Rédacteurs	Nour EL MAWAS
Relecteurs	Lahcen Oubahssi, Pierre Laforcade
Date	26/02/13
Référence	GRAPHIT-D4.1
Version	0.3

Méthode d'identification et de formalisation des langages des plateformes

D4-1

■ Historique du document

Version	Date	Auteurs	**Modi**ficationsn
0.1	12/11/13	Nour EL MAWAS	
0.2	31/01/14	Nour EL MAWAS	Modification de la méthode d'identification et de formalisation
0.3	18/02/14	Lahcen Oubahssi	?????????
0.4	21/02/14	Nour EL MAWAS	Prise en compte des retours de LO et PL
0.5			

■ Table des matières

Index des Figures

1 Objectifs de ce document

Le projet GraphiT (Graphical Visual Instructional Design Languages for Teachers) a pour objectif d'élaborer et instrumenter des langages de modélisation pédagogiques graphiques centrés sur le métier des plate-formes de formation et dirigés vers les besoins des praticiens. Il s'appuie sur les théories et pratiques issues des domaines scientifiques de l'Ingénierie Dirigée par les Modèles (IDM) et du Domain Specific Modeling (DSM). Ce projet comprend 6 tâches. Ce document concerne la tâche 4 du projet, qui a pour objectif d'établir une analyse des besoins des plateformes, et plus précisément à l'identification et la formalisation des métiers de conception pédagogique des plateformes.

Notre travail de recherche s'inscrit dans le domaine de l'ingénierie et de la réingénierie des Environnements Informatiques pour l'Apprentissage Humain (EIAH). Un EIAH est la combinaison d'un système informatique et d'un système d'apprentissage. Selon [Tchounikine et al. 2004], un EIAH représente un environnement informatique proposant une intention didactique et/ou pédagogique et dont la finalité est de susciter ou d'accompagner un apprentissage. En tant qu'environnement informatique, un EIAH offre des services spécifiques dédiés au domaine de l'éducation pour ses multiples clients et usagers (des universités, des académiques, des enseignants, des apprenants, des gestionnaires, etc.). Les services offerts incluent généralement un contrôle d'accès, des outils de communication (synchrones et/ou asynchrones), l'administration des groupes d'utilisateurs, etc. En tant qu'environnement d'apprentissage, un EIAH est proposé pour toute personne impliquée dans un processus d'apprentissage ou dans un parcours pédagogique. Il embarque une intention didactique/pédagogique et des éléments métiers relevant de la conception pédagogique. Durant les dernières décades, plusieurs systèmes informatiques spécifiques aux EIAH sont apparus tels que les plateformes d'apprentissage en ligne, les plateformes e-learning (ou de la FOAD), les systèmes de gestion de l'apprentissage, les environnements numérique d'apprentissage, etc.

Actuellement, les plateformes de Formation Ouverte et A Distance (FOAD) sont largement utilisées et ne sont pas limitées à l'apprentissage distant, elles sont aussi utilisées dans le cadre des formations mixtes, en effet, la plupart des organisations académiques comme les établissements scolaires et universitaires fournissent à leurs enseignants des plateformes dont l'objectif est d'améliorer ou de compléter leurs cours présentiels par des activités supplémentaires comme le simple accès à des ressources ou à des évaluations en ligne. Dans ce travail, nous nous focalisons particulièrement sur les plateformes de formation à distance, encore appelées systèmes LMS (Learning Management System). Ces dispositifs ont fait l'objet de plusieurs travaux de recherche et de développement. Plusieurs définitions ont été attribuées aux plateformes de formation à distance. Par exemple, [Oubahssi 2005] définit une plateforme de formation à distance comme étant "un logiciel de création et de gestion de contenus pédagogiques et de supports aux acteurs destiné à trois types d'utilisateurs : l'enseignant, l'apprenant et l'administrateur. Elle regroupe les outils nécessaires aux trois types d'intervenants permettant d'incorporer des ressources pédagogiques multimédias, de participer à des activités et d'effectuer le suivi pédagogique et administratif des apprenants".

Plusieurs travaux ont proposé des métamodèles des plateformes de formation [Caron 2007] [Abdallah 2009], mais aucun travail ne s'est intéressé à définir un processus d'identification et de formalisation des langages de ces plateformes.

Ce document sur la méthode d'identification et de formalisation des langages des plateformes a pour objectif de définir les analyses et les étapes nécessaires à l'identification et la

formalisation du langage de conception pédagogique d'une plateforme de formation.

Ce travail de formalisation devrait permettre de mettre en évidence une des hypothèses du projet : les plateformes de formation ou les LMSs ne sont pas pédagogiquement neutres et elles embarquent un langage de conception pédagogique implicite spécifique.

Le processus d'analyse du langage de conception pédagogique d'une plateforme que nous proposons permet de formaliser le langage de conception d'une plateforme à partir de trois points de vue complémentaires: le point de vue IHM-macro, le point de vue fonctionnel et le point de vue technique.

La suite de ce document est découpée comme suit. La section 2 présente notre vue générale du processus d'identification et de formalisation ainsi que le formalisme utilisé pour modéliser les différentes étapes de ce processus. La section 3 détaille notre analyse IHM-mcaro. La section 4 définit l'étape de factorisation du modèle IHM-macro dans ce processus. La section 5 détaille notre analyse fonctionnelle. La section 6 décrit l'analyse micro permettant d'aboutir au métamodèle d'une plateforme de formation. La section 7 conclut le document et présente ses perspectives.

2 Vue générale du processus

Les plateformes de formation sont composées de plusieurs métiers et langages (métier pédagogique, d'administration, de gestion, etc.) destinés à différents usagers (apprenants, enseignants, tuteurs, etc.) [Abedmouleh 2012]. Dans le cadre de ce travail, le langage qui nous intéresse concerne la scénarisation pédagogique sur ce type d'outils de formation. Pour identifier et formaliser ce langage de conception pédagogique, un processus a été défini dans le cadre des travaux de thèse de Monsieur Abedmouleh [Abedmouleh 2012] [Abedmouleh et al. 2012a] [Abedmouleh et al. 2012b] [Abedmouleh et al. 2012C] puis nous avons poursuivi le travail sur ce processus dans le cadre de ce projet Graphit. Ce processus est le résultat de plusieurs expérimentations et analyses effectuées sur plusieurs plateformes de formation. Après l'appropriation de ces plateformes, nous avons constaté que chacune d'entre elles embarquait un paradigme spécifique. Cependant, une même analyse pouvait être définie afin de guider le travail d'identification et de formalisation de leur métier pédagogique. Le processus vise la communauté des LMSs/plateformes de formation telle que les ingénieurs pédagogiques, les développeurs-concepteurs et les experts des plateformes ayant une compétence en informatique. Il facilitera l'explicitation du métier de conception pédagogique de leur plateforme.

Le processus est spécifié en se basant sur trois points de vue différents, mais complémentaires : un point de vue centré IHM macro, un point de vue fonctionnel et un point de vue micro. Le premier point de vue porte sur l'analyse des IHMs selon deux stratégies :

- (1) l'analyse des situations existantes sur la plateforme (déjà spécifiées par des enseignants) et
- (2) l'analyse des interfaces liées à la spécification de nouvelles situations.

