIG. 4.6 – Lien entre le Munich Model et le modèle générique du domaine, en UML

### 4.4.2 Le système de Duitama

Nous avons étudié la modélisation pour les domaines que propose Duitama dans [23]. La figure 4.7 présente une modélisation UML possible du système de Duitama, que nous avons inféré de cette étude.

Notons que le DomainManager est implicite dans ce modèle, mais il ne saurait y avoir de domaine sans gestionnaire.

Nous voyons dans ce modèle une approche différente à la fois de celle que nous avons utilisé pour notre modèle pour le e-learning et de celle utilisée dans le Munich Model. En effet, un constituant concret du domaine peut être soit un composant éducatif, soit un opérateur entre deux composants éducatifs (alternative ou parallèle), ce qui permet de représenter un certain type de graphes de ressources. On a ainsi une classe DomainNodes qui représente toutes les ressources concrètes. Les composants éducatifs sont représentés par la classe EducationnalComponent et les opérateurs par la classe ChoiceOperator, qui a elle-même deux sous-classes selon le type d'opérateur. La relation de pré-requis relie un concept à un composant pédagogique : certains concepts sont pré-requis pour utiliser un composant éducatif donné.

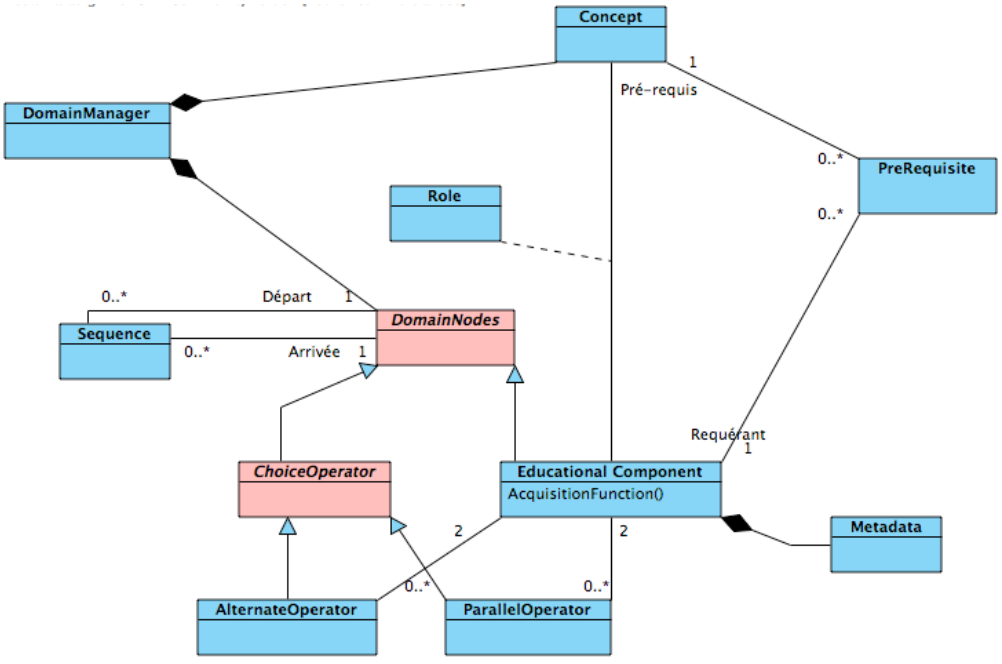


FIG. 4.7 – Modèle du domaine du système de Duitama, en UML

La figure 4.8 présente de quelle façon le modèle de domaine de Duitama spécialise notre modèle générique. Nous voyons que la classe DomainNodes hérite de la classe RepresentationRessource, et que la classe Concept hérite de la classe Constituant. À cet égard, le modèle de Duitama est assez proche du nôtre. La classe PreRequisite hérite de la classe RelationEntreRessourceEtConstituant, que nous n'avions pas encore utilisé jusqu'ici. La relation d'abstraction, qui relie Concept et EducationalComponent est plus forte que celle au niveau du modèle générique : elle est munie d'une classe

d'association, permettant de définir le rôle du concept pour la ressource à laquelle il est associé.

## 4.5 Conclusions

Dans ce chapitre, nous avons présenté notre modèle du domaine pour le e-learning. Celui-ci permet de séparer un niveau conceptuel, réutilisable, et un niveau concret, dépendant des ressources mises à la disposition du système. Ce modèle permet également de prendre en compte différentes sortes de ressources, comme les exercices, les illustrations, etc., qui sont spécifiques au domaine du e-learning, ainsi que des méta-données pédagogiques. Il présente enfin un certain nombre de relations entre concepts et entre ressources, qui doivent permettre de décrire des stratégies d'adaptation. Ce modèle est à la fois simple à appréhender et suffisamment complet pour couvrir les besoins envisagés dans le domaine du e-learning.

Nous avons également présenté un modèle générique pour capturer les points communs entre la plupart des modèles de domaine dans les hypermédias adpatatifs et pour permettre de créer de nouveaux modèles spécifiques à un type d'hypermédia en particulier. Nous avons montré à travers quelques exemples que ce modèle générique est bien une généralisation de modèles connus, ainsi que de notre modèle. Réutiliser ce modèle générique doit permettre de faciliter la conception d'un modèle de domaine, mais également de réimporter des données initialement fournies dans un autre modèle. Bien que nous ne nous soyons pas intéressé aux difficultés à surmonter pour faire transiter des données d'un système à un autre, la connaissance d'un modèle générique commun aux modèles participant à une opération d'importation doit permettre de faciliter cette tâche souvent complexe.

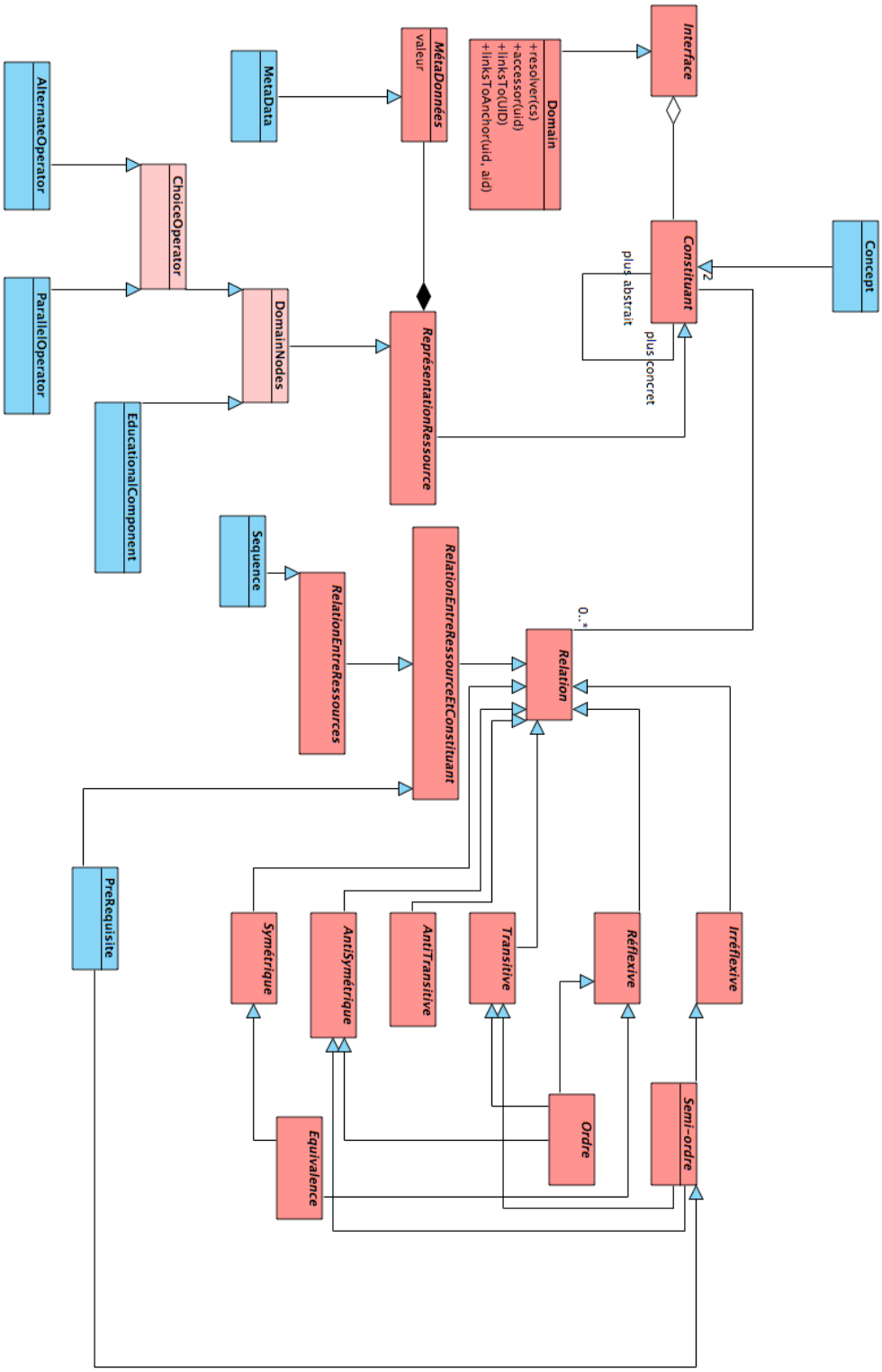


FIG. 4.8 – Modèle du domaine du système de Duitama et modèle générique du domaine, en UML

# Chapitre 5

# Modélisation logique de l'adaptation

Ce chapitre présente une modélisation basée sur le calcul des prédicats du premier ordre des mécanismes d'adaptation que l'on peut mettre en oeuvre dans de les systèmes d'hypermédias adaptatifs. Le but premier de cette approche est de fournir un modèle théorique, basé sur un formalisme bien fondé et non programmatique, pour les mécanismes d'adaptation. Ce modèle est également conçu dans le but d'être utilisé pour concevoir concrètement des moteurs d'adaptation, contrairement aux caractérisations logiques de hypermédias adaptatifs [33].

Afin de parvenir à ces résultats, nous nous sommes particulièrement intéressés à l'utilisation de la logique situationnelle pour modéliser l'ensemble de l'adaptation. Cette logique, basée sur le calcul des prédicats du premier ordre, permet de représenter nativement des situations, des actions, la façon dont les actions modifient les situations etc. Il s'agit donc d'un formalisme particulièrement adapté à nos besoins : il est basé sur la logique et permet de capturer simplement l'évolution d'un système d'hypermédia adaptatif. La logique situationnelle avait d'ailleurs été introduite par Sheila McIlraith pour le web adaptatif dans [53].

Au-delà de la modélisation d'un mécanisme d'adaptation global à l'aide de la logique situationnelle, nous avons introduit un système de règles évolué, basé sur l'utilisation de méta-règles. Il permet de prendre en compte les différents aspects (données sur l'utilisateur, le domaine etc.) du système dans différentes couches de règles. Le but de cette approche est d'empêcher l'écriture de règles longues et difficiles à maintenir, de permettre la sélection de véritables stratégies distinctes d'adaptation selon l'utilisateur et de pouvoir fournir un formalisme de déduction.

Une fois notre modèle d'adaptation et notre formalisme de règles établi, nous nous sommes intéressés à l'affinement de l'adaptation de parcours par l'établissement automatisé de tronçons de parcours, qui permettent à l'utilisateur de n'arrêter son parcours que sur des objectifs intermédiaires définis comme tels, c'est-à-dire de ne clore sa session que sur des documents terminant une partie cohérente du parcours à effectuer jusqu'au but fixé.

Dans un premier temps, nous allons détailler notre modèle d'adaptation basé sur la logique situationnelle. Ensuite, nous détaillerons notre formalisme de règles et donnerons un exemple simple

d'utilisation. Enfin, nous présenterons nos travaux concernant les parcours par tronçons.

## 5.1 Modèle de raisonnement

Afin de choisir un formalisme bien fondé pour représenter l'adaptation dans les systèmes d'hypermédias adaptatifs, nous avons choisi d'étudier différents mécanismes de planification. En effet, fournir de l'adaptation dans un tel système revient à prévoir, au fur et à mesure des actions entreprises par l'utilisateur, la succession des documents que l'on peut ou doit lui présenter. Il nous paraît donc naturel de s'intéresser aux approches entreprises dans le domaine de la planification pour formaliser le problème de l'adaptation dans notre domaine. L'échelle des données à traiter est néanmoins très différente dans les deux cas : dans les hypermédias adaptatifs, la quantité de données à traiter - l'ensemble des constituants du domaine et des attributs d'un utilisateur - est beaucoup plus faible que dans les problèmes de planification. De ce fait, nous ne nous intéresserons pas particulièrement aux problèmes d'optimisation en jeu dans les systèmes de planification.

Les problèmes d'ordonnancement d'un chantier, d'une chaîne de production, de mise en place d'un emploi du temps... peuvent tous être modélisés et résolus en utilisant des techniques de planification. STRIPS (cf. [27]) a été introduit en 1971 pour modéliser l'ensemble des problèmes de planification, rejoint un peu plus tard par d'autres formalismes (cf. [56]). De nombreuses techniques ont été développées pour résoudre avec un bon rapport précision/rapidité, les différents problèmes de planification, modélisés sous forme d'un graphe STRIPS. À la fin des années 1990, la planification en ordre partiel s'est particulièrement distinguée, et notamment le système Graphplan (cf. [3]).

Au début des années 2000, l'idée d'utiliser des heuristiques pour prévoir des plans optimisés a été mise en pratique [5]. Le principe général consiste à chercher une approximation d'une fonction d'optimisation d'un problème dit «relâché», et à appliquer cette fonction au problème réel pour construire, étape après étape, un chemin optimisé. Une autre technique de planification heuristique basée sur des heuristiques choisies au cas-par-cas a été introduite en 2004 (cf. [63]). Cette méthode utilise des règles de sélection d'action et la logique situationnelle. Ces règles sont décrites différement pour chaque problème posé. Cette méthode permet une compréhension plus directe des phénomènes décrits dans les règles que la méthode décrite dans [5].

Dans cette section, nous allons montrer succinctement que notre problème d'adaptation est comparable, pour certains aspects, à un problème de planification. Nous nous intéresserons ensuite à une méthode de planification qui s'avère particulièrement intéressante pour l'adaptation, puisqu'elle met en oeuvre des règles déclaratives pour trouver un plan. Cette méthode repose sur une logique situationnelle, que nous avons en partie modifié pour fournir la base de notre système d'adaptation.

### 5.1.1 Méthodes de planification

Dans cette section, nous nous intéressons tout d'abord à la modélisation STRIPS des problèmes de planification, qui sert de base à la description de ces problèmes. Puis nous étudions une méthodes utilisant des heuristiques et la logique situationnelle pour chercher des plans. Nous nous sommes particulièrement intéressés à la planification heuristiques, car le problème d'adaptation dans les hypermédias adaptatifs nécessite des moyens de dégager les actions souhaitables des autres, autrement qu'en sélectionnant le parcours qui mène le plus rapidement au but. De plus, il n'est pas possible de définir un parcours absolument meilleur que les autres : il s'agit de parcours de documents, pas de la construction d'un immeuble ! Les heuristiques permettent de résoudre ces problèmes particuliers.

### 5.1.2 Modélisation d'un problème de planification

Dans cette partie, nous donnons quelques éléments de base pour modéliser un problème de planification en utilisant le formalisme STRIPS.

Un problème de planification est constitué d'une situation initiale (celle dont on souhaite partir pour résoudre le problème), d'une situation finale (celle à laquelle on souhaite arriver) et de situations intermédiaires. Chaque situation comprend différents faits ou atomes. Par exemple dans un problème de construction de pont, «le tablier est construit» est un atome, et il fait entre autre partie de la situation finale. Pour passer d'une situation à une autre, on effectue des opérations (par exemple «construire le tablier»). Ces opérations ne peuvent pas toujours être effectuées : on ne peut pas construire le tablier si les piliers ne sont pas encore construits ! Les opérations ont donc des préconditions. Une opération modifie la situation en modifiant les atomes de la situation : certains atomes peuvent ne plus être satisfaits, et d'autres peuvent le devenir : après l'opération «construire le tablier», l'atome «le tablier est construit» fait partie de la nouvelle situation.

C'est sur ces considérations qu'est construit le modèle STRIPS. Dans ce formalisme, un problème de planification est un quadruplet $(A, O, I, G)$. $A$ représente un ensemble d'atomes. $O$ est un ensemble d'opérateurs. $I$ est la situation initiale. $G$ est la situation finale. $I$ et $G$ sont des sous-ensembles de $A$.

Pour chaque opérateur, on définit trois sous-ensembles de A :

1. $Prec(op)$, qui définit les atomes préconditions de $op$ i.e. ceux qui doivent faire partie de la situation courante pour pouvoir lui appliquer $op$ ;
2. $Add(op)$, qui définit les atomes que $op$ ajoute à la situation courante ;
3. $Del(op)$, qui définit les atomes que $op$ retire à la situation courante.

### 5.1.3 Exemple de modélisation STRIPS du problème d'adaptation

Le but de cette partie est de montrer à travers un exemple simple que notre problème peut être modélisé en tant que problème de planification.

Nous considérons dans cette partie que nous disposons d'un espace de documents indivisibles annotés, et reliés entre eux par des relations de pré-requis. Nous disposons également d'un modèle de l'utilisateur, qui recense l'état des connaissances de ce dernier.

Le domaine est constitué uniquement de documents indivisibles. On considère ici un cours de mathématiques portant sur la dérivation et l'intégration des fonctions réelles. On dispose des documents suivants :

1. Introduction
2. Illustration graphique de la dérivation par la construction de tangentes
3. Définition de la dérivation
4. Méthode de calcul de la fonction dérivée
5. Définition de la primitive
6. Définition de l'intégration
7. Application de l'intégration au calcul d'une aire
8. Exercices sur la dérivation
9. Exercices sur les primitives et l'intégration
10. Test

On se dote de la relation de pré-requis entre les document pour construire le graphe. On utilise deux concepts pour annoter les documents : le concept «dériver» et le concept «intégrer». Les documents traitant du concept «dériver» sont les documents 1, 2, 3, 4, 8 et 10, ceux traitant du concept «intégrer» sont les documents 1, 5, 6, 7, 9 et 10. Le graphe des documents est représenté sur la figure 5.1

Nous considérons que chaque document correspond à un concept, et que le but à atteindre est constitué d'un ou plusieurs concepts. Nous considérons également que chaque document lu est su, i.e. que l'ensemble des connaissances d'un utilisateur correspond à l'ensemble des documents qu'il a lu.

Nous définissons alors le quadruplet STRIPS $(A, O, I, G)$ de la façon suivante :

- Chaque atome correspond à la connaissance par l'utilisateur d'un document. Au document $doc_i$ correspond l'atome $a_i$ ;
- Une opération consiste en la lecture d'un document. Il y a donc autant d'opération que de documents. On notera naturellement $op_i$ l'opérateur correspondant à la lecture du $i^{eme}$ document ;
- $Add(op_i) = \{a_i\}$
- $Del(op_i) = \emptyset$
- $Prec(op_i)$ est constitué de l'ensemble des $a_j$ tels que $doc_j$ est un pré-requis de $doc_i$ ;
- $I$ est constitué de l'ensemble des documents connus par l'utilisateur au lancement de l'application ;

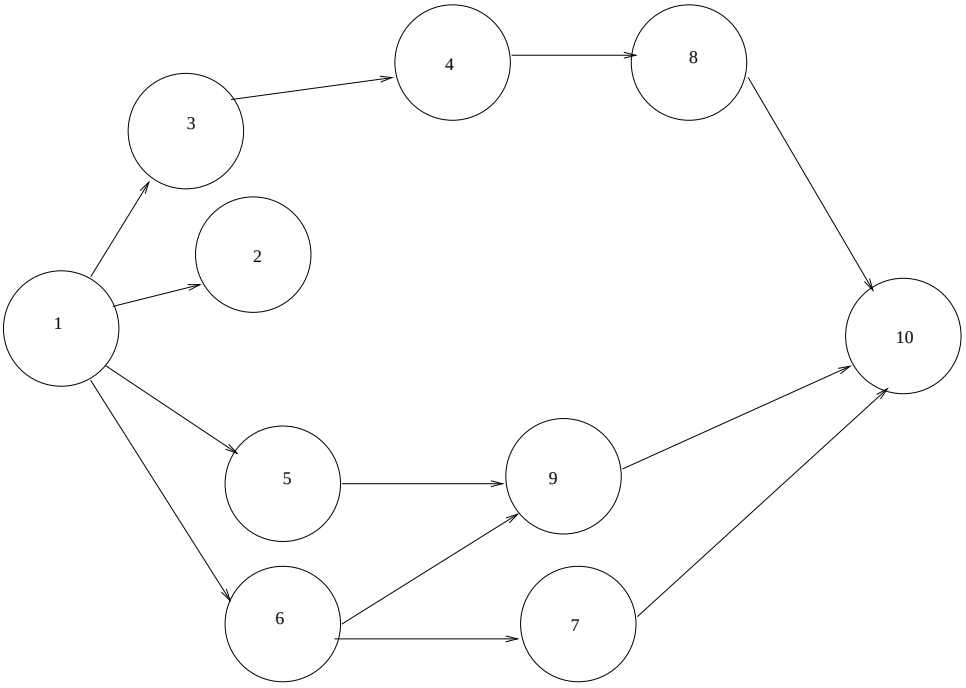


FIG. 5.1 – Exemple simple de graphe de documents

– $G$ est constitué de l'ensemble des documents dont on souhaite que l'utilisateur prenne connaissance.

Voyons comment se déroulerait un parcours dans notre exemple modélisé de la façon que nous venons de présenter.

Dans l'état initial, les connaissances de l'utilisateur sont nulles. On a donc $I = \emptyset$. Le seul opérateur tel que $Prec(Op_i) = \emptyset$ est $Op_1$. On applique cette opération, et l'utilisateur visualise et prend connaissance de $doc_1$.

L'état courant est alors le singleton $\{a_1\}$. Les opérations ayant pour seul prérequis $a_1$ sont $Op_2$, $Op_3$, $Op_5$ et $Op_6$.

Si l'utilisateur choisit par exemple $Op_6$, il prend connaissance du document $doc_6$ et se trouve ensuite dans l'état $\{a_1, a_6\}$. Les opérations possibles depuis cet état sont $Op_2$ (précondition $a_1$), $Op_3$ (précondition $a_1$), $Op_5$ (précondition $a_1$) et $Op_7$ (précondition $a_6$).

Si l'utilisateur choisit ensuite d'appliquer $Op_5$, il se retrouve dans l'état $\{a_1, a_5, a_6\}$. Les opérations qu'il peut alors effectuer sont d'une part toujours $Op_2$, $Op_3$ et $Op_7$, mais aussi $Op_9$ qui a pour précondition l'ensemble $\{a_5, a_6\}$.

On continue ainsi jusqu'à pouvoir effectuer finalement $Op_{10}$ qui a pour précondition l'ensemble $\{a_8, a_9, a_7\}$.

On voit donc que le problème de parcours de documents peut être modélisé de la même façon qu'un problème de planification. On notera la différence entre le graphe des documents, et le graphe des états qui offre plus de possibilités, puisque un document reste accessible tant qu'il n'est pas lu.

Ici, le graphe des états est composé de 70 noeuds environ, alors que le graphe des documents est composé de 10 noeuds. La figure 5.2 illustre les trois premières étapes du graphe des états. Dans cette figure, «*x/y/z/*...» signifie que l'utilisateur a pris connaissance des documents $x$, $y$, $z$....

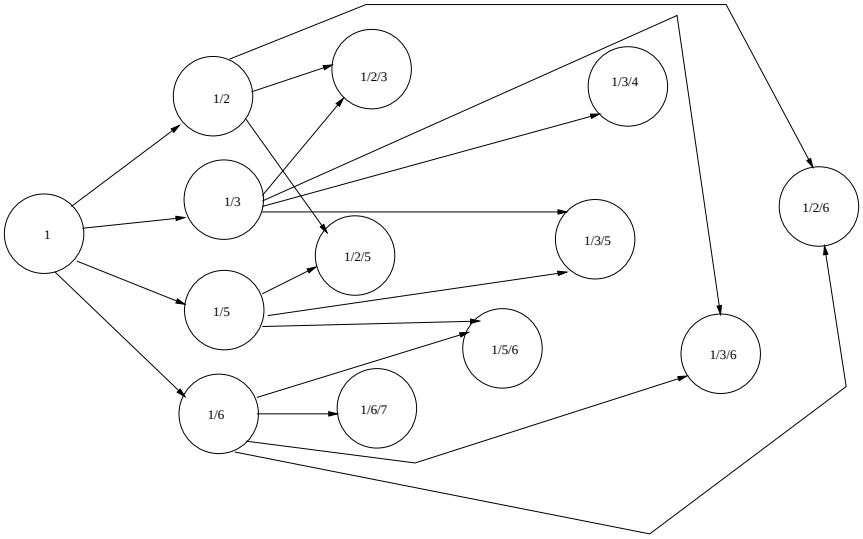


FIG. 5.2 – Partie du graphe des états tiré du graphe des documents

Nous noterons également que cet exemple, qui, pour des raisons de simplicités, ne fait mention que d'adaptation de parcours de documents au contenu fixe, peut être facilement étendu au problème d'adaptation dans sa globalité, c'est-à-dire incluant les autres formes d'adaptation. En effet, les documents composés peuvent-être considérés comme des documents différents et on obtient ainsi une modélisation du même type que précédemment, avec beaucoup plus d'états possibles.

### 5.1.4 Planification par heuristiques permettant la classification d'actions

L'auteur de [63] propose une méthode non calculatoire, permettant d'utiliser des heuristiques pour résoudre les problèmes de planification au cas-par-cas, c'est-à-dire en donnant des heuristiques différentes pour chaque problème. Son approche est orientée vers la logique situationnelle [59]. Dans la logique situationnelle, on effectue des actions pour modifier la situation courante. L'auteur de cet article propose d'utiliser une méthode heuristique de sélection d'actions, basée sur une forme améliorée de cette logique.

**Rappels sur la logique situationnelle**

La logique situationnelle est basée sur le calcul des prédicats du premier ordre. Elle définit un certain nombre de prédicats, et la façon de les utiliser pour raisonner. Nous allons voir dans cette partie comment la logique situationnelle est définie, et comment elle fonctionne.

La logique situationnelle est constituée de situations, qui définissent l'état du système à un instant donné, c'est-à-dire entre deux actions. Les actions modifient les situations.

*Situations*

Une situation est décrite comme le résultat d'une action appliquée à une autre situation. Le prédicat $do$ permet de connaître la situation obtenue par application d'une action dans une situation donnée. Ainsi, $do(a, s)$ définit la situation obtenue en appliquant l'action $a$ à la situation $s$. On peut ainsi remonter jusqu'à la situation initiale. Toute situation est alors de la forme :

$$do(a_n, do(a_{n-1}, \ldots, do(a_0, s_0) \ldots))$$

où les $a_i$ sont des actions et où $s_0$ est la situation initiale.

*Fluents*

Tout comme dans la réalité, seuls certains aspects d'une situation sont connus. Afin de connaître la valeur de ces aspects dans une situation donnée, on dispose de fluents, qu'il faut voir comme des observateurs de la situation courante. Par exemple, dans un problème de gestion d'ascenceur, un fluent permet de connaître l'étage courant. Pour chaque fluent, il faut fournir :

– Sa valeur dans la situation initiale ;

– Sa valeur pour une situation de la forme $do(a, s)$, i.e. la façon dont il évolue avec les actions.

À cet effet, la logique situationnelle définit le prédicat $holds$, qui permet de déclarer, à l'aide de règles, la valeur d'un fluent dans une situation donnée. Le plus souvent, les règles définissant les fluents permettent de calculer la valeur du fluent en fonction de la dernière action effectuée et de la situation précédent cette action. Ainsi, on définit le plus souvent des règles définissant $holds(fluent, do(a, s))$ pour décrire la façon dont les situations évoluent en fonction des actions effectuées.

Comme nous l'avons vu dans le paragraphe précédent, toute situation est décrite comme un succession d'actions appliquées à la situation initiale. Ainsi, il est possible de déterminer la valeur de chaque fluent à chaque instant, par un calcul récursif éventuellement optimisé.

*Actions*

Les actions permettent de modifier les situations. Un fluent ne pourra éventuellement changer de valeur qu'après une action, et seulement dans le cas où cette dernière a une influence sur l'aspect de la situation décrit par le fluent.

Les actions sont catégorisées selon deux axes orthogonaux. Dans une situation donnée, on distingue deux types d'action en logique situationnelle. Les actions possibles, et celles qui ne le sont

pas. La possibilité d'effectuer une action dépend de la situation courante. À cet effet, on définit le prédicat $poss$, qui permet de décrire, à l'aide de règles, quelles conditions sont nécessaires pour qu'une action soit possible. Les règles de possibilité sont de la forme :

$premisses \Rightarrow poss(action, situation)$

On sélectionne alors une action parmi les actions possibles. La possibilité d'effectuer une action peut-être donné par une ou plusieurs règles, décrites sous forme de clause de Horn.

En outre, une action peut être primitive ou composite. Par exemple, dans un problème de cubes à empiler, une action primitive peut être *dépiler un cube* et une action composite peut être *Tant qu'il reste des cubes dans la pile, dépiler le cube le plus haut* ce qui revient à dépiler toute une pile par une suite d'actions primitives. Les actions composites permettent ainsi de décrire des séquences d'actions, des disjonctions d'action (choix aléatoire) etc. On trouve plus de détails sur les actions composites dans [25].

*Fonctionnement*

On se donne une situation initiale, et on décrit les fluents dans cette situation. Puis on sélectionne, parmi les actions possibles, une action à appliquer. On applique cette action, ce qui modifie la situation initiale. On obtient une nouvelle situation, à partir de laquelle on peut effectuer diverses actions possibles et ainsi de suite.

À chaque étape du calcul, on peut soit déterminer l'ensemble des actions possibles et ne proposer que celles-ci, soit proposer toutes les actions, puis vérifier si celle qui est envisagée est possible. La première solution permet de proposer par avance les seuls choix possibles, mais nécessite potentiellement plus de calculs à chaque étape. La seconde solution peut être moins gourmande en calculs, mais ne permet pas de savoir à l'avance si l'action envisagée sera possible.

De la même façon, on peut toujours calculer tous les fluents, ou ne les calculer qu'à la demande. Dans le premier cas, le temps de calcul pour un fluent après un très grand nombre d'action sera court, mais chaque étape demandera plus de calculs. Dans le second cas, le calcul pourra être beaucoup plus long, mais si peu de fluents sont utiles à chaque étape, on gagne du temps à ne pas tout recalculer. On peut toujours, pour améliorer l'efficacité, stocker les valeurs des fluents chaque fois qu'elles sont calculées pour les besoins du programme, de sorte à ne pas systématiquement remonter jusqu'à la situation initiale.

#### Améliorations possibles de la logique situationnelle

Le modèle de planification par heuristiques proposé dans [63] repose sur une version modifiée de la logique situationnelle. Nous allons détailler ici les différences entre la logique situationnelle de base, et celle employée pour la planification.

La sélection d'actions à appliquer dans la version de la logique situationnelle proposée ici se fait par des règles heuristiques, qui permettent de déterminer la valeur de trois prédicats :

1. Le prédicat $Good$, qui caractérise les actions optimales. Ainsi $Good(a, s)$ est vrai si l'action $a$

est bonne dans la situation $s$, i.e. si elle mène vers un état qui approche de l'objectif.

2. Le prédicat $Bad$ caractérise les actions qui ne peuvent pas faire partie d'un plan optimal. Ainsi, on n'effectuera jamais l'action $a$ dans la situation $s$ si $Bad(a, s)$ est vrai.

3. Le prédicat $Better$ fournit un ordre partiel pour les actions qui ne sont ni de classe $Good$ ni de classe $Bad$. $Better(a, b, s)$ est vrai si l'action $a$ est préférable à l'action $b$ dans la situation $s$. On remarque que l'on peut facilement étendre cette relation d'ordre partiel aux actions de classes $Good$. Si deux actions sont bonnes, il peut être intéressant de déterminer celle qui est préférable.