Suite à l'analyse IHM-macro nous avons factorisé le modèle IHM-macro pour obtenir un modèle macro simplifié. Le deuxième point porte sur l'identification des fonctions déjà existantes sur le système. Le troisième point se base sur l'analyse micro du langage de conception pédagogique d'une plateforme de formation.

La figure 15 illustre le processus que nous proposons. Il est composé de quatre parties principales : l'analyse IHM-macro, la factorisation du modèle IHM-macro, l'analyse fonctionnelle, ainsi que l'analyse micro. La répartition du processus en quatre étapes principales est le résultat de l'étude de plusieurs plateformes ainsi que de plusieurs réflexions, de recherches et de réunions de travail.

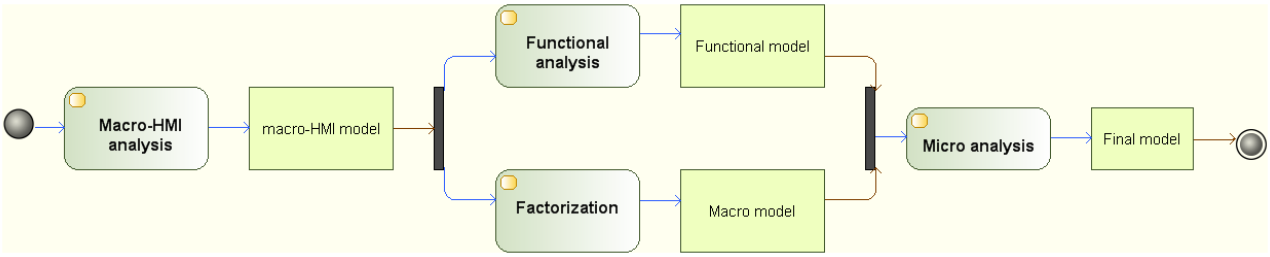


Illustration 1: Processus global d'analyse du langage de conception pédagogique d'une plateforme

Les sections suivantes présentent un aperçu de chaque partie/sous-partie du processus.

2.1 Formalisme des modèles issus des différentes étapes du processus global

Pour formaliser les modèles issus des différentes étapes du processus, nous avons étudié les formalismes possibles (schéma XML, métamodèle, cartographie (mindmap), modèle conceptuel de données, etc.).Nous avons choisi de représenter le modèle IHM-macro, le modèle macro et le modèle final par le biais du format d'un métamodèle. Le modèle fonctionnel est représenté par un diagramme de cas d'utilisation UML.

Nous avons choisi le formalisme métamodèle car il permet de présenter d'une façon claire les éléments d'une plateforme, leurs attributs, les relations entre ces éléments et leurs cardinalités.

Un métamodèle est une description formelle d'un domaine spécifique en termes d'éléments, de syntaxe et de sémantique de la notation. Notre choix a été motivé par plusieurs raisons. Ce format servira pour le développement des outils de conception pédagogique en adoptant l'approche *Domain-Specific Modeling*. Dans notre travail, nous l'utiliserons également pour le développement des modules de communication entre les plateformes de formations et les outils de conception concernés.

Le métamodèle du langage de conception pédagogique sera complété au fur et à mesure de l'application du processus global d'analyse de conception pédagogique d'une plateforme. Les éléments identifiés sont représentés par des méta-classes. Également, les attributs identifiés sont représentés par des méta-attributs au sein de leurs méta-classes. Enfin, les relations entre les éléments sont identifiées et représentées par des relations entre les méta-classes associées. Nous détaillons davantage la spécification du métamodèle au sein des sections suivantes.

Nous avons choisi la modélisation en environnement Eclipse : EMF (Eclipse Modeling Framework). En effet, EMF permet non seulement de modéliser facilement le métamodèle d'une plateforme de formation, mais aussi de faciliter le développement des prototypes (des éditeurs de modèles) grâce à la génération automatique du code source Java.

C'est un *framework* de modélisation, une infrastructure de génération de code et des applications basées sur des modèles de données structurées. Partant d'une spécification décrite généralement sous la forme d'un modèle en XMI, EMF fournit des outils permettant de produire des classes Java représentant le modèle avec un ensemble de classes pour adapter les éléments du modèle afin de pouvoir les visualiser, les éditer avec un système de commandes et les manipuler dans un éditeur.

EMF s'appuie sur le métamodèle Ecore (Figure 2) qui respecte les principes définis par le eMOF (*Extended Meta Object Facility*), un standard OMG. Les méta-éléments à instancier pour définir ce modèle sont :

- EClass : désigne les classes des modèles, identifiées par un nom, peuvent contenir des *StructuralFeatures* (attributs ou références). Elle supporte l'héritage multiple. Elle peut être une classe abstraite (pas d'instance possible) ou une interface (pas d'implémentation générée).
- EAttribute : est identifié par un nom et un type. Des bornes mini et maxi sont utilisées pour la cardinalité.
- EReference : est une association entre deux classes, identifiée par un nom et un type. Les bornes mini et maxi sont utilisées pour la cardinalité. Une association de type composition est autorisée (*containment*).

  La relation d'association par référence est représentée par un trait plein orienté.

  La relation d'association par composition se représente par un trait plein orienté et orné d'un losange noir du côté de la classe composite.

  La relation de généralisation est présentée par un trait plein, orienté de la classe spécialisée (fille) vers son modèle (mère) et se terminant par une flèche fermée. Chaque classe possède des caractéristiques qui lui sont propres. Lorsqu'une classe fille hérite d'une classe mère, elle peut alors utiliser ses caractéristiques.
- EDataType : désigne le type primitif ou le type objet défini par Java.
- EPackage : désigne les *packages* des modèles qui sont des conteneurs de *classifiers* (classes et types). Il est défini par un nom de *package* (unique) et une URI pour l'identification lors de la sérialisation.
- EOperation : désigne les opérations d'une classe pouvant être invoquées. Elle est identifiée par un nom, un type de retour et des paramètres. Elle autorise les exceptions.
- EEnum : désigne le types énumérés parmi un ensemble de EENumLiteral.

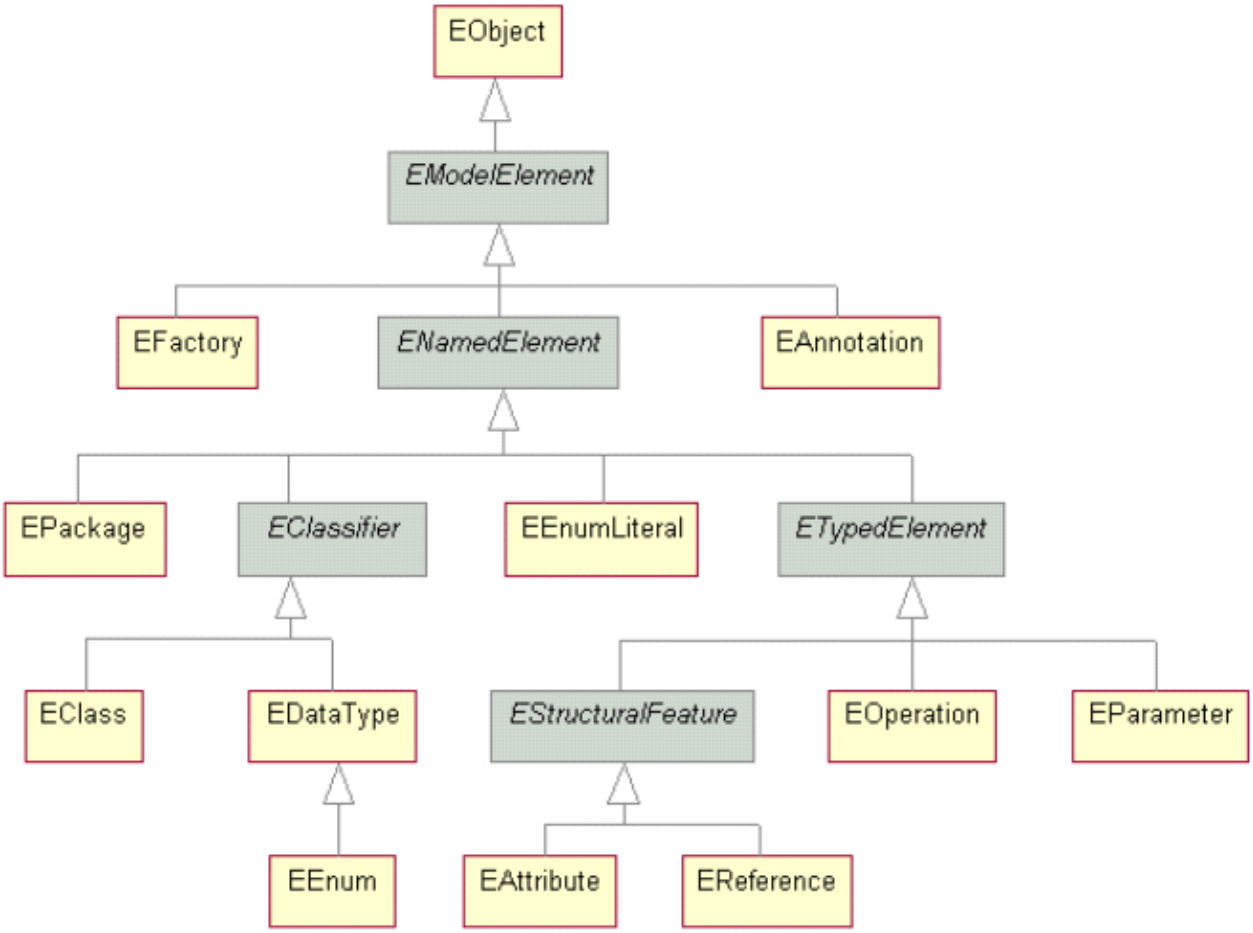


Illustration 2: Le métamodèle ECORE

Dans notre modèle final, nous utilisons les méta-éléments suivants : EClass, EEnum, EAnnotation, EAttribute et EReference. Nous avons écarté les autres méta-éléments du métamodèle ECORE parce que nous nous intéressons aux propriétés d'un concept pédagogique et non pas à ses fonctionnalités et à son comportement.

Le modèle EMF peut être visualisé selon plusieurs représentations. Nous pouvons avoir une vue de type arbre avec l'extension ecore (Figure 3) et une vue de type diagramme avec l'extension ecorediag généré à partir du modèle ecore si le framework GMF runtime est installé (Figure 4).

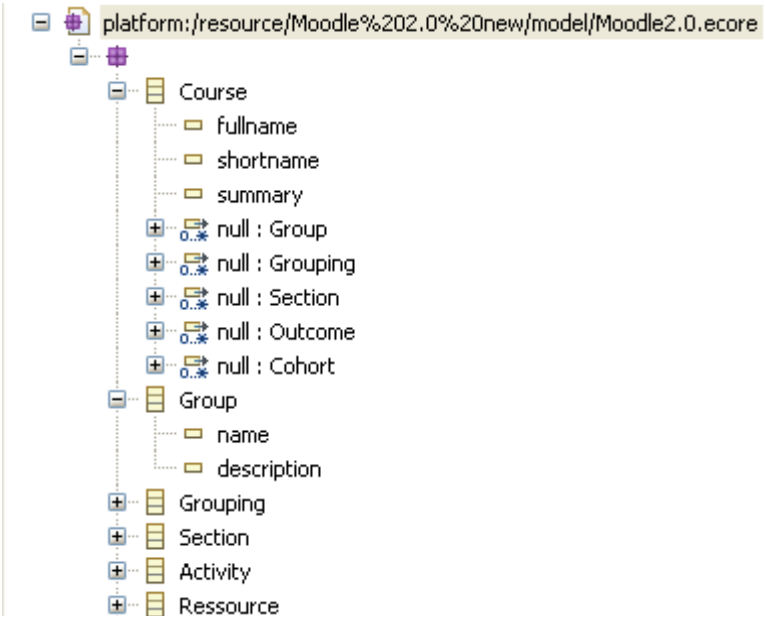


Illustration 3: exemple d'un fichier ecore, éditeur arborescent des métamodèles

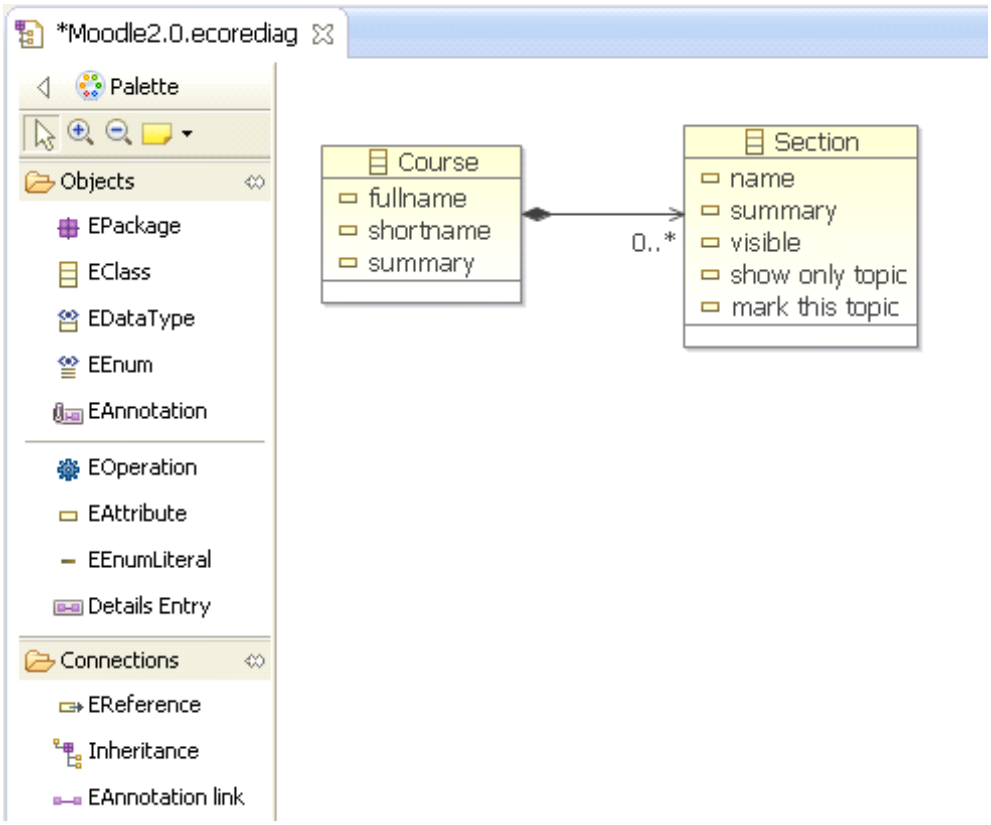


Illustration 4: exemple d'un fichier ecorediag, éditeur graphique de métamodèles

Nous adoptons une architecture de méta-modélisations à quatre couches (Figure 5) :

M0 : Ce niveau représente le système réel à modéliser. Dans notre cas, le système à modéliser est le cours réel mis en œuvre sur la plateforme.