Ces règles sont utilisées de la façon suivante : dans une situation donnée $s$, on cherche une action $a$ telle que $Good(a, s)$ et on l'applique. Si on ne trouve aucune action telle que $Good(a, s)$, on cherche parmi celles telles que $\neg Bad(a, s)$ une des meilleures (ou la meilleure) au sens de la relation d'ordre partiel $Better$. Si toutes les actions sont telles que $Bad(a, s)$, alors on retourne à la situation précédente et on sélectionne une autre action.

*Exemple de règle heuristique*

Voici une des règles de [63] dans le monde des blocs, dont le but est d'empiler des cubes dans un ordre donné sur une table :

```
bad(move(X, _, _), S) :- final(X, S).
bad(move(X, _, Z), S) :- ong(X,W), not(Z=t),
                         (not(clear(W, S));
                          not(final(W, S))).
```

On remarque que les règles sont écrites ici en Prolog i.e. sous forme de clauses de Horn, et que $Bad$, $Good$ et $Better$ peuvent donc être définis par plusieurs règles (en réalité la disjonction de ces règles). La première clause de Horn proposée signifie qu'il est mauvais de bouger un bloc quelle que soit sa destination s'il est déjà dans sa position finale telle que cette position est définie dans l'objectif. La seconde clause signifie qu'il est mauvais de déplacer un bloc ailleurs que sur la table (représentée par $t$) s'il n'est ni sur la table ni dans sa position initiale et s'il ne peut être bougé vers sa position finale.

Si on avait utilisé une méthode calculatoire de recherche de plan optimal pour résoudre ce problème, il aurait fallu décrire chaque opérateur, c'est-à-dire, par exemple, les listes $Add$, $Del$ et $Prec$ du modèle STRIPS pour chaque $move(X, Y, Z)$ et ce pour chaque valeur de $X$, $Y$ et $Z$. La description est donc beaucoup plus intuitive ici. Par contre, trouver les règles qui sélectionnent des actions «toujours bonnes» ou «toujours mauvaises» peut-être plus compliqué que de donner exhaustivement les préconditions, les atomes ajoutés et retirés, et de laisser faire l'optimisation par un calcul générique.

Il est possible de définir un quatrième prédicat pour affiner la recherche. Il s'agit du prédicat $BadSituation$, qui ne porte pas sur la sélection d'actions contrairement aux trois autres, mais qui

permet de savoir si une situation n'est pas acceptable pour la construction d'un plan optimal. Si on aboutit à une situation $s$ telle que $BadSituation(s)$ est vrai, alors on revient en arrière et on choisit une action qui mène vers un état différent. Si aucune action ne mène à une situation acceptable, on revient en arrière de nouveau.

Cette méthode est inspirée de [1] qui décrit le système de planification TLPlan, basé sur une logique temporelle. Les heuristiques appliquées à la logique situationnelle permettent de décrire plus simplement les mêmes phénomènes, et elles permettent également de décrire plus de phénomènes.

Il est possible d'adapter ces heuristiques dans des «programmes par historiques», qui traitent la négation autrement qu'en Prolog, c'est-à-dire qui n'utilisent pas la négation par l'échec qui est non-monotone.

*Fonctionnement des heuristiques et de la logique situationnelle*

On considère que le caractère possible ou non des actions est donnée par les règles heuristiques sur les actions. Toutes les actions qui ne sont pas de classe $Bad$ sont possibles. Les actions de classe $Bad$ ne sont pas possibles.

À chaque étape, on sélectionne une action. S'il existe une action de classe $Good$, on la sélectionne et on l'applique à la situation courante. S'il n'existe aucune action de classe $Good$, on sélectionne une des meilleures actions - qui ne sont pas de classe $Bad$ - au sens de la relation d'ordre partiel $Better$. Si toutes les actions sont de classe $Bad$, on revient en arrière, i.e. on retourne à la situation précédente et on tente une autre action.

Une fois l'action sélectionnée et appliquée, le système se trouve dans une nouvelle situation. Le système cherche alors à savoir si la situation est de classe $BadSituation$. Si tel est le cas, il revient en arrière et cherche à rejoindre une autre situation. Sinon le processus reprend, i.e. on sélectionne une action et on cherche à l'appliquer. On s'arrête quand on a atteint l'objectif fixé, ce qui doit être vérifié à chaque étape.

### 5.1.5 Un modèle logique de l'adaptation

Nous allons détailler ici le modèle de logique situationnelle que nous avons établi pour les hypermédias adaptatifs. Ce modèle sera considéré comme la base du modèle d'adaptation dans toutes les autres sections de ce chapitre.

Nous avons essentiellement réutilisé l'approche de [63] pour servir de base à notre modèle. Néanmoins, certaines éléments ont dû être modifiés pour obtenir un modèle utilisable dans les hypermédias adaptatifs. En effet, il est notamment impossible de revenir en arrière dans un hypermédia adaptatif : une fois qu'un document est parcouru par un utilisateur, il ne saurait le «désapprendre». Nous ne pouvons donc pas utiliser le prédicat $BadSituation$. Nous ne pouvons pas non plus, si aucune action n'est possible, nous permettre de revenir en arrière pour essayer des actions a priori moins bonnes, comme on le fait dans le domaine de la planification.

De plus, le système présenté dans [63] repose sur des règles heuristiques décrites sous forme

de clauses de Horn, en Prolog ou éventuellement en utilisant un autre raisonneur. Afin de ne pas dépendre d'un raisonneur ou d'un langage de programmation d'une part, et d'offrir un véritable langage de règles pour les hypermédias adaptatifs d'autre part, nous avons défini un langage de règle et un modèle de raisonnement pour ce langage. Nous détaillerons ces éléments dans la partie suivante, qui concerne les méta-règles.

Ainsi, nous définissons notre système de logique situtationnelle comme étant muni des éléments suivants :

- La notion de situation, qui est à la base de toute logique situationnelle.
- Les fluents qui servent à les décrire, à l'aide de règles, l'évolution du profil de l'utilisateur en fonction des actions entreprises par ce dernier.
- La notion d'actions primitives, qui permet de faire évoluer le système. Les actions composites présentent peu d'intérêt dans notre domaine, puisqu'une sélection d'actions possibles est proposée à chaque étape du parcours. Nous préférons reléguer au niveau des règles d'éventuels bouclages (faire refaire des exercices en cas d'échec, par exemple) afin de ne pas compliquer la description des éléments de base du système.
- La notion de degrés de désirabilités. Dans les exemples que nous donnerons dans ce chapitre, nous réutiliserons simplement $bad$, $poss$ et $good$, mais nous permettons de définir autant de degrés que nécessaire, sous la forme $degree(action, situation, integer)$ où $integer$ représente le degré de désirabilité de l'action. 0 est toujours $bad$ et plus l'entier est grand, plus désirable est l'action. Une action est de classe $bad$ si elle n'atteint aucun niveau de désirabilité strictement positif.
- La notion de classement d'action par le biais du prédicat $better$, qui permet d'affiner la classification des actions en plus de la catégorisation donnée ci-dessus. Dans notre système, $better$ sert à classer des actions de n'importe quel degré de désirabilité (il ne servait qu'aux actions possibles dans [63]).

Les fluents et les actions primitives peuvent être définis au cas par cas, pour chaque hypermédia adaptatif à créer : ils font partie des paramètres ajustables. Les degrés de désirabilité sont déterminés par des règles personnalisables, que nous détaillons dans la section suivante.

Le fonctionnement que nous donnons au système est le suivant :

1. Dans la situation courante, soit $i$ le plus haut degré de désirabilité atteint par au moins une action. On propose à l'utilisateur de choisir parmi toutes les actions de degré $i$. On note que l'on peut avoir $i = 0$ dans les « mauvais cas », c'est à dire les cas où les règles ne permettent pas toujours d'avoir toujours une action au moins possible.
2. Une fois l'action effectuée, les données de l'utilisateur sont mises à jour - i.e. les fluents sont modifiés.
3. On vérifie si l'objectif de l'utilisateur est atteint. Si c'est le cas, on arrête. Sinon on reprend à l'étape 1.

Nous avons donc défini un modèle, basé sur la logique situationnelle, qui permet de représenter les mécanismes d'adaptation pour les hypermédias adaptatifs. Cette version de la logique situationnelle peut être visualisée comme une coquille pour encapsuler les données nécessaires à l'adaptation : fluents, actions, règles. L'élément le plus important de ces données est le système de règle spécifique que nous avons défini au dessus de la logique situationnelle, et que nous allons décrire dans la partie suivante.

Pour chaque système d'adaptation construit sur ce modèle, il est nécessaire de construire une interface, capable d'interpréter les actions sélectionnées et classées, et ainsi d'afficher les éléments corrects à présenter à l'utilisateur.

## 5.2 Système de règles

Comme nous venons de le voir dans la partie précédente, notre modèle d'adaptation, comme ceux des hypermédias adaptatifs en général, utilise des règles pour décrire l'adaptation fournie à l'utilisateur final. Les règles permettent de définir une stratégie de parcours adaptée à l'utilisateur : on prend en compte sa position dans l'espace des documents, c'est-à-dire l'état de ses connaissances, mais on peut également utiliser ses préférences personnelles, par exemple le fait qu'il a besoin d'exercices.

Actuellement, les systèmes de règles pour les hypermédias adaptatifs ne prennent pas en compte de l'orthogonalité entre les critères purement relatifs à l'utilisateur (niveau, rapidité, préférences) et les critères liés au domaine des documents.

De plus, les règles utilisées sont les mêmes pour tous les utilisateurs. Si on veut que les règles ne s'appliquent que si l'utilisateur répond à un ensemble de critères donné, il faut rajouter des prémisses dans les règles. Le nombre de ces prémisses risque de devenir important si on veut prendre en compte de nombreux critères pour savoir si une règle doit être appliquée plutôt qu'une autre.

Cette partie a pour but d'étudier les différentes possibilités pour obtenir un mécanisme d'adaptation capable d'adapter les règles appliquées - autrement dit la stratégie de parcours - en fonction de critères relatifs à l'utilisateur, et indépendamment de sa position dans l'espace des documents, et de proposer un langage de règles pour répondre à ce besoin.

Nous avions envisagé, a priori, les pistes suivantes pour la réalisation d'un tel mécanisme :

- possibilité de rédéfinir l'ordonnancement des règles,
- possibilité de définir des règles pointant sur un ensemble de règles,
- possibilité de définir des règles effaçant ou créant de nouvelles règles.

La section 5.2.1 présente un état de l'art sur l'utilisation de méta-règles et de méta-stratégies. Le système présenté dans [50] fait l'objet d'une attention particulière, car il formalise un certain nombre d'aspects qu'on retrouve généralement dans les systèmes à base de règles et de méta-règles, mais aussi car il permet de faire un certain nombre de vérifications importantes.

Après avoir rappelé les caractéristiques des règles présentes par défaut dans la logique situationnelle, nous proposerons une solution au problème posé, basé sur des idées de [50] que nous avons

adaptées à la problématique des hypermédias adaptatifs.

### 5.2.1 Systèmes utilisant des méta-règles

Dans cette partie, nous nous sommes particulièrement intéressés aux systèmes utilisant des méta-règles. Les méta-règles permettent de stratifier la conception d'un système à base de règles sur plusieurs niveaux, en évitant que chacun de ces niveaux ne devienne difficile à mettre en oeuvre et à maintenir. Les méta-règles sont une solution possible à notre problème.

Les méta-règles sont utilisées dans différentes sortes de systèmes. Elles sont notamment très utilisées dans les systèmes experts. On les retrouve également dans les systèmes d'analyses de langages naturels, les systèmes de contrôle, de base de donnée etc. Nous allons passer en revue les différentes catégories de systèmes de méta-règles, avant de nous attarder sur le système de Jagadish [50], qui nous a paru particulièrement intéressant, et sur lequel nous nous baserons dans la dernière partie de ce document pour créer un système de méta-règles spécifique aux hypermédias adaptatifs.

#### Les méta-règles dans les systèmes experts

Le domaine des systèmes experts est certainement celui où les méta-règles sont le plus utilisées. Les méta-règles dans les systèmes experts ont été introduites par Randall Davis en 1980 dans [17]. Dans les systèmes experts, on utilise des méta-règles pour sélectionner un sous-ensemble de règles et les réordonner par ordre de pertinence décroissante, de sorte à en faire intervenir le moins possible dans un problème où le nombre total de règle est très important.

Si, a priori, notre problème ne fera pas intervenir un nombre critique de règles, il nous paraît néanmoins important d'étudier ces méthodes, car elles peuvent être utiles pour sélectionner les règles en fonction du profil de l'utilisateur.
Les règles de base présentées ici sont de la forme :

```
Prémisses => Conclusion
```

En voici un exemple, qui indique que, si on souhaite un investissement à long-terme, dont le niveau de risque est faible et si l'économie s'oriente vers une recession, alors la compagnie PGE est probablement (à 70%) un bon investissement :

```
PREMISE     ($AND (SAME OBJCT TIME-SCALE LONG-TERM)
                   SAME OBJCT RISK-LEVEL LOW-RISK)
                   SAME OBJCT RECESSIONONTHEWAY TRUE))
ACTION     (CONCLUDE OBJCT STOKC-NAME PGE .7)
```

Les méta-règles peuvent sélectionner des règles, ou plus précisément donner des informations sur leur degré d'utilité, ou les réordonner. Deux syntaxes génériques existent donc pour décrire les méta-règles :

```
under conditions A and B,

rules which do {not} mention X [at all|
                                in their premise|
                                in their action]

will [definitely be useless|
      probably be useless|
      ...
      probably be useful|
      definitely be useful]
```

Pour cette sorte de méta-règles, on a d'abord des préconditions (`A` et `B`) pour appliquer la méta-règle, puis des prémisses et une conclusion. Une règle à discriminer est repérée par des éléments mentionnés soit dans ses prémisses, soit dans ses conclusions, soit dans n'importe lequel de ces deux éléments. Les conclusions concernent la pertinence de la règle. Par exemple, une règle *definitely useless* ne devra pas être utilisée.

```
under conditions A and B,
rules which does {not} mention X [at all|
                                  in their premise|
                                  in their action]
should [definitely|
        probably|
        ...
        possibly]
be used [first.|
         last.|
         before|
         after]
rules which do {not} mention X [at all|
                                in their premise|
                                in their action]
```

Cette sorte de méta-règles permet de donner un ordre partiel sur les règles à utiliser. Si une règle mentionne un élément (prémisse ou action) donné, il faut la déclencher en premier, ou avant, ou après... une règle qui mentionne un autre élément donné.

Les méta-règles sont utilisées de la façon suivante :

1. Le système place l'ensemble des règles dans une liste $L$ ;

2. Le système sélectionne les méta-règles utiles pour l'objectif (en fonction des préconditions remplies) ;
3. Chaque méta-règle est appliquée pour déterminer quelles règles doivent être éliminées et quelles règles doivent être placées avant ou après quelles autres règles ;
4. Le système en déduit une liste $L'$, dont les éléments forment un sous-ensemble ordonné - par une relation d'ordre partiel - de l'ensemble des éléments de $L$, et dont l'ordre répond aux critères donnés par les méta-règles. Les règles sont déclenchées par ordre d'utilité décroissante.

Ici nous avons vu un système à deux niveaux de règles : les règles de base et les méta-règles. Rien n'empêche d'envisager des méta-règles de second, troisième... niveau. Elles s'appliqueraient à la sélection de méta-règles dans le premier niveau inférieur au leur.

La sélection de règles ne nécessite en fait pas de méta-règles. Il suffit de placer dans chaque règle les préconditions nécessaires pour que celle-ci ne se déclenche pas dans tel ou tel contexte. L'intérêt des méta-règles pour la suppression de règles de base réside dans le gain de temps procuré par cette méthode - qui évite que toutes les règles ne se déclenchent - ainsi que dans la simplification des règles de base. Notamment, le réordonnancement des règles ne pourrait pas se faire directement dans les règles de base, par exemple en leur ajoutant des prémisses.

Notons que des systèmes de règles très similaires à celui que nous venons de présenter ont été mis en place dans des systèmes d'hypermédias pour adapter la présentation en fonction du média utilisé. Finkelstein présente un tel système dans [24], où les règles de base sont de la forme "trigger, condition, action" (à l'instar d'AHA ! [8]). Elles permettent de sélectionner un jeu de règles différent suivant le média utilisé pour visualiser l'hypermédia.

La prise en compte de critères vagues voire contradictoires dans des systèmes experts a également été considérée et est réalisable notamment grâce aux propositions faites dans [60], où le cas d'un système d'aide à la décision pour les juristes est mis en avant.

Enfin, il est intéressant de noter que [4] apporte une réponse au problème de la création automatique de méta-règles. Les auteurs de cet article présentent un système permettant de créer automatiquement des méta-règles de sélection de règles de niveau de base en utilisant des réseaux de neurones.

Prenons l'exemple suivant :

**(1)** $\alpha_1 \wedge \alpha_2 \Rightarrow \alpha_4$

**(2)** $\alpha_1 \wedge \neg\alpha_3 \Rightarrow \alpha_5 \wedge \alpha_6$

**(3)** $\alpha_1 \wedge \alpha_7 \Rightarrow \alpha_8$

**(4)** $\alpha_4 \wedge \alpha_5 \Rightarrow \alpha_9$

**(5)** $\alpha_4 \Rightarrow \alpha_{10}$

**(6)** $\alpha_5 \wedge \alpha_6 \wedge \alpha_8 \Rightarrow \alpha_{11}$

Le but est de déterminer la valeur de $\alpha_{10}$ connaissant $\alpha_1$, $\alpha_2$ et $\alpha_3$.

Si le système déclenche toutes les règles dans l'ordre, il calculera donc les 4 règles calculables à partir des données de base avant d'obtenir la valeur de $\alpha_{10}$. Un parcours optimal consiste à déclencher la règle **(1)** puis la règle **(5)**. La sélection de règles est donc importante pour des raisons de rapidité d'exécution, et c'est principalement cette problématique qui est traitée dans l'ensemble des systèmes de méta-règles destinés aux systèmes experts.

**Les méta-règles dans les systèmes d'analyse grammaticale**

Dans les systèmes d'analyse grammaticale, les méta-règles sont souvent des règles de réécriture, permettant "d'assouplir" les contraintes posées par les règles de base, quand celles-ci ne sont pas vérifiées.

Les auteurs de [73] présentent un tel système. Les règles de base de ce système sont de la forme : $C_0 \rightarrow C_1,\ C_2,\ ...,\ C_n$

Il s'agit de règles de dominance immédiate. Les $C_i$ sont des types d'éléments grammaticaux comme 'phrase', 'verbe', 'groupe verbal' etc. Ici, $C_0$ est le type du noeud d'un arbre représentant une phrase ou une partie d'une phrase, et les autres $C_i$ sont les types des fils de ce noeud. Plus précisément, le noeud de type $C_0$ doit avoir exactement un fils pour chacun des types $C_1$ à $C_n$.

Ici, les méta-règles permettent de réécrire des règles. Plus précisément une méta-règle est de la forme :

```
'Input rule scheme' (1) => 'Output rule scheme' (2)
```

`Input rule scheme` et `Output rule scheme` sont des règles de base. Si une des règles de l'ensembles des règles en entrée correspond au modèle (1), on ajoute la règle (2) à l'ensemble de ces règles d'entrée.

Par ailleurs, de telles méta-règles entraînent des problèmes similaires à ceux que posent les systèmes de réécriture, notamment en terme de terminaison. Les auteurs de cet article s'intéressent d'ailleurs à ce problème.

Le système de méta-règles proposé ici est utilisé dans des systèmes d'analyse des langages naturels. Les auteurs de [69] présentent un système similaire. Dans ce système, les méta-règles sont utilisées en cours d'exécution du processus d'analyse des entrées d'un système. Lorsque l'exécution échoue, les méta-règles sont déclenchées. Elles analysent l'état du système. En fonction de cette analyse, des actions sont déclenchées. Ces actions permettent la réécriture des contraintes non-respectées par les entrées du système.

**Les méta-règles dans les systèmes de contrôle**

L'auteur de [58] présente succintement l'utilisation de méta-règles dans des systèmes de contrôle. Ici, les règles sont des formules portant sur les paramètres des systèmes. Ces formules étant des heuristiques, elles portent le nom de 'fuzzy' (floues).

Les méta-règles peuvent :

– modifier l'échelle des variables d'entrée du contrôleur
– modifier l'échelle des variables de sortie du contrôleur
– modifier la base de règles

Le fonctionnement détaillé des méta-règles n'est pas fourni. Néanmoins, il est intéressant de noter que les règles modifient les échelles des paramètres.

**Les méta-interpréteurs [57]**

Les méta-interpréteurs ne sont pas des méta-règles. Ils sont des moteurs d'inférences sur les règles ayant des comportements différents les uns des autres, et potentiellement très variés. Le langage dans lequel est implémenté l'interpréteur est appelé méta-langage, et le nouveau langage est appelé langage de niveau de base.

Prenons l'exemple d'un langage de programmation en logique qui ne supporte pas, de manière native, la disjonction. Il est possible de définir un méta-interpréteur dans ce langage qui prenne en compte la disjonction. Dans l'exemple que nous allons donner, `&` représente le "et logique", `v` représente le "ou logique" inclusif, et <= représente l'implication dans le langage que l'on crée. Le méta-langage est un Prolog sans disjonction. Une règle de base est de la forme : `A <= B & (C v A)`. Voici le code d'un méta-interpréteur possible pour résoudre le problème de l'absence de disjonction :

```
prove(true).
prove(A & B) :- prove(A), prove(B).
prove(A v B) :- prove(A).
prove(A v B) :- prove(B).
prove(H) :- built_in(H), call(H).
prove(H) :- (H <= B), prove(B).
```

Prouver `A v B` revient donc à prouver `A`, et, si cela échoue, à prouver `B`. Si un prédicat prédéfini nécessite d'être prouvé, il sera appelé par la procédure `call`, qui permet d'appeler l'éxécution d'un prédicat au niveau méta.

De la même façon, il est possible de définir un méta-interpréteur qui ne cherche pas à prouver une assertion sur plus d'une certaine profondeur de recherche, ou bien un méta-interpréteur qui fasse un parcours en largeur d'abord, ou encore un méta-interpréteur qui remette à plus tard la preuve de certains prédicats.

Les méta-interpréteurs, s'ils ne permettent pas de modifier les règles à proprement parler, permettent d'une part de changer la stratégie de résolution (en largeur, en profondeur d'abord), et d'autre part de modifier le sens qui est donné aux règles (par exemple, en limitant la profondeur de recherche de preuve), de réduire ou d'augmenter leur champs d'action. De plus, on peut imaginer des méta-interpréteurs qui fassent la preuve des règles de différentes manières en tenant compte du profil de l'utilisateur.

En outre, nous pourrions envisager d'utiliser des méta-règles pour sélectionner un méta-interpréteur ou un moteur d'inférence parmi plusieurs dont nous pourrions disposer. Des méta-règles analyseraient alors le profil de l'utilisateur et sélectionneraient une "bonne" façon d'interpréter les règles de base, i.e. un méta-interpréteur.

**Le système de méta-règles de Jagadish [50]**

Il nous a paru particulièrement intéressant de détailler le système de méta-règles présenté par les auteurs de [50]. En effet, il s'agit d'un système formel de méta-règles sur lequel il est possible d'effectuer un certain nombre de vérifications. Il est notamment possible de savoir si une règle ne sera jamais exécutée, si le système est déterministe ou encore si l'ordre obtenu pour les règles à exécuter est unique. L'ensemble de ces calculs est réalisable en temps polynomial en fonction de la taille des ensembles des règles et des méta-règles. Dans cette partie, nous allons décrire en détail ce système, initialement destiné à la mise à jour automatique de bases de données.

*Définitions*

Un système de règles est donné par $S =< V, M >$ où $V$ est un ensemble de règles et $M$ est un ensemble de méta-règles.

Une règle (i.e. un élément de $V$) est constituée d'un déclencheur ou *trigger*, de prémisses et d'une conclusion. À chaque déclencheur correspond un ensemble de règles, l'ensemble $V$ présenté plus haut.

Une règle est dite déclenchable si l'ensemble de ses prémisses sont vrais. On note $I$ le sous-ensemble de $V$ des règles déclenchables dans un état donné du système.

Le sous-ensemble des règles qui sont réellement exécutées, après sélection et ordonnancement par les méta-règles, est noté $O$.

On a la relation triviale suivante : $O \subseteq I \subseteq V$.

*Types de méta-règles*

Il existe quatre types de méta-règles dans l'ensemble $M$. Ces méta-règles permettent le contrôle des intéractions et de l'exécution des règles de $V$. Voici ces quatre types, avec $A, B \in V$ :

- Méta-règles d'exigence positive. On note $A \supset B$. Si $A$ est exécutée, $B$ doit l'être aussi. On dira que "$A$ nécessite $B$".
- Méta-règles d'exclusion mutuelle. On note $\overline{AB}$. Les règles $A$ et $B$ ne peuvent pas être sélectionnées simultanément pour l'exécution. On utilise la notation $\overline{A}$ plutôt que $\overline{AA}$ dans le cas d'une règle mutuellement exclusive d'elle-même, i.e., d'une règle morte.
- Méta-règles de préférence. On note $A > B$. Si $A$ et $B$ sont toutes deux déclenchables, et si elles ne peuvent être exécutées ensemble, alors il faut choisir $A$.
- Méta-règles d'ordonnancement. On note $A \prec B$. Si $A$ et $B$ sont toutes deux sélectionnées pour l'exécution, il faut exécuter $A$ avant $B$.

*Sémantique*

On dit qu'un modèle de $M$ est un quadruplet $< V, I, O, \omega >$, avec $V$ un ensemble de règles, $O \subseteq I \subseteq V$ et $\omega$ une relation d'ordre totale sur $O$. On dit que $< V, I, O, \omega >$ satisfait une méta-règle $\mu$ si :

1. $\mu$ est de la forme $A \supset B$ et si $A \in O$, alors $B \in O$.
2. $\mu$ est de la forme $\overline{AB}$ et $\neg(A \in O \wedge B \in O)$.
3. $\mu$ est de la forme $A > B$ et si $A, B \in I$ et $A \in O$, alors $B \in O$.
4. $\mu$ est de la forme $A \prec B$ et si $A, B \in O$ alors $A$ est avant $B$ dans l'ordre total $\omega$.

N.B. : On a réutilisé dans cette partie les notations $V$, $I$ et $O$ utilisées précédemment pour représenter respectivement les règles du système, les règles déclenchables et les règles à exécuter. Dans la pratique, un modèle de $M$ sera constitué de ces trois sortes d'ensemble, d'où la notation similaire.

*Axiomes d'inférence*

Les axiomes suivants permettent de rendre compte des inférences possibles au niveau des méta-règles. Ils forment un ensemble complet et consistant. Ils permettent de savoir si une méta-règle $\gamma$ peut être déduite d'un ensemble de méta-règles $\Gamma$.

Voici ces axiomes :

- A1 : $\overline{AB} \vdash \overline{BA}$
- A2 : $(A \supset B \wedge \overline{BC}) \vdash \overline{AC}$
- A3 : $\overline{A} \vdash \overline{AB}$
- A4 : $\vdash A \supset A$
- A5 : $(A \supset B \wedge B \supset C) \vdash A \supset C$
- A6 : $\overline{A} \vdash A \supset B$
- A7 : $\vdash A > A$
- A8 : $(A \supset B \wedge C > B) \vdash C > A$
- A9 : $(A \prec B_1 \wedge B_1 \prec B_2 \wedge ... \wedge B_k \prec B) \wedge A \supset X \wedge B \supset Y \vdash A \prec B$, où $X \cup Y = \{B_1, B_2, ...B_k\}$
- A10 : $(B_1 \prec B_2 \wedge ... \wedge B_k \prec B_1) \wedge A \supset X \wedge B \supset Y \vdash \overline{AB}$ où $X \cup Y = \{B_1, B_2, ...B_k\}$
- A11 : $\overline{AB} \vdash A \prec B$

Le problème de l'implication pour les méta-règles, c'est-à-dire le problème qui consiste à vérifier si l'on peut ou non déduire une méta-règle à partir de $M$ et des axiomes **A1** à **A11**, est décidable en temps polynomial.

*Exécutabilité d'une règle*

Une règle $A$ est dite exécutable s'il existe un modèle de $M$ pour lequel l'ensemble $O$ contient $A$. Dans le cas contraire, $A$ est dite morte. $A$ est morte si, et seulement si, on peut déduire $\overline{A}$ de $M$. De ce fait, l'exécutabilité d'une règle est décidable en temps polynomial.

*Déterminisme*

$S =< V, M >$ est déterministe si, pour chaque $I$, il existe un unique $O$ maximal muni d'un ordre total $\omega$ et tel que $< V, I, O, \omega >$ est un modèle pour $M$. Le déterminisme d'un système est

obtenu en temps polynomial en fonction de la taille de $M$.

Le problème du déterminisme est dû aux règles d'exclusions mutuelles. Si deux règles sont déclenchables mais ne peuvent être ensembles dans l'ensemble d'exécution, peut-on toujours choisir entre les deux, i.e., a-t-on toujours une relation de préférence entre les deux règles ?

Voici la méthode utilisée pour calculer rapidement si un système est déterministe. Il est important de noter que ce mécanisme ne fait pas intervenir les règles, mais uniquement les méta-règles.

Tout d'abord, on réduit le système de règles :

- On construit le graphe de précédence de $M$ pour la relation d'exigence positive. Il existe un arc reliant le noeud $A$ au noeud $B$, et seulement si, $A \supset B$.
- Dans ce graphe, on cherche les composants fortement connexes, i.e. les ensembles maximaux de noeuds tels que pour chaque couple de noeuds de cet ensemble, il existe un chemin de l'un à l'autre dans les deux sens. On nomme chaque composante connexe de manière distincte des autres. On note qu'il est possible de trouver des composantes connexes dans ce graphe de précédence grâce à l'axiome **A4**, qui relie chaque noeud à lui-même.
- On remplace, dans toutes les méta-règle, chaque règle par la composante connexe, représentée par son nom, à laquelle appartient cet élément. Finalement, on élimine les règles en double.
- Le système muni de ces nouvelles méta-règles est appelé système réduit.
- Enfin, on analyse l'exécutabilité des règles de $V$ dans le système réduit et on élimine les règles mortes.

Cette réduction permet de traiter les cycles de règles plus rapidement. Ainsi, les règles de chaque composante fortement connexe sont soit toutes dans $O$, soit toutes absentes de $O$. Le calcul des composantes fortement connexes se fait en temps linéaire en fonction du nombre de noeuds et d'arc, i.e. en fonction du nombre de règles et de méta-règle d'exigence positive.

On effectue ensuite une deuxième étape de traitement, en calculant l'ensemble minimal de méta-règles d'exclusion mutuelle. On considère que $\overline{AB}$ et $\overline{BA}$ sont identiques. Si $\mu \in M$ est une règle d'exclusion mutuelle, et qu'elle peut être déduite de $M - \{\mu\}$, on supprime $\mu$ de $M$. On appelle $D$ le sous ensemble de $M$ constitué uniquement des méta-règles d'exclusion mutuelle restantes. On prouve que $D$ est unique et toujours défini.

Enfin, un système réduit $S$ est déterministe si, et seulement si, dans le système réduit, pour toute méta-règle d'exclusion mutuelle $\overline{AB}$ de $D$, on a soit $M \vdash (A > B)$, soit $M \vdash (B > A)$.

Le temps de calcul de cette méthode est polynomial en fonction de la taille de $M$.

*Ordonnancement*

Un système déterministe est dit bien ordonné si, et seulement si, $M$ implique un unique ordre total $\omega$ sur $O$. Un système est donc bien ordonné si, et seulement si, pour tout couple $A, B \in V$, on a soit $M \vdash (A \prec B)$, soit $M \vdash (B \prec A)$.

Comme le problème de l'implication est décidable en temps polynomial (cf. 5.2.1), le problème du bon ordonnancement l'est aussi.