M1 : Ce niveau représente modèle représentant le système réel à modéliser (instance du modèle ecore. Dans notre cas, il s'agit du scénario à déployer sur la plateforme.

M2 : Ce niveau représente le méta-modèle décrivant le langage de modélisation (le modèle Ecore). Dans notre cas, il s'agit du métier de conception pédagogique de la plateforme.

M3 : Un niveau qui comporte le méta-méta-modèle MOF décrivant le langage de méta-modélisation (ECORE).

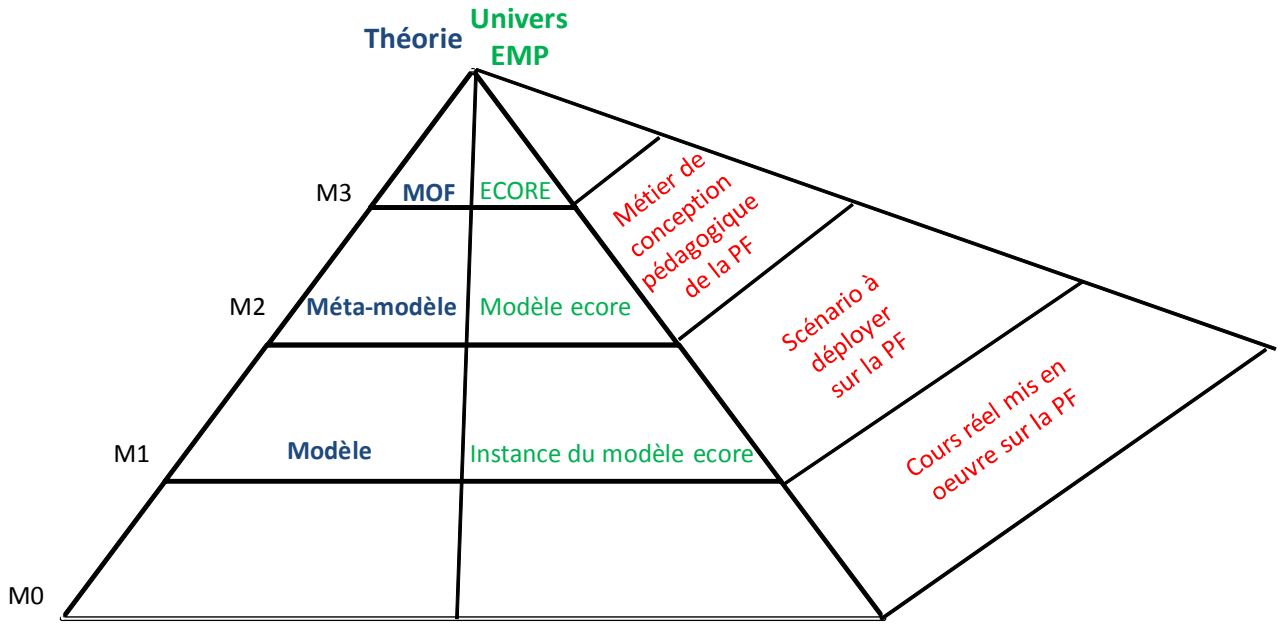


Illustration 5: Architecture EMF à 4 niveaux

3 Analyse IHM macro

L'analyse IHM-macro consiste à identifier les IHMs de la plateforme relevant de la conception pédagogique.

Les plateformes de formation à distance sont généralement composées de plusieurs IHMs qui sont développées pour des diverses finalités et pour des catégories d'usagers différentes. Un nombre important des IHMs ne portent pas sur la conception pédagogique des situations d'apprentissage. Elles font partie des IHMs que nous écartons telles que les IHMs d'administration, les IHMs de gestion, etc. Pour cela, notre première hypothèse consiste à encadrer le champ d'analyse.

Cette hypothèse fait l'objet de la première activité du processus proposé (analyse IHM-macro) qui concerne l'analyse des IHMs de la plateforme à une échelle macro. Cette analyse

consiste principalement à identifier les interfaces relevant de la conception pédagogique. Son objectif est de :

1. capitaliser les concepts principaux de conception pédagogique situés dans ces interfaces
2. produire le diagramme de classe de ces interfaces. Chaque interface est représentée par le concept principal qu'elle décrit.

Ces concepts sont identifiés grâce à deux méthodes :

- l'analyse des titres des interfaces,
- l'analyse des chemins de navigation / des URLs des interfaces.

L'analyse IHM-macro concerne toutes les interfaces de la plateforme. Dans une première étape, il est important de parcourir tous les liens possibles dans une interface donnée. Souvent, ces liens mènent à des nouvelles interfaces. Cependant, l'analyse IHM-macro peut se limiter aux interfaces situées dans l'espace associé aux enseignants-concepteurs. Le parcours de ces interfaces dépend du paradigme de la plateforme. Par exemple, la plateforme Moodle impose de parcourir ses interfaces en suivant une démarche descendante. Elle dispose d'une interface principale pour la spécification et la représentation du contenu du cours (figure 13). Les autres interfaces telles que l'interface d'ajout d'un forum (figure 7) sont accessibles à partir de l'interface principale.

Cependant, dans la plateforme Ganesha, le parcours des interfaces s'effectue dans une démarche latérale et descendante, car son paradigme impose la création de certains éléments du module (cours) séparément. Ensuite, les relations entre ces éléments sont spécifiées. Par exemple, les séquences d'activités sont tout d'abord spécifiées, ensuite elles sont associées aux modules concernés. Quant aux activités, elles ne sont accessibles qu'à partir de l'interface des séquences.

Nous avons modélisé l'analyse IHM-macro par un diagramme d'activité UML (*Unified Modeling Language*) qui est représenté par la figure 6. La première étape de cette analyse consiste à choisir une première interface (une interface principale). Ensuite, l'analyse doit déterminer si cette interface dispose d'un critère pédagogique, tel que les interfaces d'ajout des activités et des ressources pédagogiques (leçon, quiz, etc.). Les interfaces dont leurs titres ou leurs chemins de navigation relèvent de la conception pédagogique sont prises en compte. Le concept principal et le plus significatif de cette interface doit être identifié et ensuite représenté sur le modèle IHM-macro. Enfin, les relations entre les concepts du modèle doivent être identifiées et définies sur le modèle.

L'identification des interfaces est un processus itératif. Lorsqu'une nouvelle interface est identifiée, l'analyste étudie les liens situés sur cette interface afin d'accéder à des nouvelles interfaces. Ensuite, si cela n'est pas possible, il vérifie si des nouvelles interfaces sont accessibles à partir des interfaces antérieurement identifiées. Les interfaces non relevant de la conception pédagogique ne sont pas analysées ni représentées dans le modèle IHM macro (figure 6).

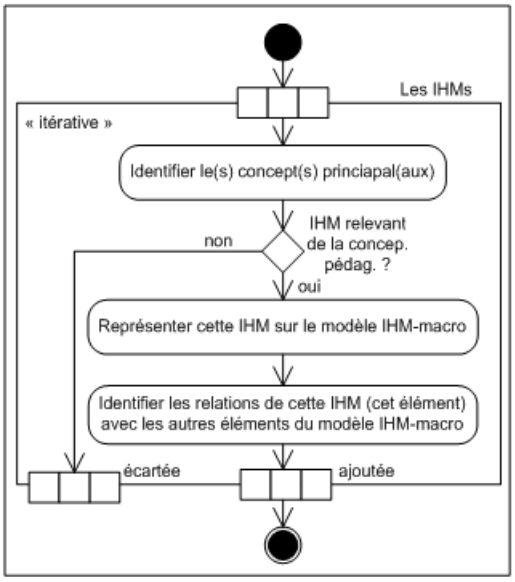


Illustration 6: Diagramme d'activité de l'analyse IHM-macro

3.1 Identification des principaux concepts pédagogiques

L'identification des principaux concepts pédagogiques dans une interface représente la première étape de notre analyser IHM macro, elle peut être réalisée par deux méthodes : l'analyse sémantique des titres (des interfaces, des blocs, etc.) et l'analyse des chemins de navigation. Nous détaillons ci-dessous ces deux méthodes.

3.1.1 Analyse des titres

Le développement des plateformes exige une ergonomie spécifique visant à faciliter l'utilisation et la manipulation de leurs interfaces et à organiser les informations de façon simple et claire. Cette ergonomie est susceptible de mettre les titres des interfaces en relief ainsi que les titres des blocs, des menus, des sections et des parties principales d'une interface particulière. Souvent, chaque interface dispose de son propre titre. Le titre est généralement situé en haut de la page avec une mise en forme spécifique (taille plus grande, couleur différente, etc.). Il désigne le concept principal de l'interface ou bien son occurrence. Toutefois, nous trouvons parfois plusieurs titres dans une même interface. Ainsi, plusieurs propositions sont possibles :

- Quand plusieurs éléments principaux sont représentés sur une interface ; les concepts les plus pertinents sont choisis.
- Les titres décrivent la hiérarchie des éléments de l'interface. Par exemple, le premier titre représente le concept ascendant. Le deuxième représente le concept principal de l'interface tandis que les autres sont des sections et des sous-éléments du concept principal. Dans ce cas, le deuxième titre et ses sections sont les éléments représentatifs de l'interface.

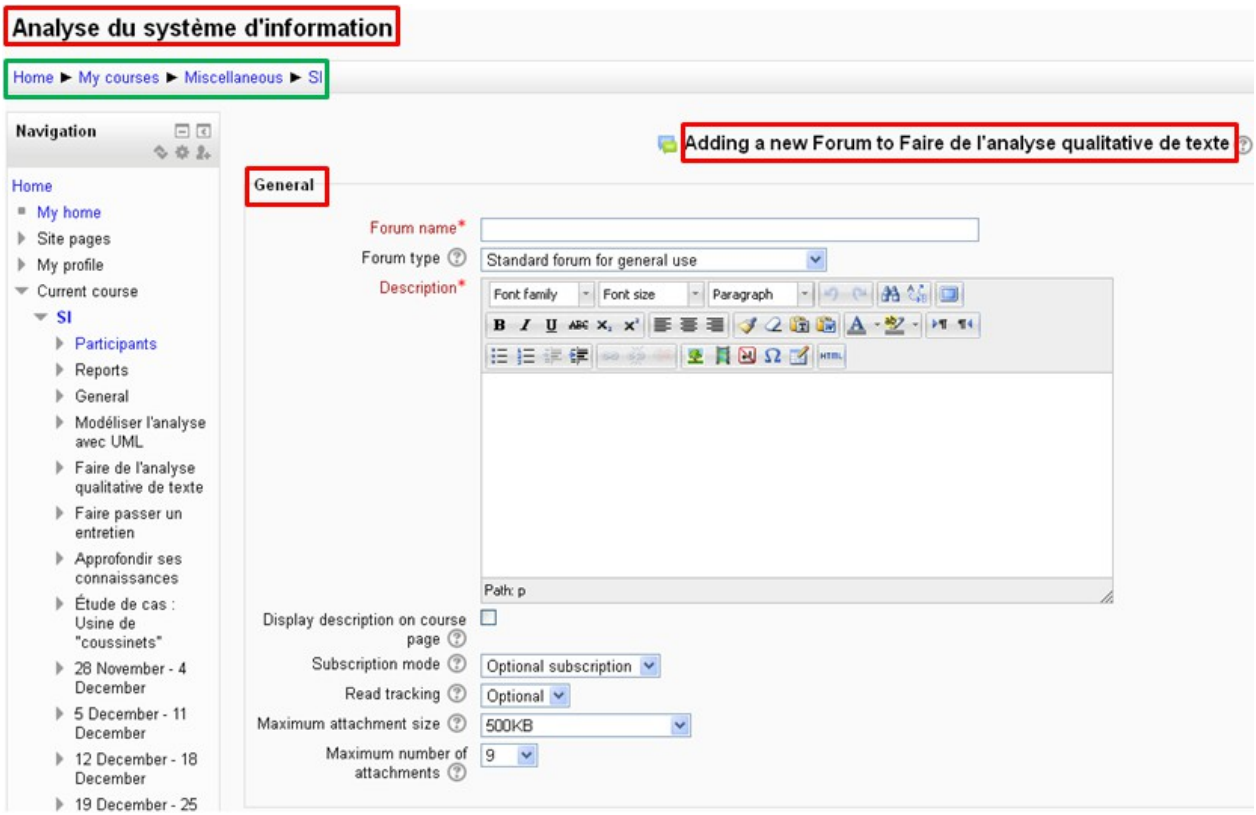


Illustration 7: Interface d'ajout d'un forum sur Moodle

Par exemple, la figure 7 illustre l'interface de spécification d'un forum sur la plateforme Moodle. Elle montre trois titres principaux dans son interface : *'Analyse du système d'information'*, '*Adding a new Forum'* et *'General'* (encadrés en rouge). Nous constatons que l'interface du *'forum'* est accessible à partir de l'interface du *'Analyse du système d'information'*. Le concept principal de cette interface est *'forum'. 'Analyse du système d'information'* représente son élément ascendant et *'General'* son élément descendant.

3.1.2 Analyse des chemins de navigation

Les plateformes de formation montrent clairement sur leurs interfaces le chemin de navigation. Souvent, ces chemins sont désignés par la concaténation des concepts principaux des interfaces déjà parcourues ainsi que l'interface en cours. Leur analyse permet de déterminer l'agencement général de ces interfaces et d'identifier certaines relations. Outre les chemins de navigation, les URLs de plateformes de formation sont également une source d'information pour repérer le titre de l'interface et par la suite son concept principal.

Par exemple, l'analyse du chemin de navigation *'Accueil/Cours/Analyse du système d'information/Forum'* sur la figure 7 permet d'identifier les interfaces déjà parcourues et la relation de la hiérarchie ou d'agencement entre elles. Dans cet exemple, *'Analyse du système d'information '* représente un cours sur la plateforme Moodle dont *'Forum'* est un élément de leur contenu et 'General' est une section dans ce contenu.

3.2 Formalisme du modèle IHM-macro

Nous rappelons que l'analyse IHM-macro consiste à recapitaliser les interfaces relevant de la conception pédagogique dans un modèle, appelé modèle IHM-macro.