*Séquence d'exécution*

Une fois toutes les vérifications nécessaires effectuées, il faut pouvoir calculer, pour $I$ donné, l'ensemble ordonné $O$ des règles à exécuter. L'algorithme proposé est le suivant :

1. Initialiser $O$ comme étant égal à $I$ ;
2. Pour tout couple de règles $A, B$ tels que $A \in O$ et $B \notin O$, si $M \vdash (A \supset B)$, alors retirer $A$ de $O$, puis retirer toute règle $C \in O$ telle que $M \vdash (A > C)$ ;
3. Identifier l'ensemble $D$ des règles minimales d'exclusion mutuelle ;
4. Le système étant supposé déterministe, pour chaque méta-règle $\overline{AB}$ de $D$, on sait si $A$ est préférable à $B$ ou si c'est le contraire. Supposons $M \vdash (A > B)$ :
   - Supprimer $B$ de $O$ ;
   - Supprimer toute règle $C \in O$ telle que soit $M \vdash (C \supset B)$, soit $M \vdash (B > C)$.
5. Établir un ordre total sur $O$ en utilisant les méta-règles de type $A \prec B$.

*Conclusions sur le système de Jagadish*

Le système que nous venons de présenter offre une approche déclarative pour la sélection et l'ordonnancement de règles. Ce système permet de faire un certain nombre de vérifications sur les méta-règles, évitant dans la mesure du possible, c'est-à-dire dans la mesure où les règles sont correctement définies, de pouvoir commettre des erreurs. Ainsi, il nous a paru intéressant, comme nous le verrons dans la prochaine section, de réutiliser ce système pour fournir un système de méta-règles destiné aux hypermédias adaptatifs.

**Conclusions sur les possibilités d'utilisation de méta-règles**

Les méta-règles sont utilisées dans des domaines bien plus variés que l'informatique, notamment en philosophie (cf. [55]), en économie (cf. [61]) ou en droit (cf. [60]). En informatique, elles sont couramment utilisées, depuis plus de vingt ans, dans les systèmes manipulant des règles. Les méta-règles sont d'autant plus présentes que le nombre de règle dans le système est grand, et c'est pourquoi on en retrouve dans la plupart des systèmes experts. Ces méta-règles permettent de séparer différents niveaux de raisonnement, de créer des couches de raisonnement dans le système. Les principaux objectifs sont :

- Réduire le temps d'exécution du système face à un problème donné ;
- Séparer des niveaux de raisonnement de sorte à faciliter la création et la maintenance du système ;
- Donner plus de puissance à règles de base que le système de règle initial, sans avoir à modifier ses règles, c'est-à-dire le plus souvent à les rendre plus complexes.

Nous avons rencontré dans toute cette première partie différentes sortes de méta-règles :

- Les méta-règles qui sélectionnent les règles à appliquer. Ces méta-règles peuvent analyser les prémisses et les conclusions des règles de base pour sélectionner les règles à déclencher.

- Les méta-règles qui réordonnent les règles. Dans les systèmes experts, le réordonnancement est essentiellement effectué pour déclencher en priorité les règles qui paraissent les plus pertinentes. On peut imaginer que, dans un système plus petit, le réordonnancement pourra avoir pour but de modifier le sens du jeu de règles.
- Les méta-règles qui ajoutent des règles en fonction des règles déclenchées. Ces méta-règles sont les plus difficiles à mettre en place, et, de fait, les plus rarement rencontrées.
- Les méta-règles qui modifient à la fois les règles de base et les entrées utilisées par les règles.

Le système de Jagadish [50] retient particulièrement notre attention car il propose non seulement un système formel et déclaratif de règles utilisant des méta-règles, mais aussi car on peut aisément raisonner sur un tel formalisme, et notamment vérifier un certain nombre de propriétés du système, comme son déterminisme ou son bon ordonnancement.

Nous allons nous attacher, dans la suite de ce document, à expliquer les différences entre les règles manipulées dans [50] et celles utilisées dans nos systèmes d'hypermédias adaptatifs, puis à montrer comment on peut construire une variante du système de Jagadish spécialisée pour nos besoins, et permettant au moins de vérifier les mêmes propriétés que celles mises en avant par Jagadish.

## 5.3 Règles dans la logique situationnelle

Dans cette partie, nous rappelons brièvement le système de règle utilisé par défaut dans la logique situationnelle, et que nous avons un temps utilisé. Dans la partie 5.4 de ce document, nous réutiliserons certaines caractéristiques fondamentales de ce système de règles pour construire un système à base de méta-règles.

Dans la logique situationnelle, les règles d'adaptation utilisées sont des clauses de Horn.

Pour les hypermédias adaptatifs, on peut utiliser les prémisses suivants dans les règles d'adaptation :

- Une interrogation des observateurs de la logique situationnelle, qui permettent de connaître la valeur d'un paramètre de la situation courante.
- Une interrogation de la base des données, qu'elle soit relative à l'utilisateur, au domaine des documents, ou aux relations entre les deux. Une interrogation de la base de donnée se fait par le biais de prédicats.

Ces règles sont toutes utilisées, sans ordonnancement particulier, pour déterminer les degrés de désirabilité des actions. Les règles sont donc parcourues, après chaque action, une et une seule fois par action potentielle dans un ordre donné quelconque, invariant et sans influence sur le résultat. On notera particulièrement que les règles sont toutes exécutées à chaque fois, pour tous les types d'utilisateurs et pour chaque action potentielle.

Exemple de règle d'adaptation :

```
good(lire(Doc)) :-
        not(connait(user, Doc)),
        prerequisConnus(Doc),
        objectif(Obj),
        mene(Doc, Obj).
```

On fait d'abord appel à la base de données concernant l'utilisateur avec :
`not(connait(User, Doc))`,
puis à deux prédicats, `prerequisConnus` qui calculent si tous les pré-requis de Doc sont connus, et `mene` qui calcule si `Doc` est pré-requis direct ou indirect de `Obj`. Enfin, on fait un nouvel appel à la base de données pour savoir quels sont les éléments qui sont des objectifs.

Les deux prédicats ne déclenchent aucune règle, et se terminent. `prerequisConnus` remonte le graphe des pré-requis d'un niveau et pour chaque noeud rencontré fait un appel à la base de données. `mene` parcourt le graphe des pré-requis depuis `Doc` et voit si un chemin mène à `Obj`. En s'assurant de ne pas parcourir de boucle dans le graphe, on obtient un prédicat qui se termine dans tous les cas.

L'ensemble des règles utilisées forme une stratégie de parcours, et le système de méta-règle que nous présenterons dans la section suivante prend en compte cette spécificité : les ensembles de règles sélectionnés constitueront, à proprement parler, une stratégie de parcours, adaptée à l'utilisateur.

## 5.4 Un système de méta-règles adapté aux hypermédias adaptatifs

Dans cette partie, nous allons présenter un système formel de règles utilisant des méta-règles et conçu spécifiquement pour les hypermédias adapatatifs. Ce système est basé sur certaines idées du système de Jagadish [50]. Dans un premier temps, nous présentons les caractéristiques que nos méta-règles doivent réunir : de quoi sont-elles capables, quel résultat doit-on obtenir ? Après avoir étudié les incompatibilités de nos objectifs avec le système initial de Jagadish, nous présentons un système formel de méta-règles pour les hypermédias adaptatifs. Nous montrons ensuite différentes propriétés de notre système. Enfin, nous donnons un exemple détaillé illustrant l'utilité de ce système.

### 5.4.1 Problématique

Comme nous l'avons expliqué, l'ensemble des règles dans un système d'hypermédia adaptatif forme une stratégie de parcours. Sans méta-règle, cette stratégie de parcours est unique pour l'ensemble des utilisateurs. Par exemple, on privilégie un parcours en profondeur d'abord, en abordant chaque document inconnu de l'utilisateur courant.

Partant de cette constatation, l'utilisation d'un système de règles sans méta-règles amène nécessairement à deux cas de figure possibles :

- soit on écrit des règles utilisant peu les données relatives uniquement au profil de l'utilisateur, et l'adaptation se fait alors essentiellement autour des relations entre l'utilisateur et le domaine - l'état de ses connaissances du domaine. La stratégie appliquée est donc globalement la même pour tous les types d'utilisateurs ;
- soit on écrit des règles prenant en compte beaucoup de données relatives à l'utilisateur. Le nombre de prémisses de chaque règle peut alors devenir très important, tout comme le nombre de règles. Les éléments relatifs au profil intrinsèque de l'utilisateur et au positionnement de l'utilisateur dans le domaine des documents se retrouvent au même niveau dans les règles, alors qu'ils sont sémantiquement très différents. Il devient difficile de prévoir quelles seront les règles effectivement déclenchées pour un type donné d'utilisateur, et leur priorité reste non modifiable quelque soit le cas de figure, c'est-à-dire que, dans notre cas particulier, aucune règle n'a de priorité particulière. Les différentes stratégie de parcours possibles sont complètement masquées et il est impossible d'ajouter une règle simplement.

Être en mesure d'utiliser des stratégies de parcours différentes pour des types d'utilisateur différents nous paraît essentiel pour rendre les systèmes adaptatifs beaucoup plus adaptatifs qu'ils ne le sont déjà. Si le deuxième cas de figure présenté ci-dessus permet en partie d'adapter la stratégie en fonction de l'utilisateur, sa mise en place est particulièrement délicate au vu du nombre de prémisses nécessaires, et le problème de changement de priorité des règles n'est pas résolu. De plus, nous ne souhaitons pas introduire des règles de type condition/action, qui permettent de donner des priorités différentes aux règles, car ce formalisme pose d'autres problèmes, comme nous l'avons vu dans le chapitre 2.

Il nous paraît donc intéressant de créer un système de règles utilisant des méta-règles qui permettront de :

- prendre en compte la différence entre le niveau purement relatif à l'utilisateur (capacités, méthodes d'apprentissage, niveau global...), le niveau relatif à la position de l'utilisateur dans le domaine des documents (documents lus, concepts acquis...) et le niveau relatif à la «forme» du domaine, et les séparer clairement.
- permettre la sélection automatique d'un sous-ensemble de l'ensemble de toutes les règles disponibles, et faire en sorte que ce sous-ensemble constitue une véritable stratégie de parcours ;
- donner des priorités différentes aux règles selon le type d'utilisateur, de sorte à désambiguïser certains choix de stratégie de parcours.

Une fois le système de règles clairement établi, nous donnerons un mécanisme de déduction permettant de raisonner de manière formelle et universellement compréhensible sur les règles que nous aurons décrit.

Pour créer un tel système de règles, nous allons nous appuyer sur les résultats de [50]. La pro-

chaine partie sera consacrée à l'étude des différences caractéristiques de ce système et de celui que l'on souhaite obtenir, et la partie suivante définira un système de méta-règles destiné aux hypermédias adaptatifs.

### 5.4.2 Incompatibilités avec le système de Jagadish

Le système de Jagadish est utilisé dans le cadre des systèmes de gestion de bases de données. Quand un évènement est déclenché, on sélectionne une série de règles. C'est sur cet ensemble de règles que sont appliquées les méta-règles. Plus précisément, les méta-règles permettent de sélectionner des règles parmi les règles déclenchables, c'est-à-dire celles dont tous les prémisses sont vrais.

Dans un hypermédia adaptatif, quand on sélectionne un ensemble de règles de stratégie de parcours, chaque règle doit être appliquée à toutes les actions potentielle dans la situation courante, afin de déterminer quelles actions sont bonnes, possibles, ou meilleures que d'autres actions. En fonction de l'action testée, la règle réussira, dans d'autres cas elle échouera, donnant ainsi une indication sur le degré de désirabilité de l'action en question. Déterminer si tous les prémisses d'une règle sont vrais dans une situation donnée n'a donc pas de sens ici, puisque les prémisses dépendent de l'action testée, et puisqu'il y a plus d'une action à tester dans une situation donnée.

De plus, les règles dans [50] sont censées mettre à jour des données dans une base, et ce en appliquant un certain nombre de calculs consécutifs. Dans un hypermédia adaptatif, le but d'une présélection par utilisation de méta-règles, comme nous l'avons décidé (cf. 5.3), est de sélectionner un ensemble de règles formant ensemble une stratégie de parcours à part entière. L'ordre des règles dans le système de Jagadish est donc très important, et la relation d'ordonnancement y est indispensable. Dans la logique situationnelle de base, on applique l'ensemble des règles, et chaque action qui est de degré de désirabilité maximal - c'est-à-dire défini comme tel par une règle au moins - est retenue. On peut ensuite classer ces actions. Il y a donc, dans ce type de système, une notion de priorité, mais pas de notion d'ordre d'exécution.

### 5.4.3 Description d'un système de règles pour les hypermédias adaptatifs

Nous allons maintenant présenter en détail le système formel que nous proposons pour répondre au problème que nous nous sommes posé. Ce système est basé sur le système de Jagadish. Il prend en compte l'ensemble des spécificités décrites dans les deux parties précédentes. Nous allons tout d'abord définir l'ensemble de ce système, puis montrer de quelle façon les propriétés vérifiables dans le système de Jagadish peuvent l'être dans le nôtre, et nous terminerons en élargissant l'ensemble de ces propriétés.

#### Définition générale

On appelle système de règle un quadruplet $S =< V, F, M, \delta >$ où :

- $V$ est un ensemble de règles de stratégie de parcours ;
- $F$ est un ensemble de critères de déclenchement ;
- $M$ est un ensemble de méta-règles ;
- $\delta$ est une fonction d'association entre les règles de $V$ et les critères de $F$.

Nous allons maintenant détailler la nature de chaque élément du quadruplet $S$.

**Ensemble des règles de parcours**

$V$ est un ensemble de règles de la forme :
$R : \forall X \in E(\bigwedge_{i \in I} P_i(X) \Rightarrow Deg(Action(X), d)$

Les notations ont les significations suivante :
$I$ est ensemble fini d'indices.

$R$ est l'identifiant de la règle.

$d$ est le degré de désirabilité de l'action.

$E$ est un ensemble de constituants du domaine de l'une des formes suivantes :

- $E$ est de la forme $\{X/type(X, \_)\}$, c'est-à-dire représente l'ensemble des constituants d'un type donné.
- $E$ est de la forme $\{X/X\Re\_\}$ ou $\{X/\_\Re X\}$, c'est-à-dire représente l'ensemble des constituants en relation avec un constituant donné.
- $E$ est égal à la constante $Cst$, qui représente par définition l'ensemble des constituants du domaine.

$P_i(X)$, prémisse de la règle, est de la forme :

- $P_i(X) = X\Re\_$ ou $\_\Re X$, c'est-à-dire que la prémisse représente le fait que $X$ est en relation avec un constituant.
- $P_i(X) = X\Re^*\_$ ou $\_\Re^* X$, c'est-à-dire que la prémisse représente le fait que $X$ est en relation transitivement avec un constituant.
- $P_i(X) = X\Re^n\_$ ou $\_\Re^n X$, c'est-à-dire que la prémisse représente le fait que $X$ est en relation transitivement avec un constituant, et que le chemin transitif reliant $X$ et le constituant ne dépasse pas $n$ en longueur.
- $P_i(X) = metadata(X, property, value)$, c'est à dire que la prémisse teste une valeur d'une méta-donnée associée au constituant $X$.
- $P_i(X) = \forall Y \in F(\bigwedge_{j \in J} P_j(X, Y))$, où $F$ de la forme de $E$, où $J$ de la forme de $I$ et où $P_j(X, Y)$ de la forme de $P_i(X)$, mais dépendant de $X$ et $Y$. Ainsi, un prémisse peut-être lui-même une conjonction quantifiée universellement de prémisses.
- $P_i(X) = \exists Y \in F(\bigwedge_{j \in J} P_j(X, Y))$, où $F$ de la forme de $E$, où $J$ de la forme de $I$, et où $P_j(X, Y)$ de la forme de $P_i(X)$, dépendant de $X$ et $Y$. Ainsi, un prémisse peut-être lui-même une conjonction quantifiée existentiellement de prémisses.

Les différents prémisses font intervenir uniquement des prédicats relatifs aux relations entre des éléments du domaine des documents ou à leurs métadonnées. En aucun cas ces prédicats ne peuvent faire intervenir des données de l'utilisateur, d'où les restrictions de types imposées sur la syntaxe de nos règles de base : il s'agit de règles de parcours du domaine, indépendamment de l'utilisateur.

De cette façon, les règles de parcours adaptent le parcours uniquement en fonction du domaine. Aucun critère relatif aux capacités, aux connaissances et aux préférences de l'utilisateur n'est pris en compte à ce niveau.

**Ensemble des critères de déclenchement**

Comme nous l'avons expliqué, il n'est pas envisageable, pour une règles de parcours, de savoir si l'ensemble de ses prémisses sont vrais dans une situation donnée, puisque chaque règle est appliquée à plusieurs actions potentielles. Nous avons décidé de séparer explicitement les critères de déclenchement des règles, pour palier à cette impossibilité, dans notre système de règles de base, de déterminer si tous les prémisses d'une règle sont vrais ou faux. Ces critères de déclenchement doivent faire intervenir uniquement le modèle de l'utilisateur. En aucun cas ces critères ne doivent faire intervenir le domaine des documents ou la position de l'utilisateur dans ce domaine.

Ainsi, nous séparons les deux aspects "domaine" et "utilisateur" comme nous le souhaitions. Les règles de $V$ sont exclusivement dévolues au choix du parcours en fonction du positionnement de l'utilisateur dans ce parcours. Les critères de $F$ permettront de présélectionner des lots de règles déclenchables pour un type d'utilisateur donné.

Exemple :

```
vitesse_apprentissage = rapide
```

Où `vitesse_apprentissage` est une propriété du profil de l'utilisateur, et rapide une valeur associée à cette propriété.

**Association règles - critères de déclenchement**

Afin d'associer les règles à des critères de déclenchement, on utilise la fonction $\delta$. À chaque règle $r \in V$, $\delta$ associe un ensemble de parties de $F$. On dit alors que $r$ est déclenchable si, et seulement si, il existe un élément $C \in \delta(r)$ tel que pour tout critère $c \in C$, $c$ est vérifié pour l'utilisateur courant.

Grâce à cet opérateur, on redéfinit la notion de déclenchement qui, selon la définition de Jagadish, n'aurait pas été calculable dans notre système d'hypermédia adaptatif.

Dans la pratique, et afin d'allèger la notation, on n'écrira pas :

$\delta(r) = \{ \{c_1, c_2\}, \{c_3, c_5, c_6\}, ...\}$

mais :

$\delta(r) = \{c_1, c_2\}$

$\delta(r) = \{c_3, c_5, c_6\}$

...

Dans cet exemple, si $c_1$ et $c_2$ sont vrais, alors $r$ est déclenchable.

La fonction $\delta$ permet donc de faire le lien entre les critères basés sur les domaines orthogonaux de l'utilisateur et des documents.

On définit, à partir de $\delta$, la fonction $\gamma$, qui joue en quelque sorte le rôle de fonction réciproque. De façon informelle, $\gamma(c)$ est l'ensemble des règles $r$ pour lesquelles $c$ est un critère de déclenchement associé. Dans l'exemple précédent, $r$ est dans $\gamma(c_3)$, $\gamma(c_1)$, ... $\gamma$ nous permet de traiter des ensembles de règles associées à un critère. $\gamma$ va de $F$ dans l'ensemble des parties de $V$, et est définie de la façon suivante :
Soit $r \in V$ et $c \in F$. $r \in \gamma(c)$ si, et seulement si, il existe $C \in \delta(r)$ tel que $c \in C$.

**Méta-règles**

*Réutilisation du système de Jagadish*

Une fois les critères de déclenchement définis, il nous est possible d'utiliser les même méta-règles que dans le système de Jagadish. Il suffit, dans une situation donnée, de construire l'ensemble $I$ des règles d'entrée comme étant l'ensemble des règles déclenchables au sens de $\delta$. Une fois que $I$ est construit, on applique les règles d'inférence comme dans [50]. On peut vouloir exprimer le fait qu'une règle nécessite une autre règle pour pouvoir être exécutée, que deux règles ne doivent pas être exécutées ensemble, qu'une règle doit être préférée à une autre.

Comme nous l'avons vu dans la partie 5.4.2, donner un ordre d'exécution n'a pas de sens pour nos règles. Néanmoins, on peut utiliser une relation d'ordonnancement afin de donner des priorités entre les règles. Plusieurs pistes concernant la façon dont on pourrait utiliser cette relation d'ordonnancement sont étudiées dans la partie 5.4.5.

Dans la partie précédente, nous avons vu qu'il est possible d'associer des règles à des critères de déclenchement. Ces critères de déclenchement correspondent à des caractéristiques de l'utilisateur. À chaque critère correspond un certain nombre de règles, qui forment un sous-ensemble d'une stratégie de parcours correspondant à ce critère. Il nous a donc paru important, pour simplifier la conception des méta-règles, de pouvoir traiter directement des ensembles de méta-règles.

Dans le paragraphe suivant, nous allons définir des méta-règles au niveau des ensembles de règles. La définition de ces nouvelles méta-règle fait intervenir les définitions des différents types de méta-règles du système de Jagadish. Il sera donc toujours possible de se ramener au cas de ce système.

*Méta-règles ensemblistes*

Dans cette partie, nous allons définir une version ensembliste des méta-règles de Jagadish. Ces méta-règles s'appuient sur les méta-règles de Jagadish, que nous appellerons également méta-règles individuelles, par oppositions aux nouvelles méta-règles ensemblistes que nous allons définir. Dans toute cette partie, on a $A, B \subseteq V$.

Le principal problème posé par l'adjonction d'une notation ensembliste est le problème de l'intersection éventuelle des ensembles mis en relation. L'approche ensembliste ne pose pas vraiment problème pour les méta-règles de type exigence positive et exclusion mutuelle. Dans ces cas là, les couples de règles de l'intersection des ensembles en relation seront respectivement nécessaire l'une par rapport à l'autre dans les deux sens (et on aura $\alpha \supset \alpha$ ce qui aurait de toutes façons prouvé par **A4**), et mutuellement exclusives d'elles-même i.e. mortes.

Dans le cas de la relation de préférence, il serait bon de ne pas inférer à la fois $\alpha > \beta$ et $\beta > \alpha$ si $\alpha \neq \beta$, pour maintenir le déterminisme du système. Nous avons pu trouver une définition de $A > B$ qui évite ce problème, en donnant une relation de préférence pour tous les couples d'éléments de $A$ et de $B$, sauf pour les couples dont les deux éléments sont dans $A \cap B$.

Nous avons tenté de faire de même avec la relation d'ordonnancement, mais le résultat était incompatible avec la formule **F11** que nous présenterons dans la partie 5.4.4. Nous avons donc décidé que, dans le cas de la relation d'ordonnancement, il est possible d'inférer $\alpha \prec \beta$ et $\beta \prec \alpha$ pour $\alpha, \beta \in A \cap B$, ce qui, par application de **A10**, donne $\overline{\alpha\beta}$ : les éléments de l'intersection sont désactivés.

On définit les opérateurs portant sur les ensembles de règle à partir des opérateurs portant sur les règles individuelles de la façon suivante :

- $(A \supset B) \Leftrightarrow (\forall\alpha \in A,\ \forall\beta \in B,\ \alpha \supset \beta)$. $A$ nécessite $B$ si chaque élément de $A$ nécessite tous les éléments de $B$.
- $\overline{AB} \Leftrightarrow (\forall\alpha \in A,\ \forall\beta \in B,\ \overline{\alpha\beta})$. Les règles de $A$ sont mutuellement exclusives des règles de $B$. Si $\tau \in A$ et $\tau \in B$ alors $\overline{\tau}$, i.e. $\tau$ est une règle morte.
- $(A > B) \Leftrightarrow ((\forall\alpha \in A\backslash B,\ \forall\beta \in B,\ \alpha > \beta) \wedge (\forall\alpha \in A,\ \forall\beta \in B\backslash A,\ \alpha > \beta))$. $A$ est préférable à $B$ si,pour chaque couple $(\alpha, \beta)$ de $A \times B$ qui n'est pas dans $(A \cap B)^2$, on a $\alpha > \beta$. Le fait de ne pas tenir compte des couples d'éléments éventuels de l'intersection des deux ensembles permet d'éviter de conclure, sur les règles de base, un relation et la relation contraire en même temps, i.e. de former un système non-déterministe.
- $(A \prec B) \Leftrightarrow (\forall\alpha \in A,\ \forall\beta \in B,\ \alpha \prec \beta)$ $A$ est avant $B$ si chaque élément de $A$ est avant chaque élément de $B$.

Dans la pratique, on utilisera essentiellement deux types d'ensembles dans les méta-règles :

- Des ensembles de plusieurs règles de la forme $\gamma(c)$ où $c \in F$. Il s'agit de règles associées à un critère.
- Des singletons de la forme $\{r\}$ où $r \in V$. Dans ce cas, les méta-règles de notre système seront les mêmes que dans le système de Jagadish.

*Conclusions*

Nous venons de définir un système de règles, muni de méta-règles, et spécialement conçu pour les hypermédias adaptatifs. Les caractéristiques intrinsèques de l'utilisateur sont clairement séparées des caractéristiques liant l'utilisateur aux documents. Il est possible, par le biais des méta-règles, de

définir des relations entre des groupes de règles correspondant à différents critères distinctifs de l'utilisateur.

Il nous faut maintenant étudier les possibilités offertes par ce système. Dans la section suivante, nous montrerons que l'on peut déduire des axiomes d'inférence de [50] des formules d'inférence portant sur les méta-règles ensemblistes, et nous étudierons les conséquences que nous pouvons en tirer. Ensuite, nous montrerons par un exemple l'utilisation concrète que l'on peut faire de notre système.

### 5.4.4 Propriétés du système

Pour faire des inférences sur notre système de méta-règles ensemblistes, deux possibilités s'offrent à nous. Soit on transforme chaque méta-règle ensembliste en méta-règle individuelle, et on utilise les algorithmes décrits dans [50]. Soit on définit un système d'inférences portant - en partie - sur les méta-règles ensemblistes.

Lors de la phase d'implémentation, nous testerons ces deux solutions. Nous allons maintenant décrire un système d'inférence portant sur les méta-règles ensemblistes.

Nous repartons des axiomes du système de Jagadish et montrons que, selon les cas, la même formule d'inférence écrite en remplaçant les règles de base par des ensembles de règles, est prouvable ou non à partir des axiomes **A1** à **A11**. Ensuite nous montrerons de quelle façon on peut prouver qu'un système de règles donné avec nos définitions est déterministe et bien ordonné. Enfin, nous montrerons quelles autres vérifications nous pouvons faire à partir de notre système.

#### Formules d'inférence

Dans cette partie, nous allons "transposer" les axiomes **A1** à **A11** du système de Jagadish, en remplaçant les règles par des ensembles de règles. Dans toute cette partie, $A$, $B$ et $C$ sont des ensembles de règles.

**F1** : Montrons que $\overline{AB} \vdash \overline{BA}$. Soit $\alpha \in A, \beta \in B$ quelconques. On a $\overline{AB}$ donc on a (au sens du système de Jagadish) $\overline{\alpha\beta}$. D'après A1 on a donc $\overline{\beta\alpha}$. $\alpha$ et $\beta$ sont quelconques donc : $\forall\alpha \in A,\ \forall\beta \in B,\ \overline{\beta\alpha}$, ce qui est équivalent, par définition, à $\overline{BA}$. On a donc : $\overline{AB} \vdash \overline{BA}$.

**F2** : Montrons que $(A \supset B \wedge \overline{BC}) \vdash \overline{AC}$. Soit $\alpha \in A, \beta \in B, \tau \in C$ quelconques. On a, par définition, $\alpha \supset \beta$ et $\overline{\beta\tau}$ donc, d'après A2, on a $\overline{\alpha\tau}$. $\alpha$, $\beta$ et $\tau$ sont quelconques donc on a $\overline{AC}$ et par conséquent : $(A \supset B \wedge \overline{BC}) \vdash \overline{AC}$.

**F3** : Montrons que $\overline{A} \vdash \overline{AB}$. Soit $\alpha \in A, \beta \in B$ quelconques. On a $\overline{\alpha}$ donc on a $\overline{\alpha\beta}$. On a donc $\overline{AB}$ et par conséquent : $\overline{A} \vdash \overline{AB}$.

**AF4** : Montrons que $\neg(\vdash A \supset A)$. On a évidemment AF4, puisque sinon toute règle nécessiterait toute autre règle (on aurait $V \supset V$). Dans le système de Jagadish, A4 sert à la construction d'un graphe de précédence pour la relation d'exigence positive dans lequel on peut dégager des graphes fortement connexes, ce qui implique que chaque noeud mène à lui même, ainsi que pour la transitivité de la relation d'ordonnancement, dont nous devrons affiner le traitement. Nous décidons de déplacer le processus de formation des $\alpha \supset \alpha$ au début de l'étape suivante.

**F5** : Montrons que $(A \supset B \wedge B \supset C) \vdash A \supset C$. Très simplement, avec $\alpha \in A$, $\beta \in B$ et $\tau \in C$, on a $\alpha \supset \beta \wedge \beta \supset \alpha$ donc d'après A5 on a $\alpha \supset \tau$. F5 est donc bien déduite.

**F6** : Montrons que $\overline{A} \vdash A \supset B$. Si on a $\overline{A}$, pour tout $\alpha \in A$ on a $\overline{\alpha}$. Donc pour tout $\beta \in B$, on a $\alpha \supset \beta$. Donc on a bien $A \supset B$. F6 est déduite.

**F7** : Montrons que $\vdash A > A$. $A \backslash A = \emptyset$ donc, trivialement, on a $A > A$. F7 est déduite.

**F8** : Montrons que $(A \supset B \wedge C > B) \vdash C > A$. Soit $\alpha \in A$ et $\tau \in C$ tels que $\alpha$ et $\tau$ ne sont pas tous deux dans $C \cap A$. Soit $\tau \in B$ et alors $\alpha \supset \tau$ par application de A7 et A8. Sinon, $\tau \notin B$, donc il existe $\beta \in B$ tel que $\tau > \beta$ par définition de la relation de préférence. De plus $\alpha \supset \beta$. Donc on a également $\alpha \supset \tau$. F8 est déduite.

Cet ensemble de démonstrations prouve que l'on peut effectuer certaines inférences sur les méta-règles ensemblistes.

Néanmoins, la seule application de ces formules ne suffit pas à inférer l'ensemble des méta-règles individuelles que l'on peut inférer en repassant d'abord par la notation individuelle. Par exemple, si on a $A \supset B$ et $C \supset D$, soit $\alpha \in A,\ \beta \in B \cap C,\ \delta \in D$, avec **A5**, on infère $\alpha \supset \delta$, ce qui n'est pas toujours le cas en faisant uniquement les inférences au niveau ensembliste. Autre exemple, on peut prouver à la fois $\alpha > \tau$ et $\tau > \alpha$ si $C \subset A$ et $B \cap A = \emptyset$ avec $\alpha$ et $\tau$ dans $C$ en utilisant **A8** mais pas en utilisant seulement **F8**, ce qui pose des problèmes, entre autres, pour savoir si le système est déterministe.

Nous proposons d'utiliser les formules **F1** à **F8** pour accélérer le calcul. Nous détaillerons la manière de procéder après avoir traiter **F9** à **F12**

Les axiomes **A9** à **A11** traitent de la relation d'ordonnancement. **A9** et **A10** traitent de sa transitivité. Dans le système de Jagadish, la transitivité de la relation d'ordonnancement est vérifiée si l'ensemble des étapes intermédiaires entre deux extrêmités d'une suite de relations d'ordonnancements sont aussi dans l'ensemble d'exécution. De plus, si deux règles requièrent deux ensembles de règles dont l'union est cyclique pour la relation d'ordonnancement, elles ne peuvent être exécutées en même temps.