La spécification du modèle IHM-macro est réalisée au fur et à mesure qu'une nouvelle interface à critère pédagogique est identifiée que nous représentons par son concept principal. Ensuite, les relations avec les autres éléments du modèle doivent être spécifiées (y compris leurs cardinalités). Ces relations sont basées principalement sur les relations entre les interfaces. La figure 8 illustre un exemple de modèle IHM-macro issu de l'application de l'analyse IHM-macro sur la plateforme Moodle.

Un cours est composé de sections, de catégories, de groupes, de groupements, d'objectifs et de barèmes. Chaque section est composée d'activité(s) et de ressource(s). Moodle propose aux enseignants-concepteurs 14 activités pédagogiques : Atelier, Base de données, Chat, Consultation, Devoir, Feedback, Forum, Glossaire, Leçon, Outil externe, Paquetage SCORM, Sondage, Test et Wiki. Concernant les ressources, Moodle propose 7 types de ressources différentes: Dossier, Etiquette, Fichier, Livre, Page, Paquetage IMS Content et URL.

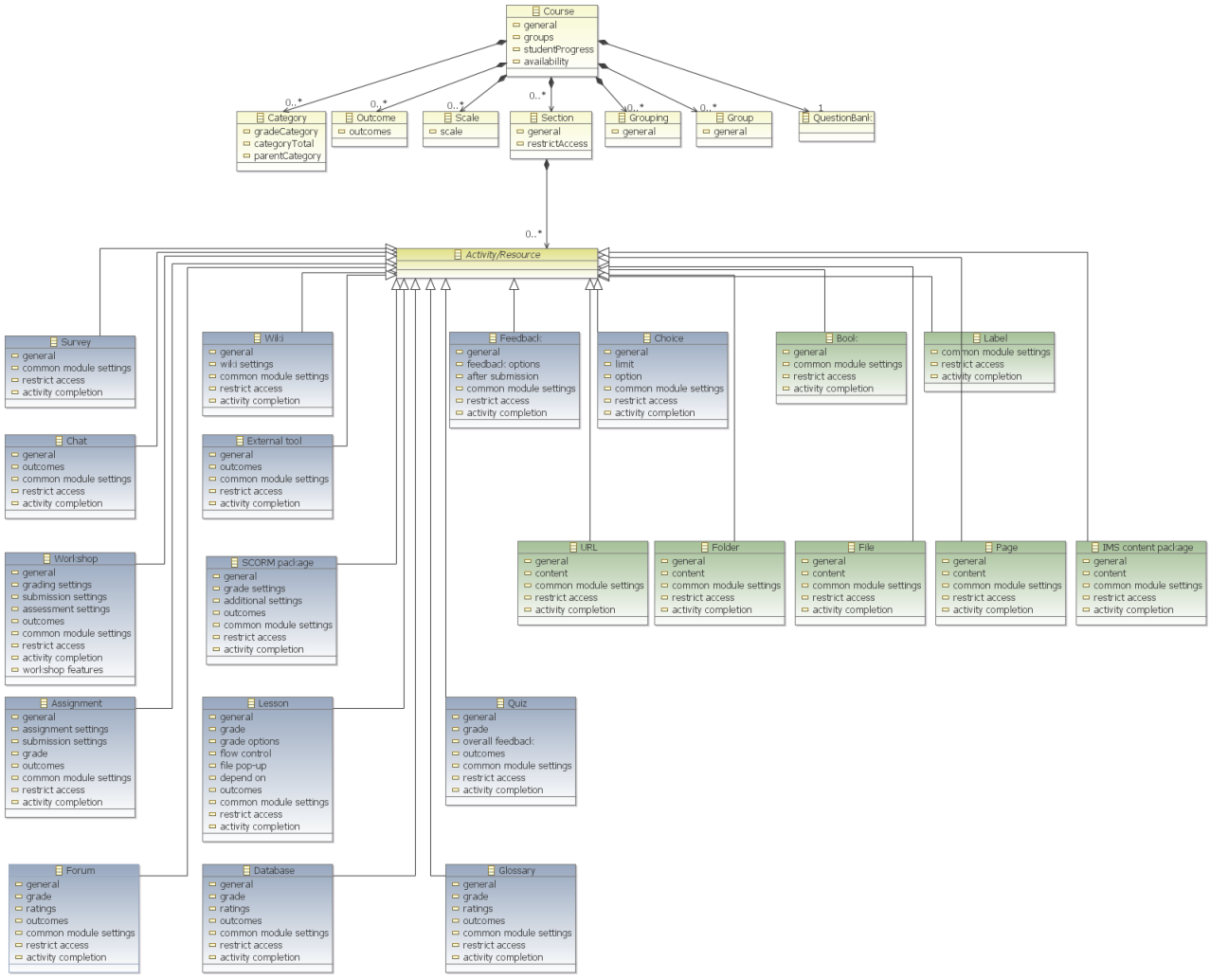


Illustration 8: Modèle IHM-macro de Moodle

L'analyse IHM-macro permet d'encadrer le champ d'analyse et d'identification du langage de conception pédagogique d'une plateforme en identifiant les IHMs liées à la conception pédagogique d'une situation d'apprentissage. Cependant, cette analyse ne permet pas d'identifier et de formaliser l'ensemble de ce langage spécifique et ses éléments caractéristiques. D'autres analyses complémentaires, telles que les analyses fonctionnelles et micro, sont ainsi proposées par ce processus.

Le modèle IHM macro constitue une base pour les phases de factorisation et d'analyse fonctionnelle.

4 Factorisation

Cette étape vise à identifier les éléments communs résultants de l'analyse IHM macro.

Nous avons remarqué par exemple pour la plateforme Moodle que toutes les activités et les ressources ont en commun les sections suivantes : « common module settings », « restrict access » et « activity completion ». De même, toutes les activités ont en commun la section « general ».

À travers cette factorisation, nous pouvons déduire le modèle macro d'une plateforme (Figure 1). Ce modèle constitue une base pour notre analyse micro.