Vérifions que l'on peut transposer **A9**, **A10** et **A11** en écriture ensembliste :

**F9** : A-t-on $(A \prec B_1 \wedge B_1 \prec B_2 \wedge ... \wedge B_k \prec B) \wedge A \supset X \wedge B \supset Y \vdash A \prec B$, où $X \cup Y = \{B_1, B_2, ...B_k\} \subset E(V)$ ? Soit $\alpha \in A,\ \beta_1 in B_1,\ \beta_2 \in B_2, \ldots,\ \beta_k \in B_k, \beta \in B$. On a, par définition, $\alpha \prec \beta_1$, $\beta_1 \prec \beta_2$, ..., et $\beta_k \prec \beta$. Or $A \supset X$ et $B \supset Y$ donc $\forall i \in [1..k],\ \alpha \supset \beta_i \vee \beta \supset \beta_i$. On est donc dans les hypothèses de A9, donc on déduit $\alpha \prec \beta$. Finalement, F9 est vérifiée.

**F10** : A-t-on $(B_1 \prec B_2 \wedge ... \wedge B_k \prec B_1) \wedge A \supset X \wedge B \supset Y \vdash \overline{AB}$ où $X \cup Y = \{B_1, B_2, ...B_k\} \subset E(V)$ ? Soit $\beta_1 \in B_1,\ \beta_2 \in B_2, \ldots,\ \beta_k \in B_k$. On a, par définition, $\beta_1 \prec \beta_2$, ..., et $\beta_k \prec \beta_1$. Soit $\alpha \in A$ et $\beta \in B$ quelconques. On a : $A \supset X$ et $B \supset Y$ donc $\forall i \in [1..k],\ \alpha \supset \beta_i \vee \beta \supset \beta_i$. On se retrouve donc dans les hypothèses de A10. Donc on a $\overline{\alpha\beta}$. Finalement, F10 est vérifiée.

**F11** : A-t-on $\overline{AB} \vdash A \prec B$ ? Soit $\alpha \in A$ et $\beta \in B$. On a $\overline{\alpha\beta}$. Donc on a $\alpha \prec \beta$ en utilisant A11. F11 est vérifiée.

On arrive donc à prouver **F9** à **F11**. Néanmoins **F9** pose problème. En effet, on a **AF4**, c'est à dire qu'on n'a pas toujours $A \supset A$, donc si $A \prec A$, on ne peut pas prouver $\overline{A}$. On propose donc de rajouter l'axiome suivant :

**F12** : $(A \prec B \wedge B \prec A) \vdash \overline{AB}$

Ainsi avec **F9** et **F10**, on arrive à prouver $\overline{AB}$ à partir des hypothèses de **A10** même si $A$ et $B$ sont dans $X$ et $Y$, et si on n'a pas $A \supset A$ et $B \supset B$.

L'ensemble des formules que nous venons de démontrer nous permet d'accélérer le calcul des méta-règles que l'on peut inférer. On propose l'algorithme suivant afin d'accélérer la méthode de Jagadish en utilisant l'écriture ensembliste et les formules **F1** à **F12** :

1. on applique **F1** à **F12** aux méta-règles ensemblistes de manière répétée jusqu'à arriver dans un état stable ;
2. on déduit de l'ensemble des méta-règles ensemblistes inférées à l'étape 1 l'ensemble des méta-règles individuelles correspondant ;
3. on applique **A1** à **A11** aux méta-règles individuelles inférées à l'étape 2 de manière répétée jusqu'à arriver dans un état stable.

De cette façon, on aboutit au même résultat qu'en passant directement par les méta-règles individuelles. Le calcul est plus rapide car les méta-règles ensemblistes sont beaucoup moins nombreuses que les méta-règles individuelles correspondantes, et les inférences de la première étape sont en temps polynomial en fonction du nombre de méta-règles ensembliste. La seconde étape est de toute façon nécessaire si on transcrit en premier lieu les méta-règles ensemblistes sous forme de méta-règles individuelles. Enfin, la dernière étape est fortement accélérée, puisqu'une grande partie des méta-règles inférables aura déjà été inférée à l'étape 1.

**Vérifications sur le système**

Nous avons défini un système formel, inspiré du système de Jagadish, mais répondant au problème que nous nous sommes posés. Nous allons désormais montrer que l'on peut faire un certain nombre de vérifications intéressantes sur ce système.

Comme nous venons de le voir, une fois que les inférences ont été effectuées jusqu'à obtenir l'état stable maximal des méta-règles, on utilise la définition de nos méta-règles pour en déduire des méta-règles portant uniquement sur sur des règles individuelles, comme dans le système de Jagadish. On finit par des inférences sur ces méta-règles individuelles. C'est à partir de ces méta-règles individuelles que l'on va effectuer un certain nombre de vérifications.

*Règles mortes*

Si une règle $r$ appartient à un ensemble $R$ tel que l'on a $\overline{R}$, alors on a $\overline{r}$. De même, si une règle $r$ appartient à l'intersection d'un ensemble $A$ et d'un ensemble $B$ tels que l'on a $\overline{AB}$, alors on a $\overline{r}$. On détermine également l'ensemble des couples $(a, b) \in V^2$ tels que $\overline{ab}$. On obtient ainsi très simplement l'ensemble des couples de règles mutuellement exclusives, et surtout l'ensemble des règles mortes. On peut donc déclarer automatiquement à un créateur d'hypermédia adaptatif quelles règles sont inutiles avant même d'avoir des données à traiter. Cela peut lui permettre soit de supprimer ces règles et les méta-règles les utilisant explicitement, soit de corriger une erreur potentielle dans le système. On suppose en effet qu'une règle fait partie du système si elle doit être utilisée au moins dans certains cas.

*Déterminisme*

On repart de méta-règles "individuelles", donc l'algorithme donné dans [50] s'applique sans problème :

- On construit le graphe de précédence pour la relation de nécessité et on en retire les éléments fortement connexes. Ceci est rendu possible par le traitement "à part" de F4 (formule fausse), qui consiste à rajouter, pour tout $r \in V$ la méta règle $r \supset r$. On réduit le système.
- Dans le système réduit, on retire ensuite toutes les méta-règles d'exclusion mutuelle qui peuvent être déduites de toutes les autres méta-règles par application de A1 à A11.
- Pour chaque méta-règle d'exclusion restante, de la forme $\overline{ab}$, on vérifie qu'on a bien soit $a > b$, soit $b > a$ (mais pas les deux en même temps).

On peut donc vérifier le déterminisme en temps polynomial en fonction du nombre de méta-règles individuelles déduites de l'ensemble initial de nos méta-règles.

*Ordonnancement*

On considère le système réduit. Il suffit de regarder si pour tout couple $(a, b) \in V^2$, on a bien soit $a \prec b$, soit $b \prec a$ dans l'ensemble des méta-règles individuelles. Une fois le calcul de l'ensemble

maximal des méta-règles inférables effectué, ce calcul s'effectue en temps polynomial en fonction du nombre de règles.

*Catégorisations d'utilisateurs*

On peut déterminer, grâce aux inférences sur les méta-règles, l'ensemble de sortie $O$ pour un ensemble de règles en entrée $I$. Notre sélection de règle portant sur une stratégie de parcours relative à un utilisateur, on peut déterminer, pour chaque combinaison de paramètres concernant l'utilisateur, l'ensemble des règles à exécuter effectivement. Dans le cas d'un système où le nombre de paramètres utilisés pour décrire l'utilisateur dans $F$ n'est pas trop important, on peut envisager de pré-calculer toutes les combinaisons possibles. Ainsi, on pourra non seulement gagner du temps de calcul à la volée, mais aussi déterminer si certaines combinaisons de paramètres aboutissent à $O = \emptyset$.

On peut alors soit corriger une possible erreur dans le système, soit déclarer qu'effectivement, cette combinaison de paramètre n'est pas souhaitable pour le système, et ainsi l'interdire en amont, au niveau du modèle de l'utilisateur.

### 5.4.5 Sémantique de la relation d'ordre

Nous avons construit un système de méta-règles pour les hypermédias adaptatifs, et nous pouvons vérifier un certain nombre de caractéristiques de ce système. La sémantique des relations d'exigence positive, d'exclusivité mutuelle et de préférence est claire et similaire à celle du système de Jagadish. Néanmoins, la notion d'ordre d'exécution des règles, présente dans le système de Jagadish, n'a pas de sens pour un système d'hypermédia adaptatif. En effet, on applique l'ensemble des règles dans n'importe quel ordre pour chaque action potentielle, et les actions qui sont apparues comme étant bonnes pour une règle donnée sont considérées, dans la situation courante, comme des actions bonnes à effectuer. Néanmoins, nous avons envisagé plusieurs possibilités pour utiliser la relation d'ordonnancement de notre système sur nos règles de base. Plutôt que de considérer l'ordre d'exécution des règles, nous proposons que l'ordre décrive une priorité entre les règles.

Nous envisageons donc deux utilisations possibles de cette relation d'ordonnancement :

- On peut définir le sens de la relation d'ordre de la façon suivante : Si $a \prec b$, alors les actions rendues "bonnes" par la règle $a$ sont meilleures (au sens du better déjà défini dans dans la logique situationnelle) que celles rendues "bonnes" par $b$. Cette utilisation est appelée "prioritarisation" des actions.
- Afin de rendre la conception de l'adaptation plus fine, en permettant de donner une importance prioritaire à certaines règles, nous pouvons également utiliser la relation d'ordre des méta-règles de la façon suivante. Si plusieurs règles confèrent plusieurs degrés de désirabilité différents à une même action dans une situation donnée, alors le degré de désirabilité retenu est celui de la règle la plus en avant pour la relation d'ordre et non plus le plus haut degré atteint par au moins une règle. Cette utilisation est appelée "prioritarisation" des degrés.

Si l'ordonnancement est correct, alors il n'y a pas de risque d'ambiguïté, puisque toutes les règles sont classées deux à deux pour la relation d'ordre. Sinon, c'est à dire s'il peut arriver que plusieurs règles soient les plus en avant pour la relation d'ordre, alors on ne retient que le plus haut degré de désirabilité fourni par ces dernières règles.

Dans les exemples de ce document, nous avons choisi d'appliquer la prioritarisation des actions.

### 5.4.6 Système de preuve

Nous avons jusqu'à présent défini un système de règles muni de méta-règles qui permettent de sélectionner un sous-ensemble de l'ensemble de toutes les règles. Nous avons donc une syntaxe pour définir ces éléments, mais pas encore de façon de prouver si l'on peut montrer qu'une règle confère un degré de désirabilité donné à une action donnée. Les règles de base prennent en compte les données relatives au domaine, les méta-règles prennent en compte les données relatives à l'utilisateur.

Dans cette section, nous allons montrer comment on peut construire des systèmes de preuve sur nos règles et méta-règles, prenant en compte les connaissances de l'utilisateur.

L'intérêt de ces systèmes de preuve est double : ils permettent de décrire de manière formelle et universellement compréhensible la façon dont doivent être interprétées les règles, la façon de raisonner sur ces règles ; et ils permettent de décrire des façons de prendre en compte les connaissances de l'utilisateur, dans une stratégie de raisonnement donnée. On peut alors imaginer laisser le choix à un concepteur d'hypermédia adaptatif entre plusieurs stratégies de raisonnement différentes, prenant en compte les connaissances de l'utilisateur de différentes façons.

Dans cette section nous proposons une base commune pour tous les systèmes de preuve que l'on peut imaginer. Les éléments de cette base ne dépendent pas des connaissances de l'utilisateur. Les éléments de déduction relatifs aux connaissances de l'utilisateur sont détaillés en dehors de la base. Nous donnons deux possibilités aisément réutilisables pour prendre en compte les connaissances de l'utilisateur. La première possibilité consiste à n'accepter de proposer un document que si ses prérequis sont tous connus. Il s'agit d'une base de raisonnement implicite de nombreux systèmes. La seconde possibilité prend en compte les spécificités de notre modèle de domaine en deux couches : la couche conceptuelle, et la couche des ressources.
Nous rappelons que la forme des règles est la suivante :
$R : \forall X \in E(\bigwedge_{i \in I} P_i(X) \Rightarrow Deg(Action(X), d)$

D'un point de vue strictement formel, une fois les implications des méta-règles calculées, les méta règles peuvent s'exprimer de la façon suivante :
$\&_{i \in I} C_i \rightsquigarrow R$
C'est-à-dire qu'un ensemble de critères réalisés entraîne l'utilisation d'une règle. Dans la syntaxe ci-dessus, $I$ est un ensemble fini d'indices. Les éléments de la forme $C_i$ sont des critères de déclenchement. $R$ est une règle. Si tous les critères $C_i$ sont vrais pour l'utilisateur courant, alors la règle $R$ est utilisée.

Nous utilisons la syntaxe décrite ci-dessus pour définir, en déduction naturelle, le base de système de preuve formel suivante :

$$\frac{doc \in E \quad R \quad R{:}\forall X \in E(P(X) \Rightarrow Deg(Action(X),d)) \quad Action(doc)|valide \quad P(doc)}{Action(doc)}$$

Cette première règle de déduction indique que, pour démontrer qu'une action a un certain de degré de désirabilité, il faut prouver les éléments suivants :

- L'objet sur lequel porte l'action appartient au domaine ;
- Une règle est sélectionnée par les méta-règles ;
- La règle sélectionnée conclut à une action de la forme de l'action recherchée ;
- L'action est valide, c'est-à-dire faisable en l'état des connaissances de l'utilisateur ;
- Les prémisses de la règle sélectionnée sont vrais quand sa conclusion porte sur $doc$.

$\frac{A \quad B}{A \wedge B}$

Pour montrer $A \wedge B$, il faut montrer $A$ et il faut montrer $B$

$\frac{X \in DP(X)}{\forall X \in DP(X)}$

Pour montrer $\forall X \in DP(X)$, il faut montrer $X \in DP(X)$ pour $X$ libre ($X$ quelconque).

$\frac{a \in DP(a)}{\exists X \in DP(X)}$

Pour montrer $\exists X \in DP(X)$, il faut montrer qu'il y a un $a$ donné tel que $a \in D$ et tel que $P(a)$

$\frac{A \Re B}{A \in \{X / X \Re B\}}$

Pour montrer que $A$ fait partie de l'ensemble des éléments en relation avec $B$, il faut montrer que $A$ est en relation avec $B$

$\frac{type(A,t)}{A \in \{X / type(X,t)\}}$

$A$ appartient à l'ensemble des constituants de type $t$ si $A$ est de type $t$.

$\frac{}{X \in Cst}$

Un constituant appartient toujours à l'ensemble des constituants.

$\frac{A \Re B}{A \Re^* B}$, $\frac{A \Re C \quad C \Re^* B}{A \Re^* B}$, $\frac{A \Re C \quad C \Re^{n-1} B}{A \Re^n B}$, $\frac{A \Re B}{A \Re^1 B}$

$A$ est en relation transitive avec $B$ si $A$ est en relation avec $B$ ou si $A$ est en relation avec un élément qui est lui-même en relation transitive avec $B$. $A$ est en relation transitive d'ordre $n$ avec $B$ si $A$ est en relation avec un élément qui est lui-même en relation transitive d'ordre $n-1$ avec $B$.

$\frac{C \rightsquigarrow R \quad C}{R}$

Une règle est utilisable si un ensemble de critère permet de l'utiliser, et si cet ensemble de critère est vrai pour l'utilisateur courant.

$\frac{C_1 \quad C_2}{C_1 \& C_2}$

La conjonction de deux critères ou de deux conjonctions de critères est vraie si chaque (conjonction de) critère est vrai pour l'utilisateur courant.

Nous venons de détailler la base de notre système de déduction. Celle-ci explicite formellement que les prémisses d'une règles doivent être vérifiées pour aboutir à la conclusion souhaitée, que la

règle doit avoir été sélectionnée par les méta-règles correspondant aux critères caractérisant l'utilisateur. Elles explicitent également comment démontrer qu'on a bien une conjonction de prémisses, comment on traite la notion de clôture transitive etc.

La seule notion qui n'est pas démontrable avec cette seule base est la notion de validité d'un document, c'est-à-dire le fait que les connaissances d'un utilisateur lui permettent d'aborder ce document. En effet, il existe plusieurs possibilités pour prendre en compte ces connaissances. Par exemple, il est possible de prendre en compte un niveau moyen sur les concepts pré-requis, ou bien de considérer que chaque pré-requis doit être connu avec un niveau minimum. Nous allons détailler ici deux complément possible à notre base de déduction.

Le premier complément, qui sert de base à de nombreux systèmes simples, consiste à dire qu'il est possible de faire une action vers un document si les pré-requis de celui-ci sont connus. Ainsi, on ajoute à la base de déduction la règle suivante :

$$\frac{PR::pre-requis::doc \quad PR|connu \quad X|inconnu}{Action(doc)|valide}$$

Ici, l'opérateur :: permet de regrouper dans $PR$ l'ensemble des éléments en relation avec $doc$ pour la relation $pre-requis$. Il s'agit simplement d'une notation raccourcie pour décrire une interrogation du domaine. $PR$ est de la forme $A\&B\&C\ldots$ où $A$, $B$, $C$ etc. sont des constituants du domaine. Pour que $action(doc)$ soit valide, il faut que tous les pré-requis de $doc$ soient connus, et que $doc$ ne le soit pas.

On complète cette règle par des règles qui permettent de prendre en compte l'opérateur :: et la notion de connu et d'inconnu pour un ensemble.

$$\frac{A|V \quad B|V}{A\&B|V}$$

La conjonction de $A$ et $B$ est validée pour le validateur $V$ (i.e. $connu$ ou $inconnu$) si $A$ et $B$ sont validés individuellement pour le validateur $V$.

Nous allons maintenant décrire un complément à notre base de déduction qui prend en compte la séparation concept/ressources. Ce complément prend également en compte le fait que certaines actions aboutissent à la lecture de documents, alors que d'autres aboutissent à l'exécution d'exercices. Ainsi, ce complément prend en compte deux types de relation. Nous obtenons les règles de déduction suivantes :

$$\frac{type(Action,lecture) \quad abstract(doc,concept) \quad PR::pre-requis::concept \quad PR|conn.\geq moyenne \quad concept|conn.<moyenne}{Action(doc)|valide}$$

Cette règle décrit qu'il est valide de faire une action de lecture, si le concept abordé par le document est peu connu et si les concepts pré-requis sont au moins moyennement connus.

$$\frac{type(Action,exercice) \quad abstract(doc,concept) \quad concept|conn.=moyenne}{Action(doc)|valide}$$

Cette règle décrit qu'il est valide de faire une action d'exercice si le concept abordé par le document visé est connu avec le niveau moyen, c'est-à-dire qu'il a été abordé mais que l'utilisateur peut avoir besoin d'entraînement.

La règles qui permettent de faire des déductions sur la conjonction des pré-requis est la même que pour le premier complément, à savoir :

$\frac{A|V \quad B|V}{A\&B|V}$

Nous voyons donc qu'il est possible de définir plusieurs modèles de raisonnement sur nos règles et nos méta-règles, prenant en compte de différentes façons les connaissances de l'utilisateur. Un système pédagogique fondé sur nos modèles pour le e-learning pourrait ainsi proposer quelques choix de modèles de raisonnement, explicités en langage naturel, pour modifier les modalités d'application des règles, et l'adaptation fournie en fonction des connaissances requises pour aborder les différents documents du domaine.

### 5.4.7 Exemple d'utilisation

Maintenant que nous avons défini clairement notre système et la sémantique que nous souhaitons lui associer, nous allons donner un exemple concret et simple d'utilisation de ce système, basé sur notre première proposition de complément de déduction (sans gestion des concepts et des degrés de connaissance). Cet exemple n'a pas vocation à représenter entièrement un cas réel. Il sert d'illustration au système de méta-règle que nous venons de construire. Un exemple plus complet, portant sur un cas d'utilisation réel, et utilisant notre deuxième complément de déduction, sera donné dans le chapitre 6.

Dans cet exemple, nous allons définir un système de règles, i.e. les ensembles $V$, $F$, $M$ et la fonction $\delta$. Nous montrerons ensuite le fonctionnement de ce système à travers quelques exemples d'utilisation concrets. Nous utilisons uniquement les degrés de désirabilité "possible" et "désirable" dans cet exemple.

*Règles de base*

Voici les règles que nous proposons d'utiliser :

**R1 :** $\forall D \in Cst,\ preRequis(documentCourant, D) \wedge preRequis^*(D, but) \Rightarrow good(lire(D))$

Cette règle décrit le fait qu'il est bon de lire un document s'il mène au but et s'il est un successeur direct du document courant. Cette règle correspond à un parcours en profondeur d'abord du graphe des documents.

**R2 :** $\forall D \in Cst,\ \forall Abr \in Cst,\ versionAbrege(Abr, D) \wedge preRequis(documentCourant, D) \wedge preRequis^*(D, but) \Rightarrow good(lire(D))$

Cette règle décrit le fait qu'il est bon de lire la version abrégée d'un document si elle existe, si elle mène au but et si elle est un successeur direct du document courant. Cette règle correspond à un parcours en profondeur d'abord du graphe des documents, avec affichage de la plus petite version possible du document. Elle est utile pour un utilisateur rapide.

**R2b :** $\forall D \in Cst, \forall Abr \in Cst,\ versionAbrege(Abr, D) \wedge preRequis^*(D, but) \Rightarrow good(lire(D))$

Cette règle décrit le fait qu'il est bon de lire la version abrégée d'un document si elle existe et si elle mène au but. Cette règle correspond à un affichage de la plus petite version possible du document. Elle est utile pour un utilisateur rapide. Elle permet de complèter la règle **R2**

en parcourant les éléments qui ne sont pas successeurs directs du document courant. Il faudra donc décrire, dans les méta-règles, que la règle R2 est prioritaire par rapport à la règle R2b

**R3 :** $\forall D \in Cst,\ \neg preRequis(documentCourant, D) \wedge preRequis^*(D, but) \Rightarrow good(lire(D))$

Cette règle décrit le fait qu'il est bon de lire un document s'il mène au but et s'il n'est pas un successeur direct du document courant. Cette règle correspond à un parcours en largeur d'abord du graphe des documents.

**R4 :** $\forall Ex \in \{Ex/type(Ex, Exercice)\}, estExercicePour(documentCourant, Ex) \wedge$
$preRequis^*(documentCourant, but) \Rightarrow good(faire(Ex))$

Cette règle décrit le fait qu'il est bon de s'entraîner sur un document s'il est un exercice , s'il est directement relié au document courant, et si ce document courant mène à l'objectif. Cette règle permet de parcourir des exercices liés aux thèmes à traiter, sans que ces exercices ne soient nécessairement des pré-requis pour l'objectif.

**R5 :** $\forall Ex \in \{Ex/type(Ex, Exercice)\}, \forall D \in Cst, estExercicePour(D, Ex) \wedge$
$preRequis^*(D, but) \Rightarrow good(faire(D))$

Cette règle décrit le fait qu'il est bon de s'entraîner sur un document s'il est un exercice et s'il est directement relié à un document qui mène au but. Cette règle permet de parcourir des exercices liés aux thèmes à traiter, sans que ces exercices ne soient nécessairement des pré-requis pour l'objectif, mais de façon potentiellement plus dispersée que la règle précédente (on n'impose pas que l'exercice soit un exercice pour le document courant).

**R6 :** $\forall D \in D/type(D, illustration),\ illustrationPour(documentCourant, D) \wedge$
$preRequis^*(D, but) \Rightarrow good(lire(D))$

Cette règle décrit le fait qu'il est bon de lire un document s'il est une illustration et s'il est directement relié à un document qui mène à l'objectif. Cette règle permet de parcourir des illustrations liés aux thèmes à traiter, sans que celles-ci ne soient nécessairement des pré-requis pour l'objectif.

**R7 :** $\forall D \in Cst,\ preRequis^*(D, but) \Rightarrow good(lire(D))$

Cette règle décrit le fait qu'il est possible de lire un document s'il mène à l'objectif. Cette règle est notamment utile en complément d'une règle de parcours en profondeur (resp. en largeur) du graphe, puisqu'elle permet de traiter le cas où on ne peut pas continuer plus en profondeur (resp. en largeur).

Cet ensemble de règle fait clairement apparaître plusieurs stratégies de parcours. Certaines sont complémentaires, d'autres sont opposées. Les règles font intervenir différentes relation entre les documents (est un pré-requis de, est un exercice pour, est une illustration de...).

*Critères de déclenchement*

On définit les critères de déclenchement suivants :

**C1** type_apprentissage = linéaire.

**C2** type_apprentissage = dispersé.

**C3** vitesse_apprentissage = lent.

**C4** vitesse_apprentissage = rapide.

**C5** besoin_exercice = oui.

**C6** besoin_exercice = non.

On utilise donc 3 caractéristiques pouvant prendre chacune 2 valeurs distinctes et opposées, soit 6 critères.

*Association règles-critères de déclenchement*

On va répartir les règles de la façon suivante :

- Les règles **R1** et **R7** permettent un parcours en profondeur, elles sont donc recommandées pour un apprentissage linéaire.
- Les règles **R3** et **R7** permettent un parcours en largeur, elles sont donc recommandées pour un apprentissage dispersé.
- Les illustrations sont recommandés pour l'apprentissage lent, on associe donc ce critère à la règle **R6**.
- **R4** correspond à un apprentissage linéaire pour quelqu'un qui a besoin d'exercices.
- **R5** correspond à un utilisateur qui a besoin d'exercice.
- **R7** est toujours utilisable, bien que moins sélective. Elle ne permet, de toutes façons, la sélection de documents que dans le cas où aucun document n'est "bon" à consulter.

À partir de ces considérations informelles, nous proposons les associations règles-critères suivantes :

- $\delta(R1) = \{C1\}$
- $\delta(R2) = \{C4\}$
- $\delta(R2b) = \{C4\}$
- $\delta(R3) = \{C2\}$
- $\delta(R4) = \{C1,\ C5\}$
- $\delta(R5) = \{C5\}$
- $\delta(R6) = \{C3\}$
- $\delta(R7) = \emptyset$

On note que ces associations pourraient très facilement être indiquées de manière graphique dans un système, de sorte à faciliter la tâche d'un créateur d'hypermédias adaptatifs.

*Méta-règles*

Dans cette partie, on utilise pour la relation d'ordonnancement le sens d'un *better* au niveau des règles. On choisit les méta-règles suivantes :

**M1** $\gamma(C3) \supset \gamma(C5)$

Les règles relatives à un apprentissage lent nécessitent l'utilisation d'exercices.

**M2** $\gamma(C1) \supset \{R7\}$

Les règles pour l'apprentissage linéaire nécessitent la règle générale **R7**.

**M3** $\gamma(C2) \supset \{R7\}$

Les règles pour l'apprentissage dispersé nécessitent la règle générale **R7**.

**M4** $\gamma(C4) \supset \{R1\}$

Les règles pour l'apprentissage rapide nécessitent les règles pour l'apprentissage linéaire.

**M5** $\overline{\gamma(C1)\gamma(C2)}$

Les règles pour l'apprentissage linéaire sont exclusives des règles pour l'apprentissage dispersé. Si un créateur d'hypermédia adaptatif associait ces deux critères à une même règle, elle serait une règle morte.

**M6** $\overline{\gamma(C3)\gamma(C4)}$

Les règles pour l'apprentissage lent sont exclusives des règles pour l'apprentissage rapide.

**M7** $\overline{\gamma(C5)\gamma(C6)}$

Les règles pour l'apprentissage avec exercices sont exclusives des règles pour l'apprentissage sans exercice. Ici, aucune règle n'est associée à **C6**, mais il est préférable de rentrer d'ores et déjà cette méta-règle pour faciliter la maintenance du système.

**M8** $\overline{\gamma(C4)\gamma(C5)}$

Les règles pour l'apprentissage avec exercices sont exclusives des règles pour l'apprentissage rapide.

**M9** $\gamma(C3) > \gamma(C1)$

Une règle pour l'apprentissage lent doit être préférée à une règle pour l'apprentissage rapide si les deux règles ne peuvent pas s'exécuter en même temps.

**M10** $\gamma(C1) > \gamma(C2)$

Une règle pour l'apprentissage linéaire doit être préférée à une règle pour l'apprentissage dispersé si deux telles règles sont déclenchables (par exemple si le type d'apprentissage ne peut être déterminé).

**M11** $\gamma(C3) > \gamma(C4)$

Une règle pour l'apprentissage lent doit être préférée à une règle pour l'apprentissage rapide si deux telles règles sont déclenchables (par exemple si la vitesse d'apprentissage ne peut être déterminé).

**M12** $\gamma(C5) > \gamma(C6)$

Une règle pour l'apprentissage avec exercice doit être préférée à une règle pour l'apprentissage sans exercice si deux telles règles sont déclenchables.

**M13** $\{R4, R5\} \prec \{R1, R2\}$

Les règles donnant accès aux exercices sont prioritaires par rapport aux règles donnant un parcours linéaire (dans le cas contraire, on ne s'arrêterait jamais sur les exercices). Ici, on prend garde à traiter à part l'intersection de $\gamma(C5)$ et $\gamma(C1)$, à savoir **R4**, dont on détaille le classement dans **M20**.

**M14** $\gamma(C5) \prec \gamma(C2)$

Les règles donnant accès aux exercices sont prioritaires par rapport aux règles donnant un parcours dispersé (dans le cas contraire, on ne s'arrêterait jamais sur les exercices).

**M15** $\gamma(C3) \prec \gamma(C5)$

Les règles privilégiant un parcours lent (passage par les illustrations) sont prioritaires par rapport aux règles donnant accès aux exercices. Cela signifie qu'on va voir les illustrations avant les exercices.

**M16** $\gamma(C3) \prec \gamma(C1)$

Les règles privilégiant un parcours lent sont prioritaires par rapport aux règles donnant un parcours linéaire (dans le cas contraire, on ne s'arrêterait jamais sur les illustrations).

**M17** $\gamma(C3) \prec \gamma(C2)$

Les règles privilégiant un parcours lent sont prioritaires par rapport aux règles donnant un parcours dispersé (dans le cas contraire, on ne s'arrêterait jamais sur les illustrations).

**M18** $\{R2\} \prec \{R1\}$

**M19** $\{R1\} \prec \{R2b\}$

Les deux méta-règles précédentes permettent d'ordonner en détail les règles **R1**, **R2** et **R2b**, pour privilégier un parcours en profondeur d'abord, puis, chaque fois que cela est possible, une version abrégée.

**M20** $\{R4\} \prec \{R5\}$

Il est préférable de faire un exercice lié au document courant plutôt que de faire n'importe quel exercice.

Cet exemple permet de voir comment on peut utiliser les 4 types de méta-règles. Il met également en avant le fait qu'on peut manipuler des règles individuelles, des ensembles de règles correspondant à des critères, mais aussi un "mélange" de ces deux types d'ensembles.

On voit que, quand une relation d'exclusion mutuelle est donné, on décrit une relation de préférence sur les mêmes ensembles. On voit également que quand un type de règle en nécessite un autre, il est nécessaire d'ordonner les règles de ces deux ensembles.

*Utilisation du système*

Maintenant que nous avons défini l'ensemble des quatre éléments caractéristiques de notre système de règle, nous allons étudier plusieurs types d'utilisateur possibles et regarder comment le système de règles se comporte.

Supposons que l'utilisateur est lent, préfère un apprentissage linéaire, et qu'il n'a pas besoin d'exercice. Avec **M1**, on infère que les règles pour les utilisateurs lent sont désactivées, puisque les règles pour les utilisateurs ayant besoin d'exercice ne sont pas déclenchables. Parmi les règles déclenchables, il reste donc les règles relatives à l'apprentissage linéraire, à savoir **R1**, et la règle **R7** qui est toujours déclenchable.

Supposons maintenant que l'utilisateur est lent, préfère un apprentissage dispersé, et qu'il a besoin d'exercices. Dans ce cas, **M1** ne pose pas de problème. La règle **M3** non plus, puisque **R7** est déclenchable et n'est exclusive d'aucune règle. Avec **M14**, on sait que les règle **R4** et **R5** qui concernent les exercices sont prioritaires sur la règle **R3**. On sait aussi avec **M15** que la règle **R6** concernant les illustrations est prioritaire par rapport aux règles portant sur les exercices. **M17** n'apporte rien puisqu'elle est simplement, dans ce cas, l'expresssion de la transitivité de la relation d'ordonnancement (avec **M14** et **M15**). Enfin, avec **M20**, on sait que **R4** est prioritaire sur **R5**.