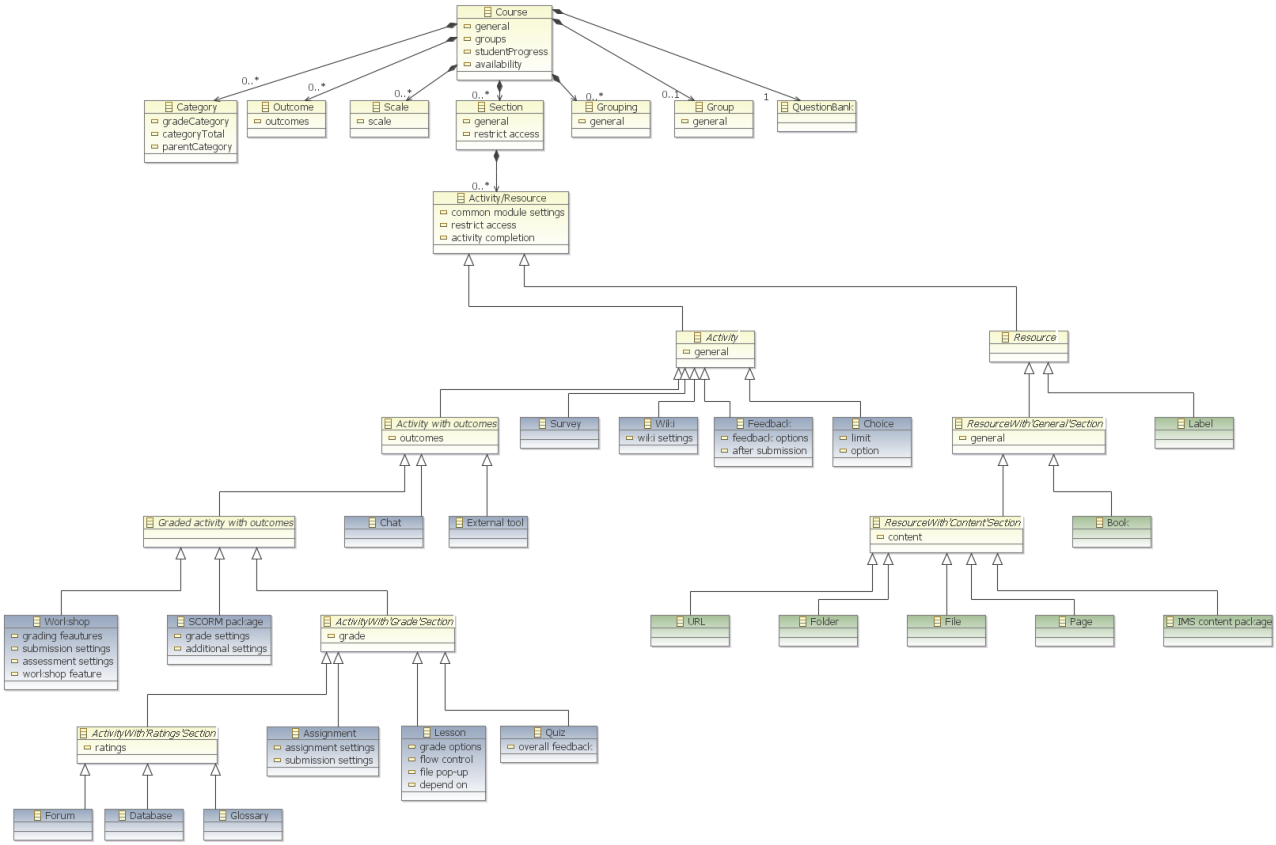


Illustration 9: Modèle Macro de Moodle

La figure 9 présente le résultat de l'application de la factorisation au modèle IHM-macro de moodle (Figure 8). La factorisation permet de factoriser les attributs communs à plusieurs classes. Par exemple, dans la plateforme Moodle, toutes les activités et les ressources ont en commun les attributs *common module settings*, *restrict access* et *activity completion*. Grâce à la factorisation, nous pouvons mettre ces attributs dans la classe *Activity/Resource* au lieu de les mettre 21 fois pour chaque activité et ressource.

5 Analyse fonctionnelle

Dans le domaine du génie logiciel, l'analyse fonctionnelle est une étape du cycle de vie d'un logiciel se situant dans la phase de spécification et d'analyse du besoin. Elle vise à exprimer le besoin en termes de services et de fonctionnalités attendus plutôt qu'en termes de solutions. Dans le cadre de notre travail, l'analyse fonctionnelle consiste à identifier les fonctions déjà existantes sur une plateforme de formation. Ces fonctions représentant le noyau fonctionnel de la plateformesont généralement représentées dans des interfaces d'interaction (IHM) afin d'être facilement accessible à un utilisateur final.

5.1 Démarche de l'analyse fonctionnelle

Avant de mener une analyse fonctionnelle d'une plateforme de formation, plusieurs documents et modèles (cahier des charges, modèle fonctionnel existant, etc.) peuvent représenter une première source d'information pour l'identification de ces fonctionnalités. Dans le cadre de nos travaux, nous nous intéressons à l'identification des fonctionnalités de la plateforme à partir de ses IHMs.

Les fonctionnalités (pédagogiques et administratives) sont déployées implicitement sur les interfaces des dispositifs d'apprentissage (plateformes et LMSs) via des éléments ou des composants de l'interface graphique (des boutons, des icones, des figures, des liens hypertextes, etc.) qui facilitent l'interaction avec l'utilisateur. L'identification de ces fonctionnalités est basée :

1. sur l'appropriation des différents éléments formant une interface,
2. sur la capacité de perception et d'interprétation de l'analyste (ingénieur pédagogique, expert de la plateforme, etc.).

Souvent, il n'est pas aisé de déterminer si une fonctionnalité relève de la conception pédagogique ou non. Pour cela, il est important d'étudier les critères pédagogiques de ses sous-fonctionnalités. Pour analyser les fonctionnalités pédagogiques d'une plateforme, nous avons suivi la démarche suivante :

1. Encadrer le champ d'analyse des interfaces en fixant les zones d'analyse des IHMs. Nous avons constaté que, pour certaines plateformes, plusieurs zones sont communes entre ses interfaces et pouvant être analysées une seule fois. Les zones ayant des intérêts administratifs ne sont pas concernées par cette analyse.
2. Choisir une zone particulière, puis tester et analyser d'un point de vue fonctionnel les composants d'interface graphique de cette zone (widgets).
3. Attribuer une fonction représentative à chaque composant. Les fonctions portant sur des objectifs non pédagogiques tels que les fonctions d'affichage, de gestion, d'administration,

etc. ne sont pas prises en compte ni représentées sur le modèle fonctionnel. Par exemple, sur la plateforme Moodle , les fonctions *'choisir la taille maximale des fichiers'* et *'sauvegarder le cours'* sont des fonctionnalités techniques non liées à la conception pédagogique.