Finalement, on sélectionne les règles **R3**, **R4**, **R5**, **R6** et **R7**. L'ordre retenu est : $R6 \prec R4 \prec R5 \prec R3$. On remarque que **R7** n'est pas ordonné. En fait, son ordonnancement n'est pas nécessaire puisque c'est une règle de possibilité, qui dans le système est déjà moins prioritaire que les règles de désirabilité. On pourrait inférer automatiquement cette propriété au niveau des méta-règles.

Supposons que l'utilisateur est rapide, privilégie l'apprentissage dispersé, et n'a pas besoin d'exercices. Comme il ne vérifie pas **C1** (apprentissage linéaire), les règles pour l'apprentissage rapide sont desactivées par **M4**. Les règles retenues seront donc les règles **R3** et **R7**. L'utilisateur fera donc un parcours en largeur d'abord, sans visualiser les illustrations et sans faire les exercices en annexe.

Supposons enfin que l'utilisateur est rapide, qu'il privilégie l'apprentissage linéaire et qu'il a pas besoin d'exercice. La méta-règle **M8** stipule que les règles pour l'apprentissage rapide sont exclusives des règles pour l'apprentissage avec exercice. Par application des axiomes d'inférence, on ne peut ni déduire $\gamma(C5) > \gamma(C4)$, ni $\gamma(C4) > \gamma(C5)$. Notre exemple n'est donc pas déterministe. On voit ici un intérêt majeur des vérifications que l'on peut effectuer, a priori, sur les méta-règles. Rajoutons la méta-règle **M21** : $\gamma(C5) > \gamma(C4)$. On privilégie donc les exercices par rapport à l'apprentissage rapide.

Les règles sélectionnées sont **R1**, **R4**, **R5**, **R7**. On obtient l'ordre suivant en utilisant **M13** et **M20** :

$R4 \prec R5 \prec R1$ et $R7$ est de priorité inférieure puisqu'il s'agit d'une règle de possibilité.

Si l'utilisateur n'avait pas eu besoin d'exercices, on aurait obtenu, avec **M18** et **M19** :

$R2 \prec R1 \prec R2b$ et $R7$ est de priorité inférieure puisqu'il s'agit d'une règle de possibilité.

### 5.4.8 Conclusions sur les méta-règles

Les systèmes actuels d'hypermédias adaptatifs utilisent rarement des techniques de la méta-adaptation, des méta-règles. Quand ils le font, la manière de le faire est souvent laissée aux soins de ceux qui devront implémenter le système : la méta-adaptation est définie comme étant un composant du système, la façon dont se fait cette méta-adaptation n'est pas définie, et est souvent ad hoc. Quand ils ne le font pas, fournir, en plus des parcours adaptés à l'utilisateur, des **stratégies** de parcours de documents différentes en fonction de l'utilisateur ne peut se faire dans des règles classiques qu'en ajoutant de nombreuses prémisses. Dans ce cas, il devient rapidement difficile de prévoir quelle stratégie de parcours correspondra à quel type d'utilisateur. La maintenance de ces systèmes est également très difficile à assurer.

Dans cette partie, nous avons présenté un système de règles, utilisant des méta-règles, et spécifiquement conçu pour les hypermédias adaptatifs. Ce système permet d'adapter la stratégie elle même (en plus des parcours calculés) en fonction de l'utilisateur, sans rendre les règles de base difficiles à concevoir et à maintenir. De plus, nous avons montré que ce système permet de faire un certain nombre de vérifications souhaitables, de sorte à faciliter la tâche des créateurs d'hypermédias adaptatifs. Enfin, nous avons donné un système de déduction formel qui rend ce système universellement compréhensible.

Il est important de noter que le système de règles mis en place dans ce document permet de générer de véritables stratégies de parcours à part entière, différentes en fonction du profil, des capacités, des préférences de l'utilisateur.

Notre système présente l'avantage de pouvoir combiner, de façon très fine, et déclarative, des règles correspondant à différents critères. On aurait pu se contenter d'une simple catégorisation associant un type d'utilisateur à un type de règle, mais le résultat aurait été beaucoup moins précis que dans le système que nous avons obtenu. De plus, notre système permet d'éviter certains problèmes de dimensionnement que l'on pourrait rencontrer en utilisant d'autres techniques. Par exemple, si on veut prendre en compte 5 critères ternaires (vrais, faux ou indéterminés) concernant l'utilisateur, il faudrait définir manuellement jusqu'à 243 ensembles de critères et de règles associés, là où quelques dizaines de méta-règles doivent suffire à décrire l'essentiel des différentes stratégies d'adaptation, comme nous le verrons dans le chapitre 6.

Ainsi, nous avons créé un système de règles, basé sur le calcul des prédicats du premier ordre, qui permet de décrire l'adaptation en séparant les axes orthogonaux. Cette approche offre deux avantages. Tout d'abord elle est fondée sur un formalisme connu, ce qui la rend plus compréhensible, plus facile à transcrire et à implémenter, d'autant plus que nous avons donné un système formel de déduction qui définit précisément ce qui peut ou non être démontré. D'autre part, elle permet de penser la conception de l'adaptation en terme de déclarations courtes, ayant un sens vis-à-vis du problème posé. Les règles ont peu de prémisses, et ne contiennent que des relations sur le domaine : elles restent suffisamment simples pour être écrites rapidement et avec un risque d'erreur minimisé.

## 5.5 Recherche de tronçons de parcours

Dans cette section, nous allons aborder un aspect différent de nos travaux. Dans ce chapitre, jusqu'à présent, nous avons décrit les fondements d'un modèle d'adaptation, où la logique situationnelle sert de base pour gérer les changements de situation et les actions, et où notre système de règles permet de décrire l'adaptation que l'on veut mettre en place.

Ici, nous allons aborder un problème différent concernant l'adaptation, qui peut s'interfacer avec notre système comme avec d'autres systèmes. Nous avons étudié la possibilité de fournir à l'utilisateur des tronçons de parcours, à travers lesquels il est entièrement guidé de document en document, et qui le mènent à l'objectif final en ne lui permettant de finir sa session qu'après avoir réalisé des objectifs intermédiaires, qui terminent chaque tronçon.

Nous avons mis au point une méthode générique de pré-traitement du domaine qui permet de déterminer, à l'aide de règles, les meilleurs objectifs intermédiaires potentiels. Une fois ces objectifs déterminés, nous construisons un graphe de graphe, appelé méta-graphe, qui permet de séparer chaque tronçon, et de prévoir l'ordre des tronçons à parcourir.

Le but de l'utilisation de tronçons de parcours est d'offrir à l'utilisateur une approche plus guidée qu'habituellement dans les hypermédias adaptatifs. Cette approche permet aussi d'empêcher l'utilisateur d'arrêter sa session sur des documents où il est peu souhaitable qu'il s'arrête. De plus, la façon dont nous construisons nos tronçons permet d'éviter que les documents proposés à l'utilisateur soient présentés dans un ordre trop aléatoire : les regroupements de documents pour chaque tronçon peuvent aborder un sujet commun, si les règles de construction des objectifs intermédiaires vont dans ce sens.

Dans cette section, nous nous baserons sur un domaine simple, constitué essentiellement de documents liés entre eux par des relations et annotés. En effet, les idées développées ici doivent pouvoir s'appliquer dans des domaines divers, et nous ne souhaitons pas prendre en compte particulièrement les différents niveaux d'abstraction potentiels.

### 5.5.1 Quelques définitions

Avant de décrire notre méthode de parcours par tronçon, nous rappelons quelques définitions liées aux graphes, et nous présentons la définition de quelques éléments nouveaux que nous introduisons ici. Les définitions qui sont reprises sont tirées de [26].

Un graphe orienté $G = [X, U]$ est déterminé par la donnée :

- d'un ensemble $X$ dont les éléments sont appelés des *sommets* ou des *noeuds*. Si $N =\mid X \mid$ est le nombre de sommets (de noeuds), on dit que le graphe $G$ est *d'ordre N* ;
- d'un ensemble $U$ dont les éléments $u \in U$ sont des couples ordonnés de sommets appelés *arcs*. Si $u = (i, j)$ est un arc de $G$, $i$ est l'*extrémité initiale* de $u$ et $j$ l'*extrémité terminale* de $u$.

Un *graphe orienté étiqueté* $G = [X, E, V]$ est déterminé par la donnée :

- d'un ensemble $X$ de noeuds ;
- d'un ensemble $E$ d'étiquettes ;
- d'un ensemble $V$ dont les éléments $v \in V$ sont des couples constitués :
  - d'un couple d'éléments de $X$ ;
  - d'un élément de $E$.

Par exemple, dans notre cas, le graphe des documents sera un graphe étiqueté, et $E$ sera composé de l'ensemble des relations possibles entre documents.

Un *chemin* (ou *parcours*) de longueur $q$ est une séquence de $q$ arcs ; soit $P = \{u_1, u_2, \ldots, u_q\}$ cette séquence. On a :

- $u_1 = (i_0, i_1)$
- $u_2 = (i_1, i_2)$
- $\ldots$
- $u_q = (i_{q-1}, i_q)$

Autrement dit, un chemin est une chaîne dont tous les arcs sont orientés dans le même sens.

Les définitions ci-dessous sont celles que nous rajoutons pour le problème dont nous traitons ici.

Un *sous-parcours* est un sous-ensemble d'éléments successif d'un chemin $P$. Par exemple $u_2, u_3, u_4$ est un sous-parcours de $\{u_1, u_2, \ldots, u_q\}$.

Un *métagraphe*, littéralement un graphe de graphe, est un graphe dont chaque noeud est lui-même associé à un graphe appelé *sous-graphe* du métagraphe. Un métagraphe se construit à partir d'un graphe orienté en regroupant les noeuds du graphe dans différents noeuds du métagraphe. Chaque noeud du graphe initial peut faire partie de plusieurs noeuds du métagraphe. Un arc $(m_i, m_j)$ relie alors deux noeuds du métagraphe si et seulement si tous les noeuds sans successeur du sous-graphe de $m_j$ sont reliés à au moins un noeud sans prédécesseur du sous-graphe de $m_j$.

Le *noeud initial* d'un graphe orienté est le noeud $i$ tel que : $\forall x \in X : (x, i) \notin U$. S'il existe plusieurs noeuds $i_1..i_n$ vérifiant cette propriété, on rajoute le noeud $i$ tel que : $\forall j = 1..n,\ (i, i_j) \in U$ et $\forall x \in X \setminus \{i_1..i_n\},\ (i, x) \notin U \wedge (x, i) \notin U$.

Un *objectif* du graphe $G$ est un sous-ensemble non-vide $O$ de sommets. Nous utiliserons la notion d'objectif dans le graphe des documents. Si $O$ est un singleton, on parle d'*objectif simple*, sinon on parle d'*objectif composite*.

L'*objectif final* est l'ensemble de noeuds du graphe qu'il faut que l'utilisateur atteigne.

Un *objectif intermédiaire* est un objectif qui n'est pas l'objectif final, mais qui constitue un ensemble de noeuds sur lequel il est intéressant de terminer une succession d'étapes.

Un *document* est un noeud du graphe des documents. Dans la pratique, il peut s'agir de fichiers HTML, PDF ou tout autre média...

### 5.5.2 Problème posé

Dans la plupart des hypermédias adaptatifs, on permet à l'utilisateur de choisir l'étape suivante dans son parcours en fonction de ses connaissances, de ce qu'il vient d'apprendre, et plus généralement de sa position dans le graphe des états, ou alors on le guide totalement.

Le problème qui nous est posé est de pouvoir prévoir des tronçons de parcours contenant plusieurs étapes fixées à l'avance. On peut fixer le nombre d'étapes à proposer pour chaque tronçon. On peut aussi se donner une fourchette et essayer de sélectionner au mieux un parcours en fonction des informations dont on dispose sur les documents. On souhaite pouvoir contraindre l'utilisateur à un seul tronçon possible, ou lui laisser le choix entre plusieurs tronçons.

### 5.5.3 Méthode suivie

Nous devons tenter, dans un premier temps, de trouver des objectifs intermédiaires pertinents sur lesquels l'utilisateur puisse arrêter son cheminement. Puis, dans un second temps, nous souhaitons pouvoir regrouper les documents qui mènent à des objectifs intermédiaires. Ce regroupement nous fournit un métagraphe de documents à parcourir. Chaque noeud du métagraphe des documents est lui-même un graphe de documents, dans lequel on va effectuer une recherche de plan correspondant aux règles choisies. On aura ainsi une recherche sur un nombre restreint de documents groupés selon des règles correspondant au problème que pose le domaine d'application, et ces documents mèneront toujours vers un véritable objectif intermédiaire.

En résumé, les problèmes à résoudre sont les suivants :

- recherche d'objectifs intermédiaires ;
- regroupement de documents menant à un objectif intermédiaire et construction d'un métagraphe pour les documents ;
- parcours du métagraphe ;
- parcours des graphes qui constituent les noeuds du métagraphe.

## 5.6 Solution proposée

La solution que nous envisageons est découpée, comme nous venons de le voir, en quatre étapes. La première consiste à chercher des objectifs intermédiaires. Une fois ces objectifs intermédiaires déterminés, on cherche à regrouper des documents qui mènent à ces objectifs. On obtient, par ce regroupement, un métagraphe. Il faut dans un troisième temps parcourir cet métagraphe depuis des

noeuds accessibles à l'utilisateur jusqu'à l'objectif qui lui est proposé. Enfin, chaque noeud du métagraphe étant un graphe, il faut fournir des règles pour parcourir ses graphes et poser les principes d'utilisation de ces règles. Nous allons étudier plus en détail chacun de ces points dans la suite de cette section.

### 5.6.1 Recherche d'objectifs intermédiaires

Il nous paraît intéressant de déterminer des objectifs intermédiaires non-aléatoires puisque l'on dispose d'informations sur les documents. Si l'on dispose directement de documents étiquetés comme objectifs intermédiaires, on les choisira. Si l'information concernant ces documents est moins explicite, on pourra appliquer des règles dépendant du domaine pour trouver les objectifs intermédiaires.

Dans cette sous-section, nous allons d'abord présenter la recherche d'objectifs intermédiaires dans le cas où l'on a des candidats prédéterminés, puis la recherche de tels objectifs dans le cas où l'on ne dispose pas d'informations suffisantes pour donner des objectifs intermédiaires a priori. Enfin nous présenterons une solution pour le cas où on ne dispose d'aucune information sur les documents, si ce n'est le graphe qui les lie.

Dans toute-cette partie, «le graphe» fait référence au graphe des documents.

#### Sélection d'objectifs intermédiaires parmi des candidats prédéterminés

On suppose que l'on dispose du graphe orienté des documents et d'un certain nombre de documents correspondant à des objectifs intermédiaires potentiels, ou de règles permettant de déterminer directement ces objectifs intermédiaires potentiels. Certains documents constituent à eux seuls un objectif intermédiaire, dit objectif simple, d'autres objectifs peuvent être constitués d'un ensemble de documents, qui constituent un objectif composite.

Dans ce cas, la seule chose à faire est de vérifier que les objectifs intermédiaires ne sont pas trop proches les uns des autres dans le graphe. On se donne une distance minimale, et on vérifie que tous les objectifs intermédiaires la respectent :

- On compte, pour chaque objectif intermédiaire, le nombre d'objectifs intermédiaires trop proches de lui dans le graphe. On considère que la proximité entre deux documents dans le graphe est le plus court chemin qui mène de l'un des deux à l'autre. Certains documents peuvent être séparés d'une distance infinie si aucun chemin ne mène de l'un à l'autre.
- On supprime celui qui a le plus d'objectifs intermédiaires trop proches (s'ils sont plusieurs dans ce cas, un de ceux-ci aléatoirement). Calculer le nombre de documents proches à la fois avant et après dans le graphe orienté permet de supprimer en premier les noeuds qui contribuent le plus à de trop grandes proximités dans le graphe entre les objectifs intermédiaires. Ainsi, l'algorithme résoud le problème plus rapidement et en supprimant un minimum d'objectifs intermédiaires potentiels.

– On réitère le processus jusqu'à obtenir des objectifs intermédiaires tous suffisament éloignés les uns des autres.

Dans l'exemple décrit sur la figure 5.1, on considère que les documents 8 et 9 sont des objectifs intermédiaires potentiels. Leur distance dans le graphe est infinie, aucun chemin ne menant de l'un à l'autre. Ils sont donc tous deux retenus comme objectifs intermédiaires.

Pseudo-code de l'algorithme :

```
FAIRE
  Pour chaque noeud candidat n[i] FAIRE
    c[i] = nombre de candidats trop proches de n[i]
  finFAIRE
  SI tous les c[i] sont nuls, BREAK
  j = valeur de i qui maximise c[i]
  supprimer n[j] des candidats
finFAIRE
```

**Sélection d'objectifs intermédiaires en utilisant des informations sur les documents**

Ici, on ne dispose pas d'informations permettant de dire a priori si un document peut constituer ou non un bon élément pour faire partie d'un objectif intermédiaire. La solution que nous proposons consiste à définir des règles pour la sélection d'objectifs intermédiaires en fonction du domaine à traiter. Par exemple on peut décider qu'un bon objectif intermédiaire est un document traitant d'un concept et qui n'est le pré-requis d'aucun document traitant de ce concept, mais qui est le pré-requis d'un ou plusieurs document(s) traitant d'autres concepts.

Dans un premier temps, il faut donc préciser les critères à utiliser pour créer les groupes de documents. Ces critères dépendent du contexte et sont à définir au cas-par-cas, par exemple sous forme de règles.

Une fois les critères déterminés, on utilise l'algorithme suivant :

1. On prend un noeud situé à une distance donnée $d$ du noeud initial, que l'on utilise comme centre de la recherche de groupe et on cherche des documents qui lui sont rattachés au sens des règles définies. On obtient un groupe de documents.
2. On cherche ensuite un un objectif intermédiaire pour ce groupe. Pour ce faire on va déterminer les documents finaux du groupe, c'est-à-dire ceux qui ne sont les prédécesseurs d'aucun autre document du groupe. On regroupera éventuellement plusieurs de ces documents finaux pour former un objectif composite.
3. On procède ensuite en se déplaçant de proche en proche depuis les noeuds constituant les objectifs intermédiaires en avaçant de $d$ à chaque fois, et en reprenant alors les étapes précédentes. On parcourt ainsi tout le graphe deux fois, une fois pour avancer, et une fois -par morceaux - pour chercher les documents qui n'ont pas de successeurs dans le regroupement.

4. Une fois les objectifs intermédiaires déterminés, on procèdera au regroupement de documents (cf. 5.6.2).

Dans l'exemple de la figure 5.1, on choisirait les documents nécessaires pour chacun des deux concepts. On aurait ainsi les documents 1, 2, 3, 4 et 8 pour le concept «dériver» et 1, 5, 6, 7, et 9 pour le concept «intégrer». Le document 10 ne fait partie d'aucun des deux groupes car il a des pré-requis dans les deux parties. Le document 8 (resp. 7 et 9) étant les plus avancés pour le groupe «dériver» (resp. «intégrer»), on les choisit comme objectifs intermédiaires.

Pseudo-code de l'algorithme :

```
nc = document initial
listenoeud = tous les noeuds à une distance
             d prédéfinie de nc
POUR chaque n de listenoeud
  Fnoeud = tous les noeuds rattachés à
           n au sens de règles
  Objectif = ensemble des noeuds de Fnoeud
             sans successeur dans Fnoeud
  Listenoeud = Listenoeud + ensemble des noeuds
      situés à une distance d des noeuds de Objectif
finPOUR
```

*Remarque* : Cette méthode pourrait laisser penser que l'on parcourt le graphe pour chercher des objectifs intermédiaires alors que l'on pourrait les chercher au fur et à mesure du parcours des documents. En fait, les arcs étiquetés utilisés pour chercher les objectifs intermédiaires ne sont pas nécessairement les mêmes que ceux utilisés pour chercher de bons parcours dans le graphe. Par exemple, on peut utiliser la notion de pré-requis pour chercher de bons parcours dans le graphe, alors que les règles utilisés pour chercher les objectifs intermédiaires utiliseront la notion «fait partie de». En résumé, on parcourt dans chaque cas des projections différentes du graphes de documents étiquetés.

**Sélection d'objectifs intermédiaire dans le cas où on ne dispose pas d'information sur les documents**

On suppose dans cette partie que l'on ne dispose que d'un graphe reliant les documents entre eux, mais pas d'information sur les documents comme par exemple les concepts dont ils traitent. On va donc chercher des documents vers lesquels convergent un maximum d'arcs du graphe des documents et/ou ceux depuis lesquels partent un minimum d'arcs. En effet on considère qu'une conclusion partielle termine un certain nombre de parcours partiels, et mène ensuite vers un nombre restreint de nouveaux parcours potentiels.

L'algorithme proposé est le suivant :

1. Pour chaque document, on compte le nombre d'arcs entrant et sortant. On sélectionne un pourcentage de ces documents en fonction du nombre de tronçons à fournir, i.e. de la taille du graphe sur celle des tronçons, et d'une règle à fournir sur le poids des arcs entrants et sortants.
2. Si deux ou plusieurs documents sélectionnés sont trop proches dans le graphe, on en élimine certains pour qu'il existe toujours une distance minimum entre deux objectifs intermédiaires.
3. Une fois les objectifs intermédiaires sélectionnés, on regroupe les documents (cf. 5.6.2).

On considère que dans l'exemple de la figure 5.1, on ne dispose que du graphe avec les documents numérotés, mais que l'on ne possède pas d'autres informations sur nos documents. La simplicité du graphe fait qu'on pourrait considérer presque tous les documents comme des objectifs intermédiaires selon l'algorithme présenté. On note tout de même que les documents ayant le plus de noeuds entrant sont le document 10 (objectif final) et le document 9, qui était un des objectifs intermédiaires des cas précédents.

Pseudo-code de l'algorithme :

```
POUR chaque noeud n du graphe
  coefficient c[n] = f(n, nombre d'arcs entrants,
                       nombre d'arcs sortants)
finPOUR
Annoter comme noeud candidat les noeuds n tels que c[n]
        dépasse un seuil
 FAIRE
  Pour chaque noeud candidat n[i] FAIRE
    c[i] = le nombre de candidats trop proches de n[i]
  finFAIRE
  SI tous les c[i] sont nuls, finFAIRE
  j = indice qui maximize c[i]
  supprimer n[j] des candidats
finFAIRE
```

Notons que cette méthode peut s'avérer hasardeuse et ne constitue qu'un essai pour palier à un possible défaut d'annotation des documents. Néanmoins, dans tout système d'hypermédia adaptatif, il est préférable que les documents soient annotés le plus possible.

### 5.6.2 Construction du métagraphe

Une fois les objectifs intermédiaires déterminés, nous souhaitons en déduire des regroupements de documents menant à ces objectifs intermédiaires.

Nous considérerons que les documents à regrouper pour un objectif intermédiaire donné sont l'ensemble des documents qui y mène sur le graphe initial, en s'arrêtant aux objectifs intermédiaires

précédents. De ce fait, un document peut faire partie de plusieurs regroupements. Pour simplifier le problème, dans toute la suite, on considèrera qu'un document qui apparaît dans deux groupes distincts existe en deux instances dans le système.

Voici l'algorithme que nous utiliserons pour créer les regroupements de documents voulus :

1. On sélectionne un objectif intermédiaire, c'est-à-dire un ou plusieurs documents constituant cet objectif.
2. On ajoute récursivement leurs prédécesseurs directs au regroupement, puis les prédécesseurs au sens large des prédécesseurs directs.
3. Pour chaque noeud inclus, on parcourt les branches qui partent de ce noeud. Si une branche courte, i.e. une branche qui ne mène pas à un objectif intermédiaire apparaît, on inclut ses noeuds également : selon les règles utilisées, ils peuvent servir. Par exemple si les règles disent qu'il faut parcourir tous les documents d'un groupe avant d'accèder à l'objectif, il peut être intéressant de garder des noeuds qui ne sont pas des prédécesseurs de l'objectif.
4. On s'arrête de parcourir une branche du graphe quand on tombe sur un document constitutif d'un autre objectif intermédiaire.
5. On s'arrête quand on s'est arrêté pour toutes les branches. L'ensemble des documents regroupés constitue un noeud du métagraphe en construction.

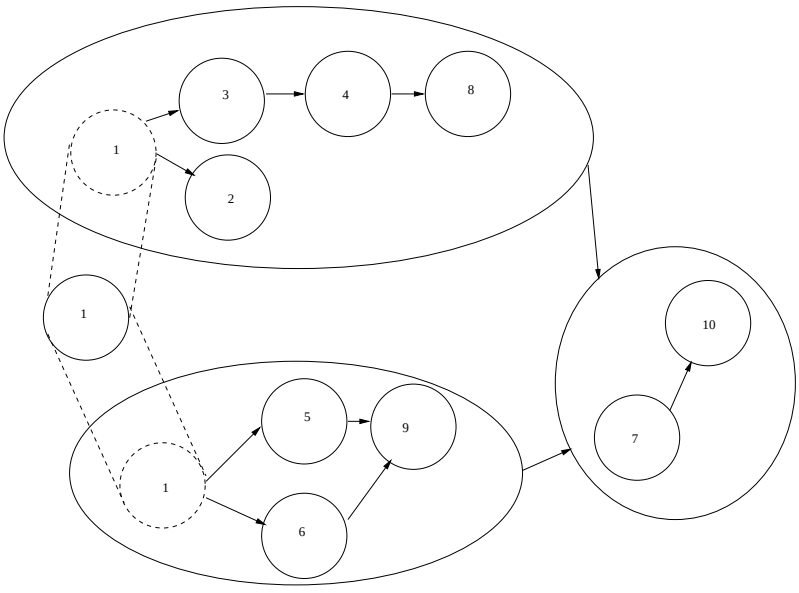


FIG. 5.3 – Métagraphe construit à partir du graphe des documents

Les arcs du métagraphe sont obtenus de la façon suivante : on place un arc entre deux noeuds

$A$ et $B$ du métagraphe, si, et seulement si, il existe un arc dans le graphe initial entre un objectif intermédiaire de $A$, et un noeud quelconque de $B$.

Appliquons cette méthode à l'exemple de la figure 5.1. On suppose que les exercices constituent des objectifs intermédiaires prédéterminés. On remonte depuis le document 8 (resp. 9) jusqu'au document 1. On obtient donc un premier regroupement constitué des documents 1, 2, 3, 4 et 8, et un second constitué des documents 1, 5, 6 et 9. Les documents 7 (utilisation de l'intégration pour le calcul d'aire, non nécessaire aux exercices) et le test (objectif final) ne font pas partie de ces regroupements. On constitue ensuite le dernier regroupement, fait des documents 7 et 10. La figure 5.3 représente le métagraphe obtenu.

Pseudo-code de l'algorithme :

```
PARCOURIR l'ensemble du graphe
  POUR chaque objectif intermédiaire o
    Trouver les predecesseurs pred[o][i] de o
    SI pred[o][i] n'est pas un objectif intermédiaire
      Groupe de documents associé à o :
  g[o] += p[o][i]
    finSI
    Chercher récursivement tous les prédécesseurs
    de p[o][i]
    Chercher les branches courtes à inclure
    à partir de chaque noeud ajouté
    S'arrêter quand on tombe sur des objectifs intermédiaires
  finPOUR
finPARCOURIR
```

### 5.6.3 Parcours du métagraphe

Plusieurs possibilités s'offrent à nous pour le parcours du métagraphe. Selon le contexte on pourra vouloir parcourir le métagraphe le plus vite possible, utiliser un ordre prédéfini ou appliquer des règles pour trouver un parcours adéquat.

Dans le cas d'un plus court chemin, on se reservira d'algorithmes existant déjà, optimisés et utilisés actuellement.

#### Parcours du métagraphe

On peut, selon le contexte, envisager plusieurs jeux de règles permettant le parcours du métagraphe. En voici quelques exemples.

*Exemple 1*

Dans ce premier exemple on considère que l'on a un cours, et que pour valider ce cours, il faut avoir tout appris, en respectant la notion de pré-requis. On appliquera les règles suivantes :

- Un noeud non-final du métagraphe est 'bon à parcourir' s'il n'a pas déjà été parcouru, s'il est accessible depuis un état déjà parcouru du métagraphe et s'il amène vers un noeud final et si on connaît déjà ses pré-requis.
- Un noeud final du métagraphe est 'bon à parcourir' si tous les noeuds non-finaux ont été parcourus. (Ici, on rajoute implicitement des arcs de type «pré-requis» au graphe des documents puisque tout devient pré-requis de chaque noeud de l'objectif final).
- Un noeud est mauvais si au moins un de ses pré-requis n'a pas été parcouru par l'utilisateur.

Dans notre exemple de graphe, on pourra choisir indifférement le noeud du métagraphe contenant 1, 2, 3, 4 et 8 ou celui contenant 1, 5, 6 et 9 pour commencer, puis on devra nécessairement choisir l'autre, puisque le noeud contenant 7 et 10 a ces deux noeud pour pré-requis. Enfin, on accèdera au noeud final (7 et 10).

*Exemple 2*

Dans ce second exemple, on considère qu'un chemin amenant directement rapidement à l'objectif est un meilleur chemin, en respectant la notion de pré-requis :

- Un noeud non-final du métagraphe est 'bon à parcourir' s'il n'a pas déjà été parcouru, s'il est accessible depuis un état déjà parcouru du métagraphe et s'il amène vers un noeud final et si on connaît déjà ses pré-requis.
- Un noeud final du métagraphe est 'bon à parcourir' si tous ses pré-requis sont connus.
- Un noeud est meilleur qu'un autre si le noeud courant est un précurseur direct de ce noeud.
- Un noeud est mauvais si au moins un de ses pré-requis n'a pas été parcouru par l'utilisateur.

### 5.6.4 Conclusions sur les tronçons de parcours

Dans les hypermédias adaptatifs qui existent actuellement, la prévision de tronçons de parcours n'est pas envisagée. L'utilisateur détermine, en général, les étapes de son parcours une à une. Le fait de proposer des tronçons de parcours fixés permet d'utiliser les règles pour guider de manière plus intelligente l'utilisateur.encore été envisagé.

De plus, le regroupement de documents par règles constitue également une nouveauté. En effet, le plus souvent, le domaine des documents est considérés globalement, sans subidivision. On le parcourt donc dans sa globalité, sans se soucier du rapport que peuvent avoir les documents entre eux selon le contexte, mis-à-part les relations décrites dans le graphe des documents. Certains systèmes empêchent d'accèder à un chapitre si le précédent n'a pas encore été appris, ce qui confère au système une certaine linéarité qui n'est pas toujours indispensable. En regroupant les documents en fonction des informations (sémantiques, structurelles...) qu'on peut avoir sur chacun d'entre eux, nous proposons une solution intermédiaire, qui ne nécessite pas de regroupement explicite de documents, par exemple en chapitres, mais qui permet d'éviter à l'utilisateur de se disperser dans tout

l'ensemble des documents proposés dans le système d'hypermédia adaptatif.

Nous avons donc proposé dans cette partie une méthode de guidage originale, permettant de modifier à «moindre frais» un système qui existerait déjà. Les éléments à rajouter dans un système conçu sans la méthode de tronçons de parcours sont :

- Des règles de sélection des objectifs intermédiaires ou une sélection manuelle d'objectifs intermédiaire ;
- Des règles de parcours du métagraphe.

Notre technique constitue donc une surcouche d'adaptation, qui peut s'appliquer aussi bien à notre modèle d'adaptation qu'aux systèmes qui existent déjà.

## 5.7 Conclusions générales

Dans ce chapitre, nous avons abordé sous un angle théorique le problème des hypermédias adaptatifs. Cet aspect théorique est rarement abordé. L'étude de Nicola Henze **??** fait néanmoins exception. Sa caractérisation logique des hypermédias adaptatifs n'offre néanmoins pas la possibilité de construire un modèle d'adaptation utilisable dans un système d'hypermédia adaptatif. Ce que nous proposons n'est pas une caractérisation logique de l'adaptation, mais un modèle complet basé sur le calcul des prédicats du premier ordre. Cette approche logique offre plusieurs avantages : elle est basée sur un formalisme bien fondé, donc à vocation universelle. Ce formalisme est déclaratif, donc indépendant de toute approche programmatique, de tout langage particulier.

Nous avons décrit une base de raisonnement pour la sélection et la classification d'action utilisant une version légèrement modifiée de la logique situationnelle. La logique situationnelle est base sur le calcul des prédicats du premier ordre. Elle définit un certain nombre de prédicats, ainsi que la façon dont on raisonne sur ces prédicats. La logique situationnelle offre l'intérêt de permettre de décrire, de manière très déclarative, un système représentant directement les aspects réels du problème. La représentation des situations, des actions, de la possibilité des actions est très proche de la façon naturelle de décrire ses aspects.

Plutôt que de réutiliser les clauses de Horn très générales qui sont le plus souvent utilisées, pour des raisons de langage de programmation, pour définir les règles à appliquer dans le moteur de logique situationnelle, nous avons choisi de définir notre propre formalisme de règles. Ce formalisme est spécifiquement conçu pour les hypermédias adaptatifs. Il utilise les différents types de données présents dans ces systèmes, à différents niveaux du modèle de raisonnement. La nature des données mises en jeu peut-être vérifiée.

Le système de méta-règles permet de séparer à différents niveaux les différents aspects de l'adaptation. Les méta-règles permettent la sélection de règles formant des stratégies d'adaptation. Des vérifications permettant la conception d'un système viable sont effectuées et détectent d'éventuelles erreurs. De plus, nous avons défini un système de déduction formel qui donne un sens sans la moindre

ambiguïté au système de règle. Il est possible de définir plusieurs systèmes de déduction, qui fournissent plusieurs stratégies distinctes pour l'adaptation.

Nous avons donc fourni un modèle d'adaptation, décrit de manière entièrement formelle, et sur lequel on sait comment raisonner. Nous avons réutilisé ce modèle pour définir une technique innovante d'adaptation, basée sur un découpage raffiné des documents du domaine. Ce découpage permet de fournir à l'utilisateur une adaptation fortement guidée, ainsi que des étapes intermédiaires pour clore sa session. Cette méthode d'adaptation peut être adaptée pour être appliquée dans d'autres systèmes que notre système à base de logique situationnelle.

# Chapitre 6

# Exemple d'implémentation

Dans ce chapitre, nous mettons en avant la démarche que nous avons mise en oeuvre pour créer un cours sur les bases de données adaptatifs en utilisant l'ensemble des propositions que nous avons émises dans les chapitres 3, 4 et 5.

Pour modéliser nos données, nous utilisons les modèles pour le e-learning que nous avons proposé dans les chapitres 3 et 4, et qui spécialisent les modèles génériques décrits dans ces mêmes chapitres.

Nous avons défini les éléments nécessaires à l'adaptation selon notre modèle à base de logique situationnelle : actions, observateurs, règles, méta-règles etc. Nous avons ainsi obtenu un système capable de fournir différentes sortes d'adaptation : guidage direct, annotation de liens, masquage de liens, tri de liens, composition de documents à partir de fragments. Ainsi, notre cours propose-t-il des éléments essentiels d'adaptation, aussi bien au niveau du parcours que de la composition des documents.

Nous allons détailler les trois composants du système dans la suite de ce chapitre. Nous allons décrire les quelques attributs non-prédéfinis que nous avons utilisé pour modéliser les utilisateurs ; nous allons donner les spécifications détaillé de notre domaine sur les systèmes de gestion de bases de données ; enfin, nous allons décrire les éléments logiques permettant d'adapter le contenu de l'application à l'utilisateur.

## 6.1 Architecture du système

La figure 6.1 présente l'architecture de notre système. Au plus haut niveau d'abstraction, appelé couche générique, nous trouvons nos modèles génériques pour l'utilisateur et le domaine. Nous trouvons également le modèle d'adaptation que nous avons détaillé au chapitre 5. Au niveau suivant, appelé couche modèle, nous trouvons nos modèle d'utilisateur et de domaine spécifiques au e-learning, qui spécialisent les modèles génériques.

Au niveau de la couche système, nous voyons qu'il est nécessaire de décrire le domaine d'appli-

cation (ici, un cours sur les bases de données), et les utilisateurs. Il est également indispensable de décrire l'adaptation que l'on souhaite fournir. Pour traiter toutes ses données, un moteur d'inférence, implémentant les principes décrits au chapitre 5 doit être fourni. Enfin, une interface interprétant les actions sélectionnées permet de visualiser les meilleurs documents possibles.

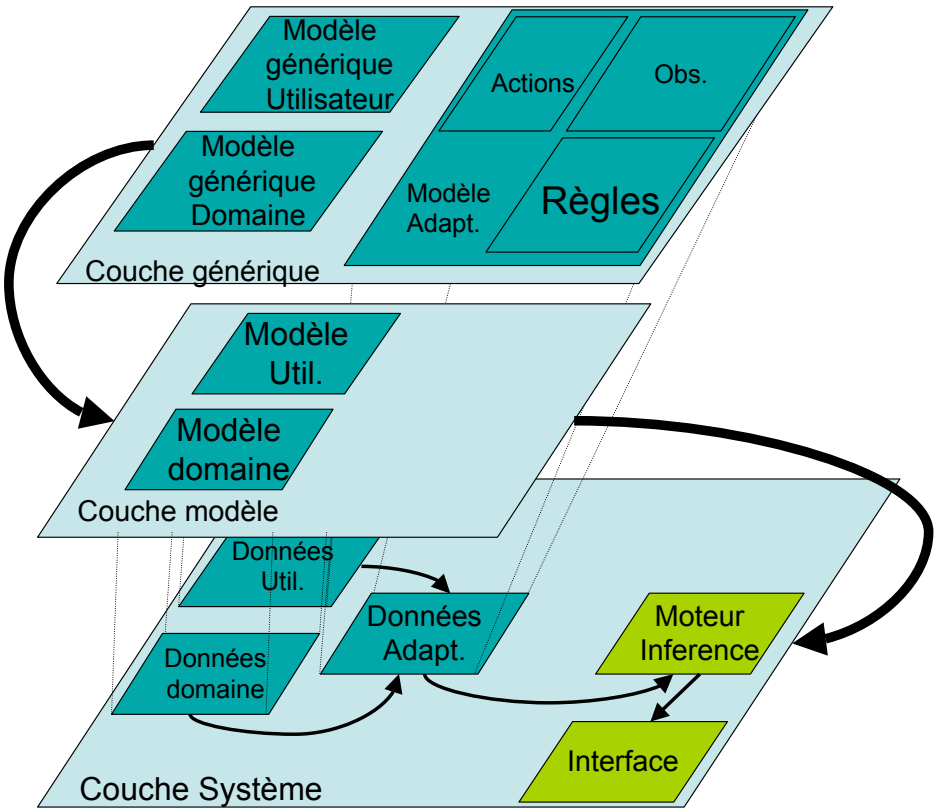


FIG. 6.1 – Optimisations dans les bases de données

## 6.2 Attributs de l'utilisateur

Dans cette section, nous détaillons les stéréotypes d'utilisateur que nous utilisons. Ces stéréotypes vont ensuite nous servir pour donner nos méta-règles (ils seront nos critères de déclenchement), et, après leur application, pour obtenir des ensembles de règles adaptés à chaque utilisateur. Nous donnons ci-dessous la liste des noms des stéréotypes, puis la liste des valeurs possibles pour chaque stéréotype. À côté de chaque valeur, nous indiquons en **gras** l'identifiant du critère de déclenchement de règle associé à cette valeur.

– Vitesse d'apprentissage. Ce stéréotype caractérise la rapidité à laquelle un utilisateur est capable d'assimiler des notions. Ainsi, un utilisateur rapide aura-t-il moins besoin d'exemple,

d'illustration, d'exercice etc. pour assimiler un concept. Les valeurs possibles pour ce stéréotype sont lent (**Vit-Lent**) et rapide (**Vit-Rapide**). La valeur de ce stéréotype peut être indéterminée (**IND**).

- Capacité d'abstraction. Ce stéréotype caractérise la capacité d'un utilisateur à assimiler un concept avec le moins possible d'éléments concrets à sa disposition. Les valeurs possibles pour ce stéréotype sont abstrait (**Abstr-Positif**), intermédiaire (**Abstr-Neutre**) et concret (**Abstr-Negatif**). La valeur de ce stéréotype peut être indéterminée (**IND**).
- Notion du détail. Ce stéréotype caractérise la capacité d'un utilisateur à approfondir une notion le plus possible avant d'en aborder une autre. Certains utilisateurs préfèrent cette façon d'aborder de nouveaux concepts, alors que d'autres préfèrent avoir en priorité une vue d'ensemble, avant d'aller plus en profondeur. Les valeurs possibles pour ce stéréotype sont "profondeur d'abord" (**Detail-Prof.**) et "largeur d'abord" (**Detail-Largeur**). La valeur de ce stéréotype peut être indéterminée (**IND**).
- Niveau d'objectif. Ce stéréotype caractérise le niveau de profondeur des connaissances que l'utilisateur doit acquérir sur les bases de données. Nous avons deux sortes d'étudiants : certains vont étudier l'informatique en 3ème année, et doivent aborder l'ensemble des notions de ce domaine, alors que les autres n'ont pas besoin d'atteindre ce niveau de détail. Les valeurs possibles pour ce stéréotype sont "spécialité informatique" (**3A-Info**) et "autre spécialité" (**3A-Autre**). La valeur de ce stéréotype est toujours déterminée.
- Besoin d'exercices. Ce stéréotype caractérise le besoin d'exercices de l'utilisateur. Les valeurs possible pour ce stéréotype sont "besoin" (**BesoinEx**) et "absence de besoin" (**NonBesoinEx**). La valeur de ce stéréotype peut être indéterminée (**IND**).

Nous avons défini cinq stéréotypes, qui prennent en compte différents aspects de l'utilisateur, de sa catégorie d'étudiant à ses préférences pédagogiques en passant par ses capacités intrinsèques. En prenant en compte les éventuels stéréotypes indéterminés, fournir un jeu de règle d'adaptation potentiellement différent pour chaque type d'utilisateur manuellement, i.e. chaque combinaison possible de ces quatre critères reviendrait à donner un ensemble de 216 jeux de règles différents ! Nous allons voir dans la suite de ce chapitre que les méta-règles permettent d'éviter d'avoir à décrire autant d'éléments.

## 6.3 Description du domaine

Dans cette section, nous allons détailler les éléments de notre domaine. Nous allons commencer par décrire le niveau conceptuel, puis le niveau ressource, où nous indiquerons quels documents sont à notre disposition.

### 6.3.1 Niveau conceptuel

Afin de décrire le niveau conceptuel, nous avons procédé de la façon suivante : nous avons d'abord énuméré les concepts utilisés dans notre domaine d'application, puis nous les avons hiérarchisé en utilisant la relation hiérarchique et finalement, nous avons décrit les autres relations entre concepts, telles que la relation de pré-requis.

#### Concepts utilisés

Voici les différents concepts que nous utilisons dans notre application, précédés de leur identifiant :

**sgbd** Système de gestion de base de données (SGBD)

**bd** Base de données (BD)

**SQL** Structure Query Language

**Composants** Composants des systèmes de gestion de bases de données

**Description** Description des données dans un SGBD

**interne** Niveau interne de description des données

**conceptuel** Niveau conceptuel de description des données

**externe** Niveau externe de description des données

**objectifs** Objectifs des SGBD

**independance-physique** Indépendance physique des SGBD

**independance-logique** Indépendance logique des SGBD

**coherence** Cohérence des donées des SGBD

**securité** Sécurité des données des SGBD

**modele-relationnel** Modèle relationnel des BD

**bd-relationnelle** Base de données relationnelle

**difference-rationnelle** Différence rationnelle de deux relations

**attribut** Attribut (colonne) d'une relation

**intersection** Intersection de deux relations

**domaine** Domaine

**langage** Langages de manipulations de données relationnelles

**recherche** Recherche dans une base de données

**insertion** Insertion dans une base de données

**suppression** Suppression dans une base de données

**modification** Mise à jour dans une base de données

**jointure** Jointure de deux relations

**produit-cartesien** Produit cartésien de deux relations

**limitations-relationnelles** Limitation des BD relationnelles

**operateur-relationnel** Opérateur entre relations

**restriction** Restriction des tuples d'une relation

**relation** Relation d'un domaine

**projection** Projection d'une relation (suppression d'attributs)

**schema-relation** Schéma de relation

**sgbdr** Système de gestion de bases de données relationnel

**tuple** Tuple, ligne d'une relation

**union** Union de deux relations

**optimisation-statique** Optimisation statique de requête

**optimisation** Optimisation de requête

**optimisation-physique** Optimisation physique de requête

**reecriture** Réécriture des requêtes

**optimisation-logique** Réécriture des requêtes

**concurrence** Concurrence des accès à une BD

**coherence-conccurence** Cohérence des accès concurrents

**algo-concurrence** Algorithmes de contrôle de la concurrence

**index** Index des BD

**transaction** Notion de transaction dans une BD

**verrous** Verrous dans une BD

**dfe** Dépendance fonctionnelles élémentaires

**decomposition-relation** Décomposition d'une relation

**cle** Clé de relation

**dependance-fonctionnelle** Dépendance fonctionnelle

**conception-bd** Conception de base de données

**modele-entite-associtiation** Modèle entité-association

**forme-normale** Formes normales des BD

**1nf** Première forme normale

**2nf** Deuxième forme normale

**3nf** Troisième forme normale

**bcnf** Forme normale de Boyce-Codd

**graphe-dfe** Graphe des dépendances fonctionnelles élémentaires

**normalisation-relations** Théorie de la normalisation des relations

### Hiérarchie des concepts

Nous allons décrire ici la hiérarchie, en terme de spécialisation et de généralisation, des concepts décrits dans la partie précédente.

Nous présentons sur les schémas ci-dessous les relations entre concepts pour notre domaine. Les flèches vont toujours du concept spécialisé vers le concept général. Nous avons découpé notre hiérarchie en plusieurs schémas, afin d'en faciliter la lecture.

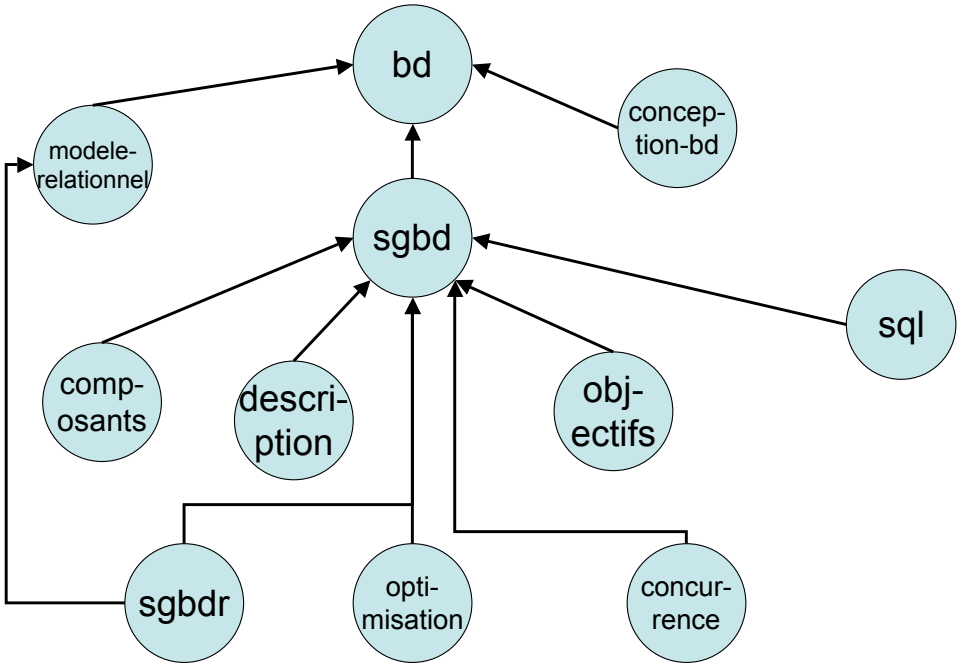


FIG. 6.2 – Niveau le plus haut de la hiérarchie conceptuelle

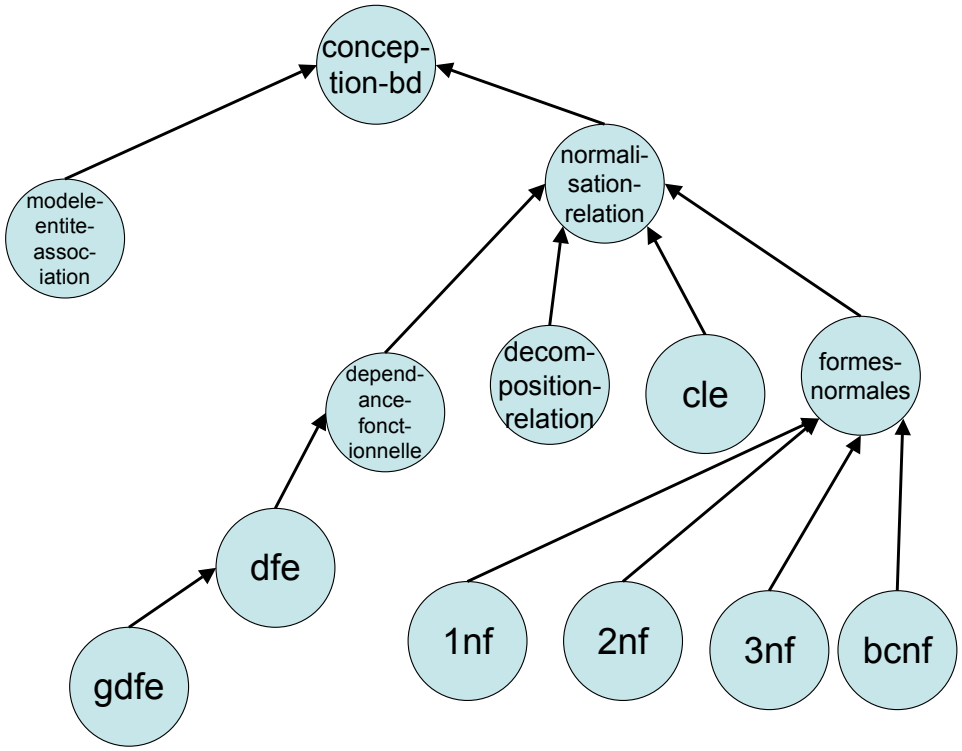


FIG. 6.3 – Conception de bases de données

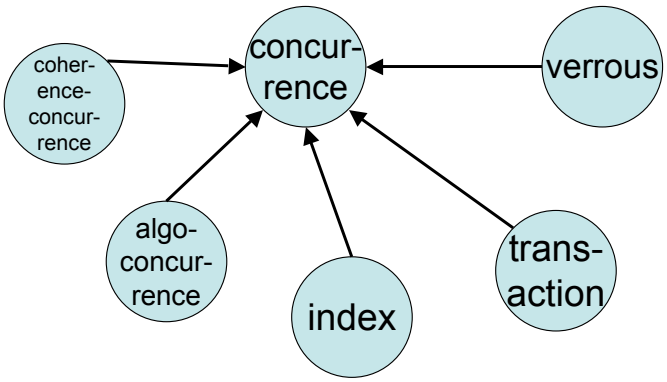


FIG. 6.4 – Concurrence dans les bases de données

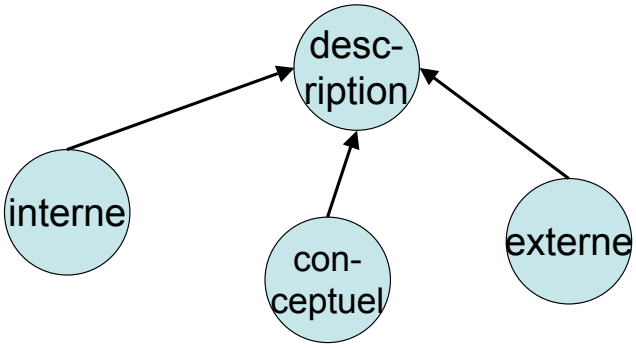


FIG. 6.5 – Description des bases de données

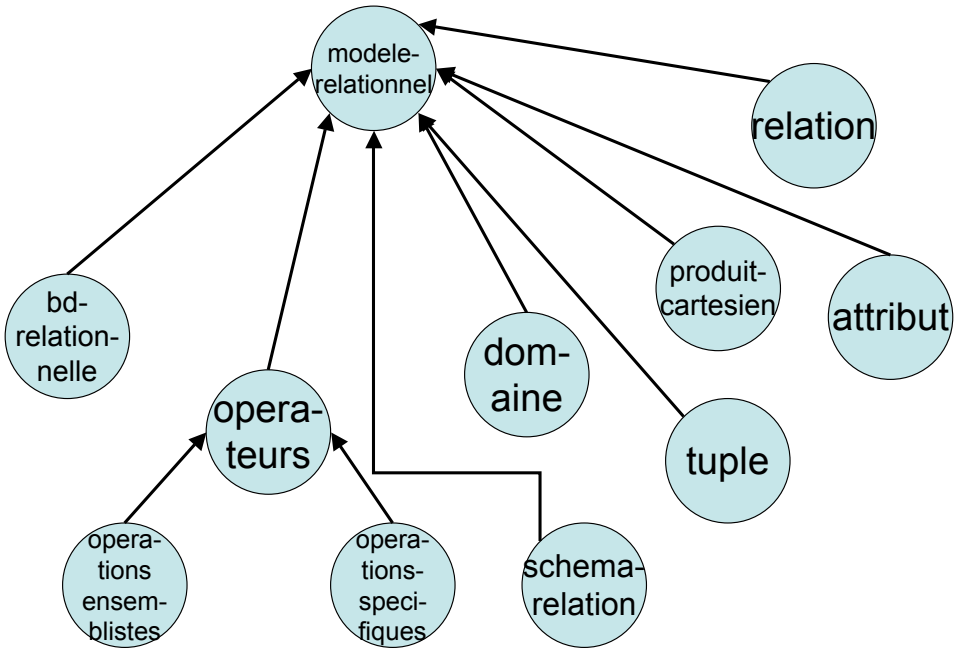


FIG. 6.6 – Modèle relationnel des bases de données

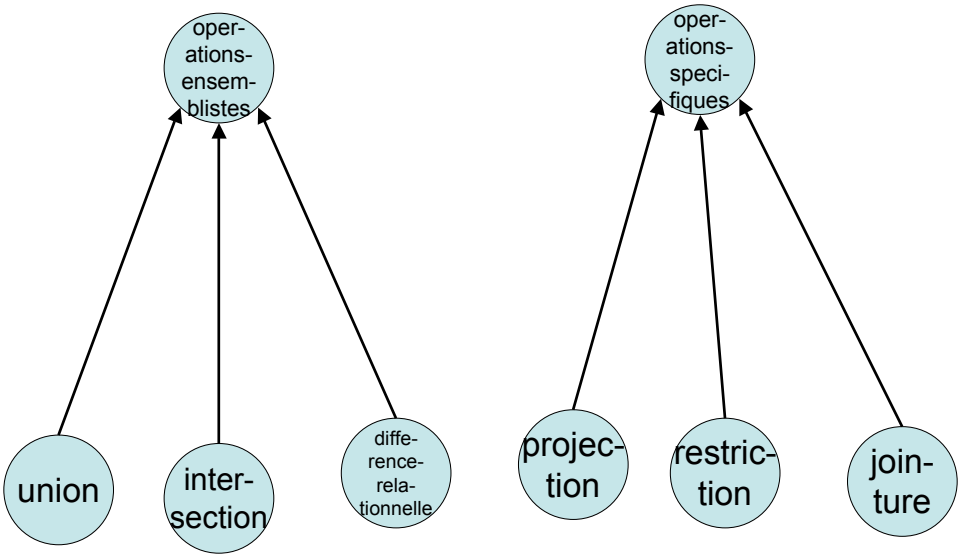


FIG. 6.7 – Opérateurs dans les bases de données relationnelles

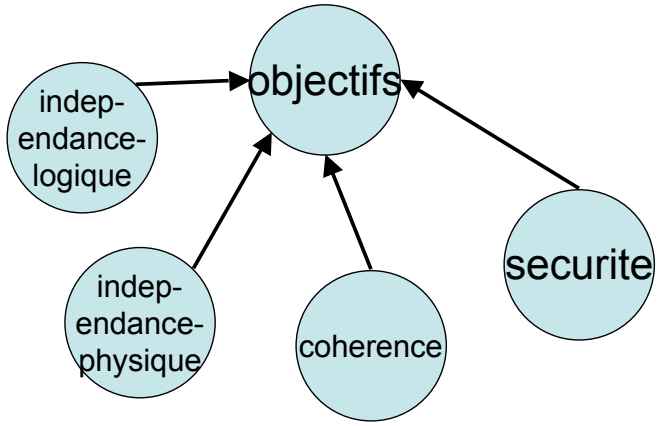


FIG. 6.8 – Objectifs des bases de données

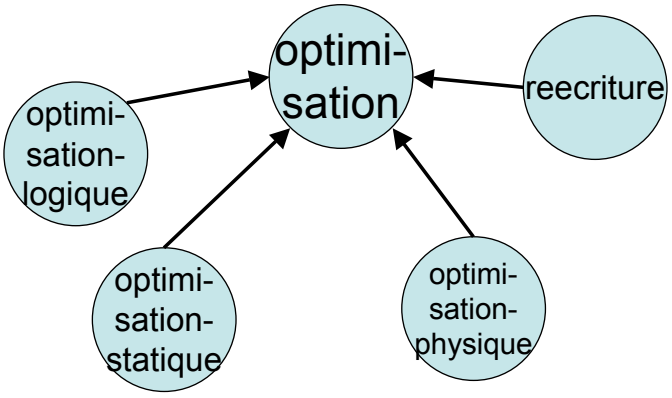


FIG. 6.9 – Optimisations dans les bases de données

**Relations entre concepts**

Nous avons maintenant un domaine abstrait hiérarchisé à notre disposition. Il nous faut décrire quelles autres relations lient les différents concepts. Pour ce faire, nous utiliserons la relation de pré-requis, qui est d'une importance particulière dans les hypermédias adaptatifs. Nous utiliserons également la relation de contraste, puisque certains concepts proposent des alternatives à d'autres concepts.

La hiérarchie des concepts constitue quasiment un arbre, et permettait donc que nous le représentions entièrement. Le graphe des concepts pour la relation de pré-requis est très dense, et même en jouant sur la transitivité de cette relation, la représentation de toutes les relations est très difficile si l'on veut maintenir une lisibilité correcte. Aussi présentons-nous ici quelques extraits de ce graphe, à titre d'illustration. La liste complète des relations de pré-requis est disponible en annexe A. Les exemples que nous proposons ici, sur les figures 6.10, 6.11 et 6.12, illustrent notamment les différences importantes entre la relation hiérarchique, et la relation de pré-requis.

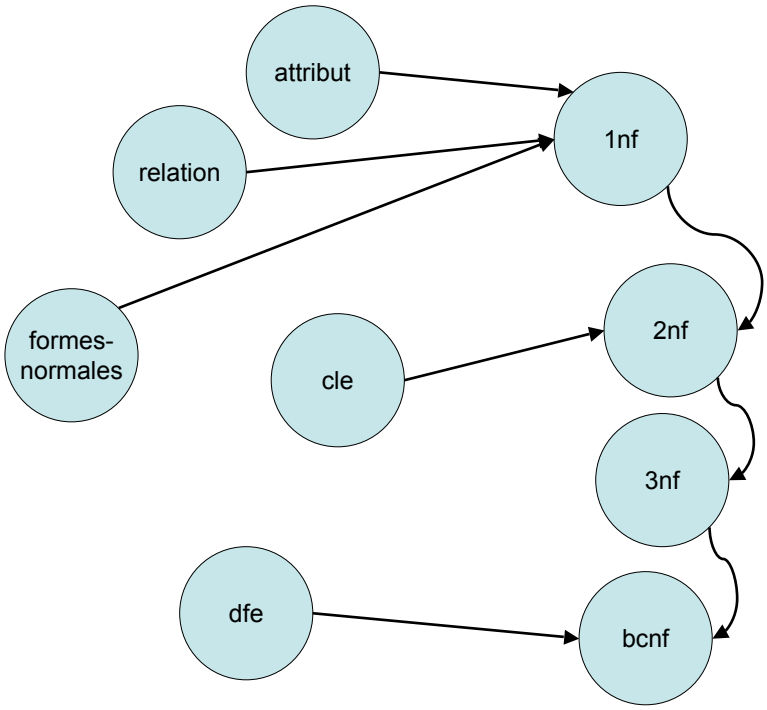


FIG. 6.10 – Relation de pré-requis pour les concepts de forme normale

La liste complète de toutes les relations utilisées pour notre exemple est donnée en annexe A.

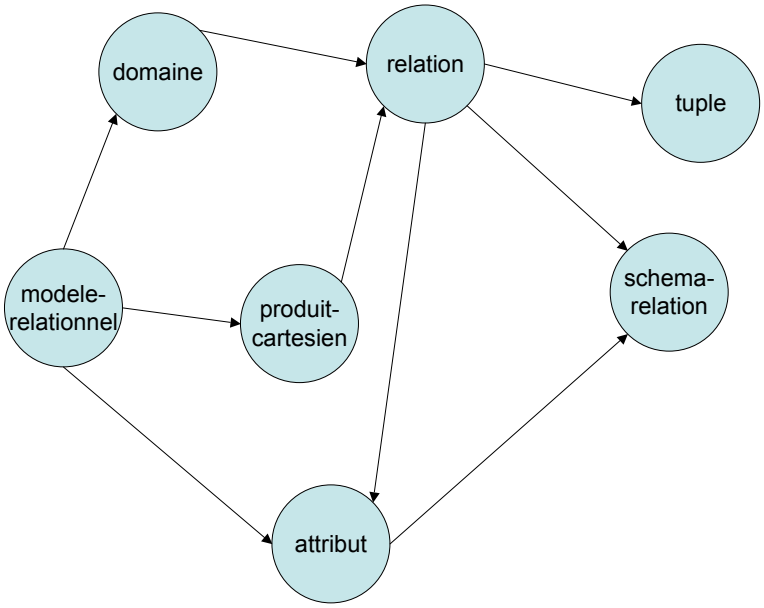


FIG. 6.11 – Relation de pré-requis pour les concepts du modèle relationnel

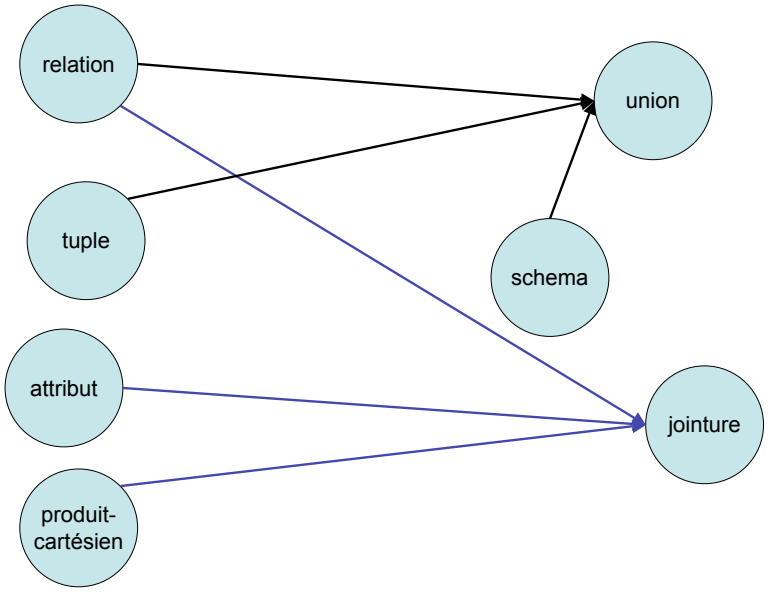


FIG. 6.12 – Relation de pré-requis pour les concepts d'opérations

### 6.3.2 Ressources

Dans cette partie, nous détaillons les principales caractéristiques des ressources que nous manipulons. Ces ressources sont issues pour la plupart d'un cours sur les bases de données antérieur à la création de notre hypermédia adaptatif pédagogique. Nous avons simplement complété ce cours avec quelques exercices supplémentaires.

#### Nature des ressources

Nous utilisons une partie des différents types de ressource définis dans notre modèle de domaine pour le e-learning. Ces types correspondent à ceux qui étaient proposés initialement dans le cours que nous avons réutilisé. Ainsi, nous disposons de **définitions**, d'**explications**, d'**exemples** et d'**exercices**.

#### Méta-données

Parmi les différentes méta-données proposées, nous mettons tout particulièrement en avant la notion de **difficulté** d'une ressource, que nous utilisons particulièrement pour fournir des documents adaptés à l'utilisateur.

#### Relations

Nous avons défini la relation "cite" pour l'ensemble des ressources dont nous disposons. En effet, ces ressources présentent souvent des liens vers d'autres ressources qui abordent des notions connexes. Avoir une carte de ces liens nous permet de masquer des liens vers des notions qui ne devraient pas encore être abordées.

## 6.4 Mécanismes d'adaptation

Dans cette section, nous détaillons les différents mécanismes d'adaptation que nous souhaitons mettre en place. Ainsi, nous allons définir nos actions et nos observateurs, puis nos règles et nos méta-règles.

### 6.4.1 Types d'adaptation

Avant de définir les éléments sus-cités, voici les différentes formes d'adaptation que nous souhaitons mettre en oeuvre dans notre système. Nous souhaitons être capable de fournir des liens annotés à l'utilisateur. Nous souhaitons pouvoir masquer des liens. Nous souhaitons pouvoir trier des liens en fonction de leur pertinence.

Nous souhaitons également être capables de composer des pages, relatives à un concept donné, à partir de différentes ressources. Nous souhaitons que ces ressources puissent être ordonnées les unes

par rapport aux autres. Nous souhaitons également fournir une liste de lien sur la page composée, vers des concepts connexes judicieusement choisis.

Ainsi, nous souhaitons fournir différentes sortes d'adaptation, aussi bien au niveau de la navigation que de la composition des documents, et ce, en définissant correctement les actions utilisées par le système.

### 6.4.2 Actions de la logique situationnelle

Afin de répondre aux exigences que nous nous sommes fixé en matière de forme d'adaptation nous devons être capables de fournir au générateur de pages les éléments suivants :

- dans quel ordre il est préférable d'aborder les différents documents du système ;
- quels sont les éléments qu'il faut absolument éviter d'aborder ;
- quels éléments peuvent être composés, quels autres éléments ne sauraient l'être ;
- quelle est la nature du document à proposer : s'agit-il d'un élément à lire, ou d'un exercice à effectuer ;
- quel est la priorité d'affichage d'un élément à composer avec d'autres éléments ;
- quel lien il peut être intéressant d'indiquer à l'utilisateur sur la page qu'il va consulter.

En retour, l'interface fournit au moteur d'inférence les éléments suivants :

- quel concept a été abordé par le document présenté à l'utilisateur ;
- quel niveau de succès l'utilisateur a-t-il atteint, dans le cas où l'action entreprise consistait à faire un exercice.

Pour prendre en compte l'ensemble de ces éléments, nous définissons les actions suivantes :

- $lire(ressource,\ concept,\ priorite)$ Cette action consiste à proposer la lecture de $ressource$. $ressource$ traite de $concept$. Elle peut être présentée seule ou accompagnée d'autres ressources traitant du même concept. $priorite$ correspond à l'emplacement souhaitable de $ressource$ sur le document, si elle est composée avec d'autres ressources.
- $lireSeul(ressource,\ concept)$ Cette action consiste à proposer la lecture de $ressource$. $ressource$ traite de $concept$. Elle doit être présentée comme un document entier, sans être composée avec d'autres ressources.
- $composer(ressource,\ concept,\ priorite)$ Cette action consiste à proposer la lecture de $ressource$. $ressource$ traite de $concept$. Elle doit être présentée accompagnée d'au moins une autre ressource traitant du même concept. $priorite$ correspond à l'emplacement souhaitable de $ressource$ sur le document par rapport aux autres ressources.
- $effectuer(ressource, concept, niveau)$ Cette action consiste à effectuer une ressource de type **exercice**. $niveau$ est utilisé pour caractériser le niveau de réussite de l'utilisateur. $niveau$ reste donc indéterminé quand une règle rend désirable une action $effectuer$, et est complété par l'interface une fois l'exercice effectué. Ce niveau est ensuite traité par les observateurs pour mettre à jour le profil de l'utilisateur.

– $proposer(concept)$ Cette action consiste à proposer un lien vers une page traitant de $concept$ sur la page courante, si l'affichage de celle-ci n'est pas le fait d'une action de type $lireSeul$.

Nous voyons ici que des définitions adéquates d'actions permettent de fournir des formes variées d'adaptation à l'utilisateur. Par exemple, dans une situation donnée, s'il est désirable de $lire$ une définition, de $lire$ une explication, et de $composer$ un exemple sur un même concept, l'interface va utiliser ces trois assertions de type $good$ pour proposer à l'utilisateur un lien, annoté positivement, vers un document composé des trois ressources. L'interface verra, par exemple, parmi les résultats des règles, les trois éléments suivants :

$good(lire(definition\_tuple,\ tuple,\ haute))$

$good(lire(explication\_tuple,\ tuple,\ moyenne))$

$good(composer(exemple\_tuple,\ tuple,\ basse))$

Il proposera un lien, annoté en vert, vers un document qui présentera successivement la définition d'un tuple, une explication, puis un exemple. Si on a, de plus, un résultat de règle de la forme $good(proposer(schema_relation))$, alors on rajoutera un lien vers une page traitant du concept de schéma de relation après les 3 éléments sus-cités.

Nous complétons les définitions d'action ci-dessus en les typant, afin de pouvoir réutiliser le second système de déduction décrit en 5.4.6. Ainsi, nous avons les trois prédicats suivants :

$type(lire, lecture)$

$type(lireSeul, lecture)$

$type(composer, lecture)$

$type(effectuer, exercice)$

La logique situationnelle est donc suffisamment générique dans notre contexte pour permettre de définir des actions très diverses, dont l'interprétation par une interface adéquate permet de proposer de nombreuses formes d'adaptation.

### 6.4.3 Observateurs de la logique situationnelle

Dans cette partie, nous définissons les observateurs de la logique situationnelle, c'est-à-dire la façon dont sont mises à jour les connaissances de l'utilisateur. Nous appliquons la stratégie suivante : quand l'utilisateur prend connaissance d'un document traitant d'un concept donné, deux cas sont possibles :

– Soit le niveau de connaissance de l'utilisateur est inférieur à "moyen", et alors il passe à "moyen" ;

– Soit le niveau de connaissance de l'utilisateur est supérieur à "moyen", et alors il demeure inchangé.

Quand l'utilisateur fait un exercice, son niveau est capturé par l'interface et renvoyé dans l'action $effectuer$ utilisée. Alors, le niveau de connaissance de l'utilisateur est mis à jour en utilisant le niveau capturé.

Voici les règles que nous avons créé pour capturer cette stratégie :

$holds(degre(user,\ concept,\ moyen),\ do(lire(\_,\ concept, \_), Situation)) \Leftarrow$
$holds(degre(user,\ concept,\ DG), Situation) \wedge DG < moyen$
Cette règle capture le fait que la lecture (seule ou composée) d'un document traitant d'un concept dont la connaissance est inférieure à "moyen" entraîne la mise à jour du niveau de l'utilisateur à "moyen".

$holds(degre(user,\ concept,\ moyen),\ do(lireSeul(\_,\ concept), Situation)) \Leftarrow$
$holds(degre(user,\ concept,\ DG), Situation) \wedge DG < moyen$
Cette règle capture le fait que la lecture (seule) d'un document traitant d'un concept dont la connaissance est inférieure à "moyen" entraîne la mise à jour du niveau de l'utilisateur à "moyen".

$holds(degre(user,\ concept,\ DG),\ do(composer(\_,\ concept, \_), Situation)) \Leftarrow$
$holds(degre(user,\ concept,\ DG), Situation)$
Cette règle capture le fait que la lecture composée d'un document traitant d'un concept ne modifie jamais le niveau de connaissance de l'utilisateur. En effet, un document composé est toujours accompagné d'un autre document, correspondant à une action de type $lire$. C'est cette action de type $lire$ qui rendra effective, si nécessaire, la mise à jour des connaissances de l'utilisateur.

$holds(degre(user,\ concept,\ DG),\ do(lire(\_,\ concept, \_), Situation)) \Leftarrow$
$holds(degre(user,\ concept,\ DG), Situation) \wedge DG > moyen$
Cette règle capture le fait que la lecture (seule ou composée) d'un document traitant d'un concept dont la connaissance est supérieure à "moyen" n'entraîne aucune mise à jour du niveau de l'utilisateur.

$holds(degre(user,\ concept,\ DG),\ do(lireSeul(\_,\ concept), Situation)) \Leftarrow$
$holds(degre(user,\ concept,\ DG), Situation) \wedge DG > moyen$
Cette règle capture le fait que la lecture (seule) d'un document traitant d'un concept dont la connaissance est supérieure à "moyen" n'entraîne aucune mise à jour du niveau de l'utilisateur.

$holds(degre(user,\ concept1,\ DG),\ do(lire(\_,\ concept2, \_), Situation)) \Leftarrow$
$holds(degre(user,\ concept,\ DG), Situation) \wedge concept1 \neq concept2$
Cette règle capture le fait que la lecture (seule ou composée) d'un document traitant d'un concept n'entraîne aucune mise à jour sur les connaissances que l'utilisateur vis-à-vis d'autres concepts.

$holds(degre(user,\ concept1,\ DG),\ do(lireSeul(\_,\ concept2), Situation)) \Leftarrow$
$holds(degre(user,\ concept,\ DG), Situation) \wedge concept1 \neq concept2$
Cette règle capture le fait que la lecture (seule) d'un document traitant d'un concept n'entraîne aucune mise à jour sur les connaissances que l'utilisateur vis-à-vis d'autres concepts.

$holds(degre(user,\ concept1,\ DG),\ do(composer(\_,\ concept2, \_), Situation)) \Leftarrow$
$holds(degre(user,\ concept,\ DG), Situation) \wedge concept1 \neq concept2$

Cette règle capture le fait que la lecture composée d'un document traitant d'un concept n'entraîne aucune mise à jour sur les connaissances que l'utilisateur vis-à-vis d'autres concepts.

$holds(degre(user,\ concept,\ niveau),\ do(effectuer(\_,\ concept,\ niveau), Situation))$
Cette règle capture le fait que les niveaux obtenus en exercice sont répercutés dans le profil de l'utilisateur, quel que soit l'état précédent (supposé ou avéré) de ses connaissances.

$holds(degre(user,\ \_,\ "tresbas"),\ situation\_initiale)$
Cette règle capture le fait que l'on suppose le niveau de connaissance de l'utilisateur sur n'importe quel concept comme "très bas" dans la situation initiale, c'est-à-dire avant qu'il n'aborde le cours en ligne.

Grâce à ces règles, le profil de l'utilisateur est maintenu à jour. Notons que ces règles sont complémentaires de la stratégie de déduction choisie, à savoir la seconde proposition formulée dans la partie 5.4.6. Ainsi, l'utilisateur pourra relire des cours si son niveau en exercice est trop bas, mais n'aura pas à refaire d'exercice s'il les a bien réussi (il faut un niveau moyen pour accéder aux exercices).

### 6.4.4 Règles d'adaptation

Dans cette partie, nous allons donner différentes règles correspondant à différentes stratégies de parcours du domaine. Il s'agit des règles de base de notre système de règle. Certaines règles seront donc en contradiction avec d'autres règles : les méta-règles éviteront que de telles règles puissent faire partie d'une même stratégie de parcours.

Tout d'abord, nous définissons quelques règles de possibilité que l'on peut utiliser dans tous les cas, sans stratégie particulière. Ces règles permettent d'éviter de se retrouver sans possibilité de parcours.

$RBase1\ :\ \forall R \in \{R/type(R, explication)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but)\ \Longrightarrow\ poss(lire(R, Concept, moyenne))$

Il est possible de lire une explication si elle traite d'un concept qui mène au but.

$RBase2\ :\ \forall R \in \{R/type(R, defintion)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but)\ \Longrightarrow\ poss(lire(R, Concept, moyenne))$

Il est possible de lire une définition si elle traite d'un concept qui mène au but.

$RBase3\ :\ \forall R \in \{R/type(R, exemple)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but)\ \Longrightarrow\ poss(composer(R, Concept, moyenne))$

Il est possible de lire un exemple s'il traite d'un concept qui mène au but.

$RBase4\ :\ \forall R \in \{R/type(R, exercice)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but)\ \Longrightarrow\ poss(effectuer(R, Concept, moyenne))$

Il est possible d'effectuer un exercice s'il traite d'un concept qui mène au but.

Ces règles rendent possible de lire des explications ou des définitions, de composer des exemples ou d'effectuer des exercices, si le concept auquel ils correspondent mène au but fixé.

Nous définissons ci-dessous des règles qui permettent d'effectuer un parcours en largeur du graphe des concepts.

$RLargeur1 : \forall R \in \{R/type(R, definition)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$\neg hierarchie(Concept2, Concept) \implies good(lire(R, Concept, moyenne))$

Il est désirable de lire une définition si elle traite d'un concept ne descendant pas directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors moyenne.

$RLargeur1b : \forall R \in \{R/type(R, definition)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$\neg hierarchie(Concept2, Concept) \implies good(lire(R, Concept, haute))$

Il est désirable de lire une définition si elle traite d'un concept ne descendant pas directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors haute.

$RLargeur2 : \forall R \in \{R/type(R, definition)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$\neg hierarchie(Concept2, Concept) \implies good(lireSeul(R, Concept))$

Il est désirable de lire une définition seule si elle traite d'un concept ne descendant pas directement du concept précédement traité vis-à-vis de la relation hiérarchique.

$RLargeur3 : \forall R \in \{R/type(R, explication)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$\neg hierarchie(Concept2, Concept) \implies good(lire(R, Concept, moyenne))$

Il est désirable de lire une explication si elle traite d'un concept ne descendant pas directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de l'explication est alors moyenne.

$RLargeur3b : \forall R \in \{R/type(R, explication)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$\neg hierarchie(Concept2, Concept) \implies good(lire(R, Concept, haute))$

Il est désirable de lire une explication si elle traite d'un concept ne descendant pas directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de l'explication est alors haute.

$RLargeur4 \ : \ \forall R \in \{R/type(R, explication)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$\neg hierarchie(Concept2, Concept) \implies good(lireSeul(R, Concept))$

Il est désirable de lire une explication seule si elle traite d'un concept ne descendant pas directement du concept précédement traité vis-à-vis de la relation hiérarchique.

$RLargeur5 \ : \ \forall R \in \{R/type(R, exemple)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$\neg hierarchie(Concept2, Concept) \implies good(composer(R, Concept, basse))$

Il est désirable de composer un exemple s'il traite d'un concept ne descendant pas directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de l'exemple est alors basse.

$RLargeur5b \ : \ \forall R \in \{R/type(R, exemple)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$\neg hierarchie(Concept2, Concept) \wedge metadata(R, niveau, facile)$
$\implies good(composer(R, Concept, haute))$

Il est désirable de composer un exemple s'il traite d'un concept ne descendant pas directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de l'exemple est alors haute.

Nous avons défini quelques règles de parcours en largeur, qui donnent plusieurs possibilité de lecture des explications ou des définitions. Ces types de lecture seront sélectionnés en fonction du type d'utilisateur.

Nous allons maintenant donner les règles de parcours en profondeur :
$RProf1 \ : \ \forall R \in \{R/type(R, definition)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$hierarchie(Concept2, Concept) \implies good(lire(R, Concept, moyenne))$

Il est désirable de lire une définition si elle traite d'un concept descendant directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors moyenne.

$RProf1b \ : \ \forall R \in \{R/type(R, definition)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$hierarchie(Concept2, Concept) \implies good(lire(R, Concept, haute))$

Il est désirable de lire une définition si elle traite d'un concept descendant directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors haute.

$RProf2 \ : \ \forall R \in \{R/type(R, definition)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$hierarchie(Concept2, Concept) \implies good(lireSeul(R, Concept))$

Il est désirable de lire une définition si elle traite d'un concept descendant directement du concept précédement traité vis-à-vis de la relation hiérarchique.

$RProf3 \ : \ \forall R \in \{R/type(R, explication)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$hierarchie(Concept2, Concept) \implies good(lire(R, Concept, moyenne))$

Il est désirable de lire une explication seule si elle traite d'un concept descendant directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors moyenne.

$RProf3b \ : \ \forall R \in \{R/type(R, explication)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$hierarchie(Concept2, Concept) \implies good(lire(R, Concept, haute))$

Il est désirable de lire une explication si elle traite d'un concept descendant directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors haute.

$RProf4 \ : \ \forall R \in \{R/type(R, explication)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$hierarchie(Concept2, Concept) \implies good(lireSeul(R, Concept))$

Il est désirable de lire une explication seule si elle traite d'un concept descendant directement du concept précédement traité vis-à-vis de la relation hiérarchique.

$RProf5 \ : \ \forall R \in \{R/type(R, exemple)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$hierarchie(Concept2, Concept) \implies good(composer(R, Concept, basse))$

Il est désirable de composer un exemple s'il traite d'un concept descendant directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors basse.

$RProf5b \ : \ \forall R \in \{R/type(R, exemple)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, Concept2) \wedge$
$hierarchie(Concept2, Concept) \wedge metadata(R, niveau, facile)$
$\implies good(composer(R, Concept, haute))$

Il est désirable de composer un exemple s'il traite d'un concept descendant directement du concept précédement traité vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors haute.

Ensuite, nous présentons quelques règles qui permettent de n'aborder que les notions plus haut niveau, sans rentrer dans les détails du cours.

$RHaut1 \ : \ \forall R \in \{R/type(R, definition)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge$
$prerequis^2(bd, Concept) \implies good(lire(R, Concept, moyenne))$

Il est désirable de lire une définition si elle traite d'un concept descendant directement ou au deuxième degré du concept bd (base de données) vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors moyenne.

$RHaut1b \ : \ \forall R \in \{R/type(R, definition)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge$
$prerequis^2(bd, Concept) \implies good(lire(R, Concept, haute))$

Il est désirable de lire une définition si elle traite d'un concept descendant directement ou au deuxième degré du concept bd (base de données) vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors haute.

$RHaut2 \ : \ \forall R \in \{R/type(R, definition)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge$
$prerequis^2(bd, Concept) \implies good(lireSeul(R, Concept))$

Il est désirable de lire une définition seule si elle traite d'un concept descendant directement ou au deuxième degré du concept bd (base de données) vis-à-vis de la relation hiérarchique.

$RHaut3 \ : \ \forall R \in \{R/type(R, explication)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge$
$prerequis^2(bd, Concept) \implies good(lire(R, Concept, moyenne))$

Il est désirable de lire une explication si elle traite d'un concept descendant directement ou au deuxième degré du concept bd (base de données) vis-à-vis de la relation hiérarchique. La priorité d'affichage de l'explication est alors moyenne.

$RHaut3b \ : \ \forall R \in \{R/type(R, explication)\}, \ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge$
$prerequis^2(bd, Concept) \implies good(lire(R, Concept, haute))$

Il est désirable de lire une explication si elle traite d'un concept descendant directement ou au deuxième degré du concept bd (base de données) vis-à-vis de la relation hiérarchique. La priorité d'affichage de l'explication est alors haute.

$RHaut4 : \forall R \in \{R/type(R, explication)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge$
$prerequis^2(bd, Concept) \implies good(lireSeul(R, Concept))$

Il est désirable de lire une explication seule si elle traite d'un concept descendant directement ou au deuxième degré du concept bd (base de données) vis-à-vis de la relation hiérarchique.

$RHaut5 : \forall R \in \{R/type(R, exemple)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge$
$prerequis^2(bd, Concept) \implies good(lireSeul(R, Concept, basse))$

Il est désirable de composer un exemple s'il traite d'un concept descendant directement ou au deuxième degré du concept bd (base de données) vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors basse.

$RHaut5b : \forall R \in \{R/type(R, exemple)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge metadata(R, niveau, facile) \wedge$
$prerequis^2(bd, Concept) \implies good(lireSeul(R, Concept, haute))$

Il est désirable de composer un exemple s'il traite d'un concept descendant directement ou au deuxième degré du concept bd (base de données) vis-à-vis de la relation hiérarchique. La priorité d'affichage de la définition est alors haute.

Enfin, nous donnons des règles qui rendent, pour certains utilisateurs, les exercices désirables :
$RExo1 : \forall R \in \{R/type(R, exercice)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \wedge abstraction(documentCourant, concept)$
$\implies good(lireSeul(R, Concept, \_))$

Il est désirable d'effectuer un exercice s'il traite du concept courant et s'il mène au but.

$RExo2 : \forall R \in \{R/type(R, exercice)\},\ abstraction(R, Concept) \wedge$
$prerequis^*(Concept, but) \implies good(lireSeul(R, Concept, \_))$

Il est désirable d'effectuer un exercice s'il mène au but.

$RProp : \forall Concept \in \{C/type(C, Concept)\},\ abstraction(documentCourant, Concept2)$
$\wedge hierarchie(Concept2, Concept) \wedge prerequis^*(Concept, but)$
$\implies good(Proposer(Concept))$

Il est désirable de proposer un lien vers un sous-concept du concept courant, et ce s'il mène au but.

Nous venons de définir 30 règles de parcours, correspondant à différents morceaux de stratégies d'adaptation possibles. Certaines sont en contradiction avec d'autres, alors que certaines règles sont complémentaires. Nous allons détailler, dans la section suivante, les critères de déclenchement

associés à ces 31 règles. Notons que sur ces 31 règles, qui permettent de fournir toutes sortes d'adaptation, le nombre maximal de prémisses rencontré est de 4. Ces règles restent donc assez simple à écrire, à comprendre et à maintenir, ce qui était un de nos objectifs. Le fait que ces règles traitent uniquement la forme du domaine les rend également plus lisibles.

### 6.4.5 Association règles/stéréotypes

Dans cette partie, nous allons détailler les associations entre règles et critères (stéréotypes de l'utilisateur), c'est-à-dire décrire la fonction $\delta$ définie dans le chapitre 5. Nous faisons remarquer que la première partie de la règle (Haut, Largeur, Prof, Base, Exo) dénote de la stratégie de parcours des concepts : en largeur (Largeur), en profondeur (Prof), sans descendre trop bas dans l'arborescence des concepts (Haut), en faisant des exercices (Exo), sans stratégie particulière (Base). Pour Haut, Largeur et Prof, les règles 1, 1b et 2 traitent des définitions, 3, 3b et 4 des explications, 5 et 5b des exemples. Les règles terminant par b donnent une priorité haute aux ressources traitées. Ces éléments doivent faciliter la lecture du tableau ci-dessous, qui associe chaque règle aux ensembles de critères nécessaires pour que celles si soient prises en compte.

| **Règle** | **Critères** |
|---|---|
| RBase1 | $\emptyset$ |
| RBase2 | $\emptyset$ |
| RBase3 | $\emptyset$ |
| RBase4 | $\emptyset$ |
| RLargeur1 | **Detail-Largeur, 3A-Info** |
| RLargeur1b | **Detail-Largeur, 3A-Info, Abstr-Positif, Vit-Lent** |
| RLargeur2 | **Detail-Largeur, 3A-Info, Abstr-Positif, Vit-Rapide** |
| RLargeur3 | **Detail-Largeur, 3A-Info, Vit-Lent** |
| RLargeur3b | **Detail-Largeur, 3A-Info, Abstr-Neutre,Vit-Lent** |
| RLargeur4 | **Detail-Largeur, 3A-Info, Abstr-Neutre, Vit-Rapide** |
| RLargeur5 | **Detail-Largeur, 3A-Info, Abstr-Neutre** |
| RLargeur5b | **Detail-Largeur, 3A-Info, Abstr-Negatif** |

| Règle | Critères |
|---|---|
| RProf1 | **Detail-Prof, 3A-Info** |
| RProf1b | **Detail-Prof, 3A-Info, Abstr-Positif, Vit-Lent** |
| RProf2 | **Detail-Prof, 3A-Info, Abstr-Positif, Vit-Rapide** |
| RProf3 | **Detail-Prof, 3A-Info, Vit-Lent** |
| RProf3b | **Detail-Prof, 3A-Info, Abstr-Neutre,Vit-Lent** |
| RProf4 | **Detail-Prof, 3A-Info, Abstr-Neutre, Vit-Rapide** |
| RProf5 | **Detail-Prof, 3A-Info, Abstr-Neutre** |
| RProf5b | **Detail-Prof, 3A-Info, Abstr-Negatif** |
| RHaut1 | **3A-Autre** |
| RHaut1b | **3A-Autre, Abstr-Positif, Vit-Lent** |
| RHaut2 | **3A-Autre, Abstr-Positif, Vit-Rapide** |
| RHaut3 | **3A-Autre, Vit-Lent** |
| RHaut3b | **3A-Autre, Abstr-Neutre,Vit-Lent** |
| RHaut4 | **3A-Autre, Abstr-Neutre, Vit-Rapide** |
| RHaut5 | **3A-Autre, Abstr-Neutre** |
| RHaut5b | **3A-Autre, Abstr-Negatif** |
| RExo1 | **Detail-Prof, Vit-Lent, BesoinEx** |
| RExo2 | **Vit-Lent, BesoinEx** |
| RProp | **Vit-Lent** |

### 6.4.6 Méta-règles

Dans cette partie, nous détaillons les méta-règles que nous utilisons. Pour ce faire, nous procédons en 3 étapes :

1. Y a-t-il des dépendances entre stéréotypes ? Si oui, lesquels requièrent quels autres stéréotypes ? Lesquels sont incompatibles avec d'autres, et, dans ce cas, quel critère faut-il privilégier ?
2. Quels sont les critères opposés (**Vit-Rapide** contre **Vit-Lent**) ? Lequels faut-il privilégier en cas d'ambiguïté ? Ces méta-règles sont particulièrement indispensables dans la mesure où il est possible que certains critères soient indéterminés.
3. Quelle priorité faut-il donner aux règles pour désambigüer des conclusions contradictoires ?

Ainsi, nous obtenons les méta règles suivantes :

**MRDepend1** $\gamma(Vit - Lent) \supset \gamma(BesoinEx)$

**MRDepend2** $\gamma(Abstr - Negatif) \supset \gamma(BesoinEx)$

Les utilisateurs lents, ainsi que les utilisateurs aux faibles facultés d'abstraction, ont besoin d'exercices.

**MROpp1** $\overline{\gamma(Vit - Lent)\ \ \gamma(Vit - Rapide)}$

**MROpp2** $\overline{\gamma(Abstr - Positif)\ \ \gamma(Abstr - Negatif)}$

**MROpp3** $\overline{\gamma(Abstr - Neutre)\ \ \gamma(Abstr - Negatif)}$

**MROpp4** $\overline{\gamma(Abstr - Positif)\ \ \gamma(Abstr - Neutre)}$

**MROpp5** $\overline{\gamma(Detail - Prof)\ \ \gamma(Detail - Largeur)}$

**MROpp6** $\overline{\gamma(BesoinEx)\ \ \gamma(NonBesoinEx)}$

Les différentes valeurs possibles des stéréotypes sont deux à deux incompatibles pour un même stéréotype.

**MRDesAmb1** $\gamma(Vit - Lent) > \gamma(Vit - Rapide)$

**MRDesAmb2** $\gamma(Abstr - Neutre) > \gamma(Abstr - Positif)$

**MRDesAmb3** $\gamma(Abstr - Neutre) > \gamma(Abstr - Positif)$

**MRDesAmb4** $\gamma(Abstr - Negatif) > \gamma(Abstr - Positif)$

**MRDesAmb5** $\gamma(Detail - Prof) > \gamma(Detail - Largeur)$

**MRDesAmb6** $\gamma(BesoinEx) > \gamma(NonBesoinEx)$

Nous faisons des choix quand des catégories de règles incompatibles se retrouvent déclenchables simultanément.

**MRPriorite1** $RExo1 \prec RExo2$

**MRPriorite2** $RExo1 \prec \gamma(\emptyset)$

**MRPriorite4** $RLargeur1 \prec RLargeur1b$

**MRPriorite5** $RLargeur1 \prec RLargeur2$

**MRPriorite6** $RProf1 \prec RProf1b$

**MRPriorite7** $RProf1 \prec RProf2$

**MRPriorite8** $RHaut1 \prec RHaut1b$

**MRPriorite9** $RHaut1 \prec RHaut2$

**MRPriorite10** $RLargeur3 \prec RLargeur3b$

**MRPriorite11** $RLargeur3 \prec RLargeur4$

**MRPriorite12** $RProf3 \prec RProf3b$

**MRPriorite13** $RProf3 \prec RProf4$

**MRPriorite14** $RHaut3 \prec RHaut3b$

**MRPriorite15** $RHaut3 \prec RHaut4$

Les différents critères peuvent entraîner la présence de deux règles concurrentes, par exemple une règle qui permet de lire une ressource, et une autre qui permet de lire cette même ressource seule. Nous donnons des priorités différentes à ces règles. Ainsi, selon les options de présentation du logiciel, l'utilisateur pourra se voir proposer des documents alternatifs, ou uniquement les meilleurs documents.

Pour définir les méta-règles, qui permettent la sélection de stratégie, nous avons défini 29 méta-règles. Nous avons également défini 31 ensembles de critères de déclenchement, pour les 30 règles

de base. Nous sommes donc parvenus à définir implicitement les 216 stratégies de règles nécessaires à notre système en définissant 60 éléments seulement, soit environ 36% du nombre d'éléments qu'il nous aurait fallu définir manuellement si nous n'avions pas disposé de nos outils.

De plus, la définition de nos 60 éléments repose sur notre connaissance de l'hypermédia adaptatif à construire : quels sont les critères, pour quels critères chaque règle a-t-elle été conçue, quelles restrictions doit-on appliquer aux différentes catégories de règles ? Une définition manuelle de 216 stratégie nécessite de se baser uniquement sur les identifiants des règles, à ajouter pour chaque combinaison possible de critères sur l'utilisateur. Cette approche est d'autant plus susceptible de conduire à des erreurs qu'elle ne dispose pas, contrairement à la nôtre, de mécanismes de vérifications qui nous permettent de corriger les erreurs de conception.

## 6.5 Exemples d'applications

Dans cette partie, nous explicitons quelques stratégies mises en oeuvre pour différentes catégories d'utilisateurs, dans notre domaine.

**Exemple 1**

*Stéréotypes de l'utilisateur*
Vitesse d'apprentissage : Lent
Capacité d'abstraction : Indéterminée
Notion du détail : Profondeur d'abord
Niveau d'objectif : spécialité informatique
Besoin d'exercice : oui
*Règles déclenchables*
En appliquant la fonction d'association règles/critères, on obtient la pré-sélection de règles suivante : RBase1, RBase2, RBase3, Rbase4, RProf1, RProf1b, RProf3, RProf3b, RProf5, RProf5b, RExo1, RExo2, RProp
*Règles sélectionnées*
Par application des méta-règles **MROpp2**, **MROpp3**, **MROpp4**, **MRDesAmb2**, **MRDesAmb3** et **MRDesAmb4**, portant sur l'incompatibilité des différents niveaux d'abstraction et les préférences établies entre ces niveaux, les règles sélectionnées après application des méta-règles sont :
RBase1, RBase2, RBase3, Rbase4, RProf1, RProf3, RProf3b, RProf5, RExo1, RExo2, RProp
*Possibilités offertes pour le parcours de cet utilisateur*

Notre utilisateur préfère les parcours en profondeur dans le graphe des concepts. Les définitions et explications lui sont proposée avec une priorité moyenne et les exemples avec une priorité basse. Les exercices lui sont proposés, sachant que ceux liés au concept en train d'être appris sont prioritaires.

Ainsi, quand cet utilisateur aura abordé le concept de modèle relationnel, il abordera la notion de bases de données relationnelles, puis d'opérateurs. Ensuite, comme *opérateurs* a des sous-concepts, cet utilisateur les abordera : opérations ensemblistes, puis union, intersection et différence relationnelle, avant d'aborder les opérations spécifiques.

Les documents proposés contiennent d'abord les définitions et explications, puis les exemples. À chaque fois qu'un exercice est disponible pour le concept en cours, il est proposé en priorité.

L'utilisateur verra des liens vers les sous-concepts qui mènent au but en bas de chaque page.

**Exemple 2**

*Stéréotypes de l'utilisateur*

Vitesse d'apprentissage : Rapide

Capacité d'abstraction : abstrait

Notion du détail : indéterminée

Niveau d'objectif : spécialité informatique

Besoin d'exercice : non

*Règles déclenchables*

En appliquant la fonction d'association règles/critères, on obtient la pré-sélection de règles suivante :

RBase1, RBase2, RBase3, Rbase4, RProf1, RProf2, RLargeur1, RLargeur2

*Règles sélectionnées*

Par application des méta-règles **MROpp5** et **MRDesAmb5**, portant sur l'incompatibilité des différentes notions du détail et les préférences établies entre ces niveaux, les règles sélectionnées après application des méta-règles sont :

RBase1, RBase2, RBase3, Rbase4, RProf1, RProf2

*Possibilités offertes pour le parcours de cet utilisateur*

Notre utilisateur n'a pas de préférence entre les parcours en largeur et en profondeur. Par défaut, par soucis de simplicité, ce sont les parcours en profondeur qui sont privilégiés. Seules les définitions des concepts lui sont proposées, puisqu'il est à la fois rapide et abstrait.

Ainsi, quand cet utilisateur aura abordé le concept de modèle relationnel, les concepts seront abordés dans le même ordre que dans l'exemple 1.

Pourtant, les documents proposés contiennent uniquement les définitions des concepts. L'utilisateur ne fait pas d'exercices.

Cet utilisateur ne se verra pas proposé de liens en bas de page.

**Exemple 3**

*Stéréotypes de l'utilisateur*

Vitesse d'apprentissage : indéterminée

Capacité d'abstraction : neutre

Notion du détail : largeur d'abord

Niveau d'objectif : autre spécialité

Besoin d'exercice : indéterminé

*Règles déclenchables*

En appliquant la fonction d'association règles/critères, on obtient la pré-sélection de règles suivante : RBase1, RBase2, RBase3, Rbase4, RHaut1, RHaut3, RHaut3b, RHaut4, RHaut5, RExo1, RExo2, RProp

*Règles sélectionnées*

Par application des méta-règles **MROpp1**, **MRDesAmb1** portant sur l'incompatibilité des différentes vitesses d'apprentissage et les préférences établies entre ces niveaux, les règles sélectionnées après application des méta-règles sont :

RBase1, RBase2, RBase3, Rbase4, RHaut1, RHaut3, RHaut3b, RHaut5, RExo1, RExo2, RProp

Par application de **MRPriorite14**, la règle RHaut3 fournira des actions prioritaires sur celles de RHaut3b. Ainsi, il sera préférable de donner un priorité moyenne et non pas haute aux explications.

*Possibilités offertes pour le parcours de cet utilisateur*

Notre utilisateur préfère les parcours en largeur dans le graphe des concepts. Il ne fait pas la spécialité informatique, donc ne parcourt que deux niveaux de profondeur à partir de la racine. Il n'est pas explicitement contre les exercices, donc s'en verra proposer, en priorité s'ils sont rattachés au concept traité par le document courant.

Cet utilisateur étudiera d'abord le concept de bases de données, puis le concept de modèle relationnel, puis celui de système de gestion de bases de données et enfin celui de conception de bases de données. Ensuite seulement, il abordera les concepts du niveau en dessous dans la hiérarchie conceptuelle de notre domaine.

Les documents proposés contiennent d'abord les définitions et explications, puis les exemples. À chaque fois qu'un exercice est disponible pour le concept en cours, il est proposé en priorité.

L'utilisateur verra des liens vers les sous-concepts qui mènent au but en bas de chaque page.

Nous venons de donner quelques exemples d'utilisation de notre système. Il est bien évident que de très nombreuses autres possibilités sont envisageables, pour chaque profil d'utilisateur différent. Nous voyons que nos cinq stéréotypes permettent de décliner de très nombreuses possibilités d'adaptation pour les utilisateurs de notre système.

## 6.6 Conclusions

Dans ce chapitre, nous avons présenté les éléments essentiels d'un système d'hypermédia adaptatif pour l'apprentissage d'un cours sur les bases de données, basé sur notre modèle d'adaptation et nos modèles de représentation des données pour le e-learning.

Cet exemple a mis en avant les capacités de la logique situationnelle pour les systèmes d'hypermédias adaptatifs : la définition des actions permet de créer de très nombreuses formes d'adaptation. Ces formes d'adaptation sont définies en fonction de la sémantique donnée aux actions, ce qui permet de proposer différentes techniques d'adaptation pour chaque catégorie présentée dans [11]. Ainsi, nous avons mis en place un système qui propose aussi bien de l'adaptation de liens que de la composition de documents, avec ordonnancement des fragments composés.

Nous avons également montré à quel point l'utilisation de méta-règles permet de créer de nombreuses stratégies de parcours sans pour autant multiplier les saisies à effectuer, et, par conséquent, en réduisant les risques d'erreurs.

Nous avons également montré l'intérêt qui existe à séparer le niveau conceptuel du niveau des ressources. Le domaine est ainsi clairement identifié grâce à ses concepts, et ce quel que soit les ressources disponibles. De plus, nos règles de base proposent un parcours au niveau conceptuel, beaucoup plus simple et consistant à décrire qu'un parcours au niveau des ressources, qui peuvent être nombreuses à aborder un même concept.

Nous avons d'ores et déjà implémenté notre moteur d'inférence, capable de prendre en compte correctement règles et méta-règles. Nous travaillons, au moment où nous écrivons ces lignes, à la mise en place d'une interface graphique permettant de rendre entièrement utilisable notre prototype.

# Chapitre 7

# Conclusion

Dans cette thèse, nous avons proposé un modèle générique pour la conception d'hypermédias adaptatifs. Ce modèle repose sur deux modèles génériques de données statiques, décrits en UML - et permettant la création de modèles spécifiques pour les besoins de chaque domaine d'application - et d'un modèle d'adaptation entièrement décrit dans le calcul des prédicats du premier ordre, formalisme particulièrement adéquat, bien fondé et bien connu. Ce modèle permet de créer des hypermédias adaptatifs dans des domaines très variés, de manière essentiellement déclarative. Il généralise des notions retrouvées dans de nombreux modèles existant, comme AHAM [8]. Il propose également un fondement théorique pour l'ensemble des systèmes d'hypermédias adaptatifs.

Afin de fournir de l'adaptation aux utilisateurs, notre modèle propose les éléments suivant :

- Un modèle générique de l'utilisateur, qui reprend les données essentielles, toujours présentes dans les modèles de l'utilisateur dans les systèmes d'hypermédias adaptatifs, et permet la création d'un modèle de l'utilisateur spécifique au domaine d'application désirée ;
- Un modèle générique du domaine, fournissant les mêmes éléments que le modèle générique de l'utilisateur, au niveau du domaine de l'application désiré ;
- Une base d'adaptation fondée sur la logique situationnelle, qui permet, comme nous l'avons montré au chapitre 6, de fournir des formes d'adaptation aussi diverses que nécessaire ;
- Un langage de règle très générique, capable de raisonner au niveau des modèles génériques des données statiques, qui permet la sélection et la classification d'actions de la logique situationnelle. Ce modèle utilise des méta-règles pour sélectionner des stratégies de parcours, et, en cela, fournir une forme de méta-adaptation. Il utilise des modèles de raisonnement potentiellement variés pour modifier les critères de connaissances de l'utilisateur pris en compte.

Ce modèle reprend ainsi le découpage classique utilisateur/domaine/adaptation, que l'on trouve notamment dans [11, 8, 52, 33]. Cette thèse explore également un possibilité d'amélioration du guidage de l'utilisateur, en proposant la composition de tronçons de parcours se terminant sur des objectifs intermédiaires. Elle propose enfin des modèles, basé sur les modèles génériques, pour la conception d'hypermédias adaptatifs pour le e-learning.

Un prototype du moteur d'inférence, basé sur la logique situationnelle, a été réalisé durant cette thèse. Il a permis, de manière incrémentale, de montrer le bien-fondé des idées avancées. La logique situationnelle y est apparue comme un bon cadre pour proposer des actions. Le système de règle a été construit au fur et à mesure, sans méta-règles, puis avec des méta-règles, qui ont montré comment les stratégies d'adaptation pouvaient être diversifiées tout en simplifiant la composition des règles de parcours. Une version finale, utilisable à des fins démonstratives, et reposant sur l'exemple donné dans le chapitre 6 est en cours d'implémentation.

Les apports de cette thèse peuvent être catégorisés de la façon suivante :

- Nous avons proposé un modèle beaucoup plus générique que les modèles existant, avec deux conséquences : nous facilitons la création de nouveaux modèles, et nous proposons une théorie pour ces systèmes ;
- Nous proposons un modèle qui s'abstrait des formes d'adaptation potentielles, et n'est donc pas contraint à ne proposer qu'un sous-ensemble de ces techniques ;
- Nous avons proposé un langage de règle entièrement logique qui est capable de raisonner à très haut niveau, indépendamment de la forme précise du domaine ;
- Nous avons proposé une façon de formaliser une méta-adaptation, sans complexifier la forme des règles d'adaptation ;
- Nous avons créé une base de raisonnement formelle, basée sur le calcul des prédicats du premier ordre, là où la plupart des travaux se contentent de décrire cette base de manière informelle avant de l'implémenter directement ;
- Nous avons proposé des améliorations concernant les techniques de guidage direct de l'utilisateur.

Nous discutons ci-dessous des perspectives que présentent nos travaux.

Nous souhaitons tout d'abord finir l'implémentation de notre modèle et tester son efficacité sur des élèves susceptibles de suivre un cours de base de données.

Comme nous venons de l'expliquer, notre modèle constitue une théorisation des hypermédias adaptatifs. Nous souhaitons désormais chercher à montrer que ce modèle est une théorisation de modèles existant connus. Nous avons succinctement montré, dans les chapitres 3 et 4 que nos modèles génériques sont des généralisations des modèles existant. Nous sommes actuellement en contact avec Alexandra Cristea, de l'Université Technique d'Eindhoven, afin de corriger nos modèles d'adaptation respectifs et de montrer que les langages de règles proposés dans [65] sont une représentation de notre langage générique. Nous donnerions ainsi à nos travaux une envergure plus importante en matière de théorie des hypermédias adaptatifs.

Afin de faciliter la création de systèmes réutilisant notre modèle, il serait souhaitable de fournir un outil-auteur, capable de rendre l'opération de conception du système beaucoup plus intuitive. Nous avons d'ores et déjà réfléchi aux possibilités que nous pourrions offrir à cet effet. Au niveau du langage de règle, la définition de la fonction d'association règles/critères, ainsi que la création

des méta-règles, peuvent être réalisés à l'aide de diagrammes, de manière entièrement graphique. Il suffit de choisir, d'une part, quels critères relier avec quelles règles, et, d'autre part, quels types de méta-règles utiliser pour relier deux ensembles de règles. Concernant l'écriture des règles, et sachant que la syntaxe de celles-ci est très encadrée, il serait possible de proposer un éditeur de règles vérifiant la syntaxe et l'existence des relations, types, méta-données et actions décrits dans les règles. La création de modèles spécifiques pour l'utilisateur et le domaine pourrait reposer sur un éditeur UML modifié.

Nous avons vu qu'il est possible de créer différents modèles pour le domaine et l'utilisateur. Néanmoins, toutes les personnes susceptibles de réutiliser notre modèle ne voudront pas redéfinir nos modèles spécialisés. Elles les réutiliseront donc, et risquent, de ce fait, d'être confrontées à un problème d'importation de données. Il serait intéressant d'étudier de quelle manière on pourrait faciliter, voire automatiser, l'importation de données dans un système d'hypermédia adaptatif, en s'interrogeant notamment sur la nécessite ou l'absence de nécessité de réutilisation de certaines données très précises, peu utiles pour fournir de l'adaptation.

Nous avons proposé deux systèmes de déduction sur nos règles et nos méta-règles dans cette thèse. Ces deux systèmes correspondent à des cas d'utilisation différents. Bien que nous ayons ainsi décrit une méthode pour créer un système de preuve formel, nous souhaiterions aller plus loin dans cette démarche, et ce de deux manières différentes. La première consiste à définir plusieurs systèmes de preuves formels, décrits de manière informelle, que l'on pourrait proposer au créateur d'hypermédia adaptatif. Il pourrait ainsi choisir sur quels critères un utilisateur peut faire un exercice, quel niveau de connaissance des pré-requis ce dernier doit avoir pour accéder à une partie de cours etc. La seconde consisterait à proposer une façon de construire au cas par cas ces systèmes de preuve formel : peut-on faire varier les paramètres mis en avant dans ces systèmes, ou permettre leur redéfinition entière ? Peut-on définir un méta-méta-interpréteur capable de générer des méta-interpréteurs pour chaque système formel proposé ?

Enfin, nous souhaitons étudier à travers de plus nombreux cas d'utilisation les possibilités et restrictions qu'imposent nos méta-règles. Nous souhaitons notamment étudier la possibilité de les étendre pour traiter la disjonction entre méta-règles. En effet, toutes les méta-règles actuellement sont en conjonction les unes avec les autres. Il pourrait parfois être préférable d'indiquer qu'une catégorie de règle nécessite une autre catégorie **ou bien** une troisième catégorie. Il nous faudrait alors étudier les conséquences, en terme de rapidité des algorithmes, de telles modifications. Nous pensons que l'ajout d'une disjonction permettrait également de lever une légère "inconsistance" qui demeure, sans nuire aux calculs, dans notre modèle, à savoir l'utilisation combinée de méta-règles ensemblistes et de méta-règles individuelles. L'ajout d'une disjonction pourrait permettre, dans une certaine mesure, de n'avoir à traiter que des relations entre critères.

# Bibliographie


[1] F. Bacchus and F. Kabanza. Using temporal logics to control search in a forward chaining planner. *New Direction in AI Planning pp. 141-153*, 1996.

[2] Ian H. Beaumont. User modelling in the interactive anatomy tutoring system anatom-tutor. *User Model. User-Adapt. Interact.*, 4(1) :21–45, 1994.

[3] Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90 :281–300, 1997.

[4] Rafal Bogacz and Christophe Giraud-Carrier. Learning meta-rules of selection in expert systems. In *Proceedings of the Fourth World Congress on Expert Systems*, pages 576–581. Cognizant Communication Corporation, March 1998.

[5] Blai Bonet and Hector Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2) :5–33, 2001.

[6] Craig Boyle and Antonio O. Encarnacion. Metadoc : An adaptive hypertext reading system. *User Model. User-Adapt. Interact.*, 4(1) :1–19, 1994.

[7] Paul De Bra, Peter Brusilovsky, and Ricardo Conejo, editors. *Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002, Malaga, Spain, May 29-31, 2002, Proceedings*, volume 2347 of *Lecture Notes in Computer Science*. Springer, 2002.

[8] Paul De Bra, Geert-Jan Houben, and Hongjing Wu. AHAM : A dexter-based reference model for adaptive hypermedia. In *UK Conference on Hypertext*, pages 147–156, 1999.

[9] Paul De Bra and Wolfgang Nejdl, editors. *Adaptive Hypermedia and Adaptive Web-Based Systems, Third International Conference, AH 2004, Eindhoven, The Netherlands, August 23-26, 2004, Proceedings*, volume 3137 of *Lecture Notes in Computer Science*. Springer, 2004.

[10] Paul De Bra and Jan-Peter Ruiter. Aha ! adaptive hypermedia for all. In Wendy A. Lawrence-Fowler and Joachim Hasebrook, editors, *WebNet*, pages 262–268. AACE, 2001.

[11] Peter Brusilovsky. Methods and techniques of adaptive hypermedia. *User Model. User-Adapt. Interact.*, 6(2-3) :87–129, 1996.

[12] Peter Brusilovsky, John Eklund, and Elmar Schwarz. Web-based education for all : a tool for development adaptive courseware. *Computer Networks and ISDN Systems*, 30(1–7) :291–300, 1998.

[13] Peter Brusilovsky, Elmar W. Schwarz, and Gerhard Weber. Elm-art : An intelligent tutoring system on world wide web. In *ITS '96 : Proceedings of the Third International Conference on Intelligent Tutoring Systems*, pages 261–269, London, UK, 1996. Springer-Verlag.

[14] Licia Calvi and Alexandra I. Cristea. Towards generic adaptive systems : Analysis of a case study. In Bra et al. [7], pages 79–89.

[15] Alexandra I. Cristea and Lora Aroyo. Adaptive authoring of adaptive educational hypermedia. In Bra et al. [7], pages 122–132.

[16] Alexandra I. Cristea and Licia Calvi. The three layers of adaptation granularity. In Peter Brusilovsky, Albert T. Corbett, and Fiorella de Rosis, editors, *User Modeling*, volume 2702 of *Lecture Notes in Computer Science*, pages 4–14. Springer, 2003.

[17] Randall Davis. Meta-rules : Reasoning about control. *Artificial Intelligence*, 15 :179–22, 1980.

[18] Patricia Seefelder de Assis and Daniel Schwabe. A general meta-model for adaptive hypermedia systems. In Bra and Nejdl [9], pages 433–436.

[19] Patricia Seefelder de Assis and Daniel Schwabe. A semantic meta-model for adaptive hypermedia systems. In Bra and Nejdl [9], pages 360–365.

[20] Patricia Seefelder de Assis, Daniel Schwabe, and Demetrius Arraes Nunes. Ashdm - model-driven adaptation and meta-adaptation. In Vincent P. Wade, Helen Ashman, and Barry Smyth, editors, *AH*, volume 4018 of *Lecture Notes in Computer Science*, pages 213–222. Springer, 2006.

[21] Peter Dolog, Nicola Henze, Wolfgang Nejdl, and Michael Sintek. Towards the adaptive semantic web. In François Bry, Nicola Henze, and Jan Maluszynski, editors, *PPSWR*, volume 2901 of *Lecture Notes in Computer Science*, pages 51–68. Springer, 2003.

[22] Peter Dolog, Nicola Henze, Wolfgang Nejdl, and Michael Sintek. The personal reader : Personalizing and enriching learning resources using semantic web technologies.. In Bra and Nejdl [9], pages 85–94.

[23] Freddy Duitama, Bruno Defude, Amel Bouzeghoub, and Claire Carpentier. A framework for the the generation of adaptive courses based on semantic metadata. 2005.

[24] Anthony Finkelstein et al. Ubiquitous web application development - a framework for understanding. In *The Sixth World Multi-Conference on Systemics, Cybernetics and Informatics (SCI2002)*, 2002.

[25] Hector J. Levesque et al. Golog : A logic programming language for dynamic domains. *Logic-based artificial intelligence pp 257 - 279*, 2000.

[26] Michel Gondran et Michel Minoux. *Graphes et algorithmes*. Eyrolles, 1990.

[27] R. Fikes and N. Nilsson. Strips : A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 1 :27–120, 1971.

[28] Serge Garlatti, Sébastien Iksal, and Philippe Tanguy. Scarce : An adaptive hypermedia environment based on virtual documents and semantic web. In *Adaptable and Adaptive Hypermedia System*, pages 206–224, 2004.

[29] Frank G. Halasz and Mayer D. Schwartz. The dexter hypertext reference model. *Commun. ACM*, 37(2) :30–39, 1994.

[30] Véronique Heiwy. un modèle de ressources pédagogiques générique pour la foad.

[31] Nicola Henze. From web-based educational systems to education on the web : On the road to the adaptive web.

[32] Nicola Henze. Towards open adaptive hypermedia. 2001.

[33] Nicola Henze. Logically characterizing adaptive educational hypermedia systems. 2003.

[34] Nicola Henze and Wolfgang Nejdl. Extendible adaptive hypermedia courseware : Integrating different courses and web material. In Peter Brusilovsky, Oliviero Stock, and Carlo Strapparava, editors, *AH*, volume 1892 of *Lecture Notes in Computer Science*, pages 109–120. Springer, 2000.

[35] Stefan Hoermann, Cornelia Seeberg, Luka Divac-Krnic, Oliver Merkel, Andreas Faatz, and Ralf Steinmetz. Building structures of reusable educational content based on lom. In Johann Eder, Roland Mittermeir, and Barbara Pernici, editors, *CAiSE Workshops*, volume 75 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[36] Hubertus Hohl, Heinz-Dieter Böcker, and Rul Gunzenhäuser. Hypadapter : An adaptive hypertext system for exploratory learning and programming. *User Model. User-Adapt. Interact.*, 6(2-3) :131–156, 1996.

[37] Geert-Jan Houben, Peter Barna, Flavius Frasincar, and Richard Vdovjak. Hera : Development of semantic web information systems. In Juan Manuel Cueva Lovelle, Bernardo Martín González Rodríguez, Luis Joyanes Aguilar, José Emilio Labra Gayo, and María del Puerto Paule Ruíz, editors, *ICWE*, volume 2722 of *Lecture Notes in Computer Science*, pages 529–538. Springer, 2003.

[38] http ://dublincore.org/. Dublin core metadata initiative.

[39] http ://dublincore.org/documents/education namespace/. Dublin core - education working group - education namespace draft proposal.

[40] http ://edutool.com/papi/. Papi learner, draft 8 specification.

[41] http ://eml.ou.nl/. Educational modelling language.

[42] http ://ltsc.ieee.org/wg12/. Learning object metadata.

[43] http ://server2.tecweb.inf.puc rio.br :8000/projects/hyperde/trac.cgi. Hyperde.

[44] http ://triple.semanticweb.org/. Triple homepage.

[45] http ://www.adlnet.org/. Scorm.

[46] http ://www.cancore.ca/. Cancore learning resource metadata initiative.

[47] http ://www.imsglobal.org/profiles/index.html. Ims learner information package specification.

[48] http ://www.u-picardie.fr/ cochard/IEM/. Miage à distance "international e-miage".

[49] José Antonio Macías Iglesias and Pablo Castells. An authoring tool for building adaptive learning guidance systems on the web. In Jiming Liu, Pong Chi Yuen, Chun Hung Li, Joseph Kee-Yin Ng, and Toru Ishida, editors, *Active Media Technology*, volume 2252 of *Lecture Notes in Computer Science*, pages 268–278. Springer, 2001.

[50] H. V. Jagadish, Alberto O. Mendelzon, and Inderpal Singh Mumick. Managing conflicts between rules. *J. Comput. Syst. Sci.*, 58(1) :13–28, 1999.

[51] Craig A. Kaplan, Justine Fenwick, and James Chen. Adaptive hypertext navigation based on user goals and context. *User Model. User-Adapt. Interact.*, 3(3) :193–220, 1993.

[52] Nora Koch and Martin Wirsing. The munich reference model for adaptive hypermedia applications. In Bra et al. [7], pages 213–222.

[53] Sheila A. McIlraith and Tran Cao Son. Adapting golog for composition of semantic web services. In Dieter Fensel, Fausto Giunchiglia, Deborah L. McGuinness, and Mary-Anne Williams, editors, *KR*, pages 482–496. Morgan Kaufmann, 2002.

[54] David E. Millard, Hugh C. Davis, Mark J. Weal, Koen Aben, and Paul De Bra. Aha ! meets auld linky : integrating designed and free-form hypertext systems. In *Hypertext*, pages 161–169. ACM, 2003.

[55] Jacques Moeschler. Pragmatique du discours. *http ://www.unige.ch/lettres/linge/moeschler/Discours/Discours4/Discours* 2003.

[56] E. Pednault. Adl : Exploring the middle ground between strips and the situation calculus. *International Conference on Principles of Knowledge Representation*, 1989.

[57] David Poole, Alan Mackworth, and Randy Goebel. *Computational Intelligence : a Logical Approach*. Oxford University Press, 1998.

[58] Tapio Rauma. Diagnosis information in meta-rule adaptive fuzzy systems. In *23rd EUROMICRO Conference : New Frontiers of Information Technology*, pages 564 – 569, 1997.

[59] Raymond Reiter. Proving properties of state in the situation calculus. *Artificial Intelligence, 64(2) :337-351*, 1993.

[60] Uri J. Schild and Shai Herzog. the use of meta-rules in rule based legal computer systems. In *4th international conference on Artificial intelligence and law*, pages 100 – 109, 1993.

[61] Mark Schreiner. Meta-rules. 2001.

[62] Daniel Schwabe and Gustavo Rossi. An object oriented approach to web-based applications design. *Theory and Practice of Object Systems*, 4(4) :207–225, 1998.

[63] Josefina Sierra-Santibañez. Heuristic planning : A declarative approach based on strategies for action selection. *Artificial Intelligence*, 153 :307–337, 2004.

[64] Marcus Specht, Milos Kravcik, Leonid Pesin, and Roland Klemke. Authoring adaptive educational hypermedia in winds.

[65] Natalia Stash, Alexandra Cristea, and Paul De Bra. Explicit intelligence in adaptive hypermedia : Generic adaptation languages for learning preferences and styles. 2005.

[66] Carsten Ullrich. Description of an instructional ontology and its application in web services for education, 2004.

[67] Julita Vassileva. A task-centered approach for user modeling in a hypermedia office documentation system. *User Model. User-Adapt. Interact.*, 6(2-3) :185–223, 1996.

[68] Gerhard Weber, Hans-Christian Kuhl, and Stephan Weibelzahl. Developing adaptive internet based courses with the authoring system netcoach. In *Revised Papers from the nternational Workshops OHS-7, SC-3, and AH-3 on Hypermedia : Openness, Structural Awareness, and Adaptivity*, pages 226–238, London, UK, 2002. Springer-Verlag.

[69] Ralph M. Weischedel and Norman K. Sondheimer. Meta-rules as a basis for processing ill-formed input. *Computational Linguistics*, 9 :161–177, 1983.

[70] Franz Weitl, Christian Süß, Rudolf Kammerl, and Burkhard Freitag. Presenting complex e-learning content on the web : A didactical reference model. In *Proceedings of e-learn 2002 world conference on E-Learning in Corporate, Government, Healthcare, & Higher Education*, Montreal, Canada, 2002.

[71] Arouna Woukeu, Gary Wills, Grainne Conole, Leslie Carr, Simon Kampa, and Wendy Hall. Ontological hypermedia in education : A framework for building web-based educational portals. 2003.

[72] Hongjing Wu, Erik de Kort, and Paul De Bra. Design issues for general-purpose adaptive hypermedia systems. In *HYPERTEXT '01 : Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, pages 141–150, New York, NY, USA, 2001. ACM Press.

[73] W. Weisweber and S. Preuss. Direct parsing with metarules. In *COLING 14 : Proceedings of the 14th International Conference on Computational Linguistics*, volume IV, pages 1111–1115, 1992.

# Annexe A

# Données de l’exemple

| Généralisation | Spécialisation |
|---|---|
| bd | conception-bd |
| bd | modele-relationnel |
| bd | sgbd |
| conception-bd | modele-entite-association |
| conception-bd | normalisation-relation |
| concurrence | algo-concurrence |
| concurrence | coherence-concurrence |
| concurrence | index |
| concurrence | transaction |
| concurrence | verrous |
| dependance-fonctionnelle | dfe |
| description | conceptuel |
| description | externe |
| description | interne |
| dfe | gdfe |
| formes-normales | 1nf |
| formes-normales | 2nf |
| formes-normales | 3nf |
| formes-normales | bcnf |
| modele-relationnel | attribut |
| modele-relationnel | bd-relationnelle |
| modele-relationnel | domaine |
| modele-relationnel | operateurs |
| modele-relationnel | produit-cartesien |
| modele-relationnel | relation |
| modele-relationnel | schema-relation |
| modele-relationnel | sgbdr |
| modele-relationnel | tuple |
| normalisation-relation | cle |
| normalisation-relation | decomposition-relation |
| normalisation-relation | dependence-fonctionnelle |
| normalisation-relation | formes-normales |
| objectifs | coherence |
| objectifs | independance-physique |
| objectifs | independence-logique |
| objectifs | securite |
| operateurs | operations-ensemblistes |
| operateurs | operations-specifiques |
| operations-ensemblistes | difference-relationnelle |
| operations-ensemblistes | intersection |
| operations-ensemblistes | union |
| operations-specifiques | jointure |
| operations-specifiques | projection |
| operations-specifiques | restriction |
| optimisation | optimisation-logique |
| optimisation | optimisation-physique |
| optimisation | optimisation-statique |
| optimisation | reecriture |
| sgbd | composants |
| sgbd | concurrence |
| sgbd | description |
| sgbd | objectifs |

FIG. A.1 – Relation de généralisation pour les concepts du domaine

| | |
|---|---|
| sgbd | optimisation |
| sgbd | sgbdr |
| sgbd | sql |

FIG. A.2 – Relation de généralisation pour les concepts du domaine

| **Pré-requis** | **Requerrant** |
|---|---|
| 1nf | 2nf |
| 2nf | 3nf |
| 3nf | bcnf |
| attribut | 1nf |
| attribut | projection |
| attribut | schema-relation |
| bd | conception-bd |
| bd | modele-entite-association |
| bd | modele-relationnel |
| bd | sgbd |
| bd | sql |
| bd-relationnelle | cle |
| bd-relationnelle | decomposition-relation |
| bd-relationnelle | normalisation-relation |
| cle | 2nf |
| dependance-fonctionnelle | dfe |
| dfe | bcnf |
| dfe | gdfe |
| domaine | produit-cartesien |
| domaine | relation |
| domaine | schema-relation |
| formes-normales | 1nf |
| modele-relationnel | attribut |
| modele-relationnel | bd-relationnelle |
| modele-relationnel | domaine |
| modele-relationnel | operateurs |
| modele-relationnel | produit-cartesien |
| modele-relationnel | relation |
| modele-relationnel | schema-relation |
| modele-relationnel | tuple |
| produit-cartesien | jointure |
| produit-cartesien | relation |
| relation | attribut |
| relation | cle |
| relation | dependance-fonctionnelle |
| relation | difference-relationnelle |
| relation | formes-normales |
| relation | intersection |
| relation | jointure |
| relation | projection |
| relation | restriction |
| relation | schema-relation |
| relation | union |
| schema-relation | cle |
| schema-relation | decomposition-relation |
| schema-relation | dependance-fonctionnelle |
| schema-relation | difference-relationnelle |
| schema-relation | intersection |
| schema-relation | union |
| sgbd | coherence |
| sgbd | composants |
| sgbd | concurrence |

FIG. A.3 – Relation de pré-requis pour les concepts du domaine

| | |
|---|---|
| sgbd | description |
| sgbd | independence-logique |
| sgbd | independence-physique |
| sgbd | objectifs |
| sgbd | optimisation |
| sgbd | optimisation-logique |
| sgbd | optimisation-physique |
| sgbd | optimisation-statique |
| sgbd | reecriture |
| sgbd | securite |
| sgbd | sgbdr |
| tuple | decomposition-relation |
| tuple | dependance-fonctionnelle |
| tuple | difference-relationnelle |
| tuple | intersection |
| tuple | projection |
| tuple | restriction |
| tuple | union |

FIG. A.4 – Relation de pré-requis pour les concepts du domaine

| **Element A** | **Element B** |
|---|---|
| 1nf | 2nf |
| 1nf | 3nf |
| 1nf | bcnf |
| 2nf | 3nf |
| 2nf | bcnf |
| 3nf | bcnf |
| externe | conceptuel |
| interne | conceptuel |
| interne | externe |
| optimisation-logique | optimisation-physique |
| optimisation-logique | optimisation-statique |
| optimisation-statique | optimisation-physique |

FIG. A.5 – Relation de contraste pour les concepts du domaine