4. Identifier les fonctionnalités pédagogiques et les représenter sur le modèle d'analyse fonctionnelle

Illustration 10: Extrait d'interface de cours sur Moodle

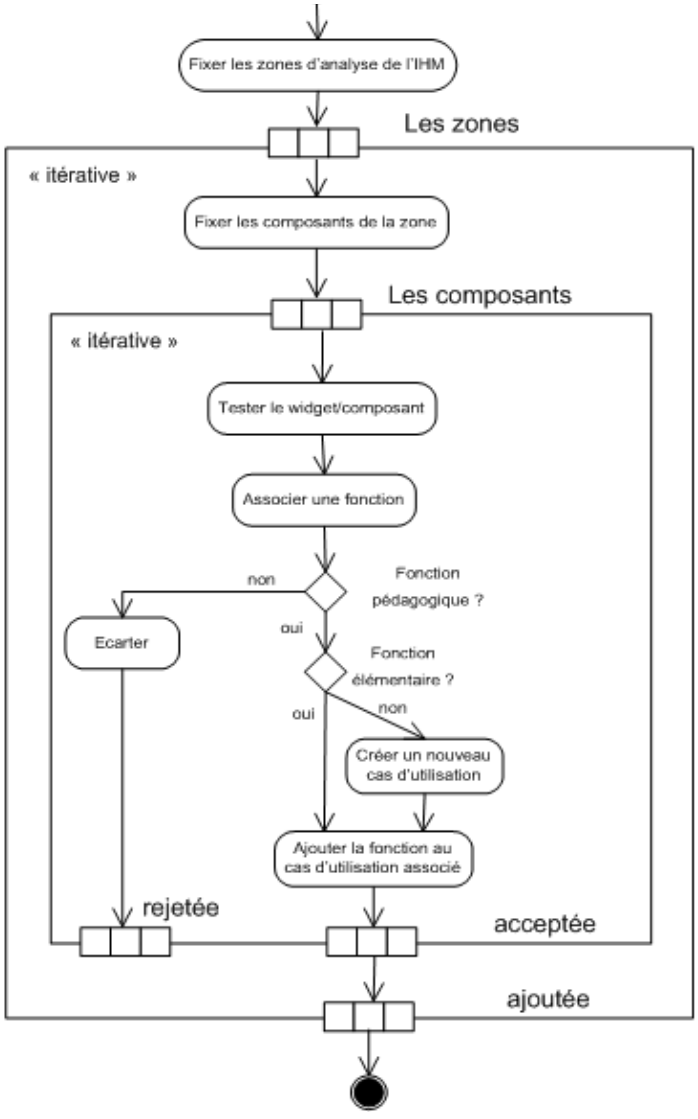


Illustration 11: Diagramme d'activité de l'analyse fonctionnelle

Par exemple, l'analyse de l'extrait d'interface de cours sur Moodle (illustration 10) consiste à analyser fonctionnellement les composants de cette interface graphique tels que les éléments 1, 2, 3 et 4. Les éléments 1, 2 et 3 désignent respectivement les fonctions *'cacher une section', 'modifier le résumer'* et *déplacer un élément de cours'* qui sont pédagogiques. Lorsqu'une fonctionnalité est non-élémentaire, ses sous-fonctionnalités sont représentées dans un nouveau diagramme de cas d'utilisation qui sera ensuite associé au diagramme principal.

La figure 11 illustre l'analyse fonctionnelle décrite par le biais d'un diagramme d'activité du langage UML.

5.2 Formalisme du modèle fonctionnel

Nous rappelons que l'analyse fonctionnelle consiste à expliciter les fonctionnalités relevant de la conception pédagogique. Ces fonctionnalités sont représentées, dans le modèle fonctionnel, indépendamment des composants qui les réalisent.

Plusieurs représentations du modèle fonctionnel sont possibles[1] telles que :

- *Hierarchical Input Process Output* (HIPO and IPO) [Boehm 2007],
- *Structured Analysis and Design Technique* (SADT) [Marca et McGowan 1987],
- *Integration Definition for Function Modeling* (IDEF0) [Ryu et Yücesan 2007],
- *Business Process Modeling Notation* (BPMN) [Muehlen et Recker 2008],
- *Axiomatic Design* [Kulaka et al. 2010],
- *Business reference model* [Choi et al. 2005],
- Etc.

Dans notre travail, nous avons choisi d'utiliser la méthode d'analyse SADT [Marca et McGowan 1987], mais en apportant des modifications sur le formalisme des modèles internes qui seront représentés par des diagrammes de cas d'utilisation UML (Unified Modeling Language) [John et Muthig 2002].

Les fonctionnalités d'une interface sont décrites par un diagramme de cas d'utilisation. Au fur et à mesure qu'une fonctionnalité est identifiée, il est nécessaire de vérifier son usage pédagogique. Seules les fonctionnalités à critère pédagogique sont représentées sur le modèle. Ensuite, les sous-fonctionnalités sont représentées dans un nouveau diagramme de cas d'utilisation. Ce dernier est émergé ultérieurement au sein du digramme principal.

1 Selon wikipedia 2012 : http://en.wikipedia.org/wiki/Function_model

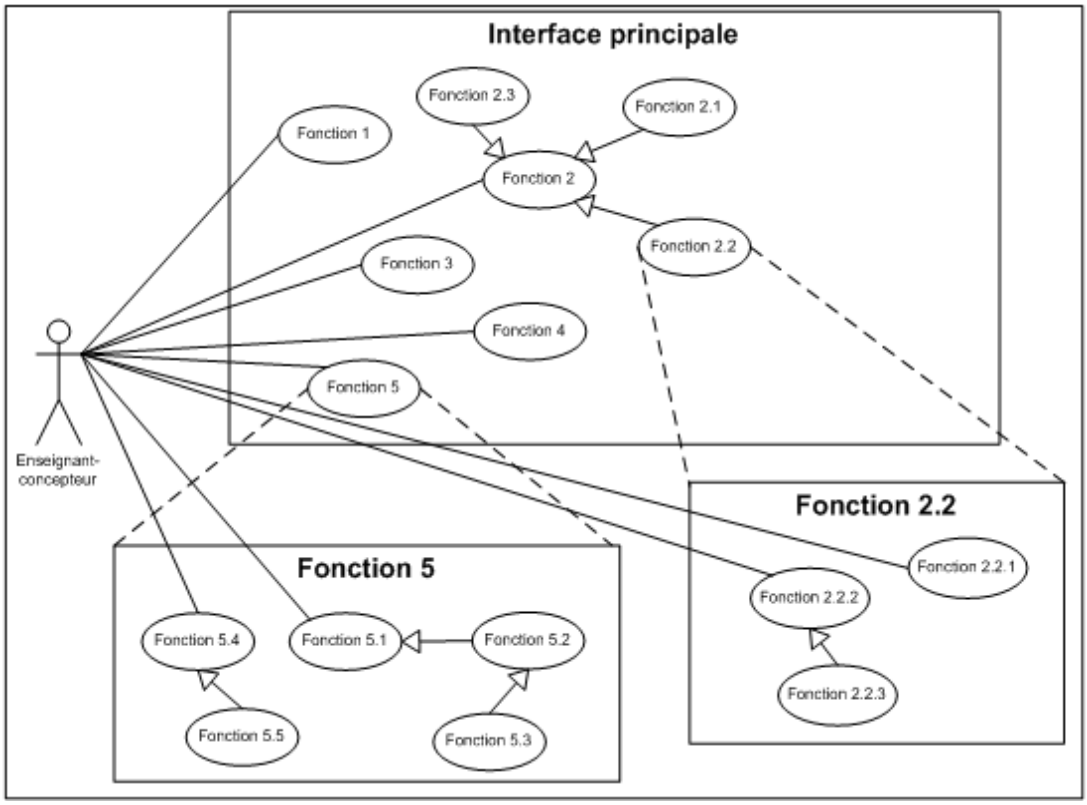


Illustration 12: Vue générale du modèle fonctionnel

La figure 12 illustre un exemple générique du modèle fonctionnel. L'interface principale d'une plateforme est analysée. Ses fonctionnalités pédagogiques sont représentées au sein d'un premier diagramme de cas d'utilisation. Ensuite, les interfaces accessibles à partir d'une de ses fonctionnalités sont également analysées. Les fonctionnalités pédagogiques identifiées sont représentées au sein d'un nouveau diagramme de cas d'utilisation avant d'être émergé au sein du diagramme principal.

5.3 Exemple d'application de l'analyse fonctionnelle sur la plateforme Moodle

Cette section présente brièvement un exemple d'application de l'analyse fonctionnelle sur l'interface principale d'un cours sur la plateforme Moodle.

La figure 13 illustre l'interface de représentation et de spécification du contenu du cours, située dans l'espace d'enseignant-concepteur de la plateforme Moodle. Pour appliquer notre analyse fonctionnelle, nous avons, suivi les étapes suivantes :

1. Dans un premier temps, découper cette interface en quatre zones principales.
2. Ensuite, choisir une zone particulière et appliquer l'analyse fonctionnelle à cette zone : Les zones 1, 3 et 4 sont communes aux interfaces et ne sont analysées

qu'une seule fois.

3. Attribuer une fonction représentative à chaque composant : Plusieurs fonctionnalités sont accessibles par des composants de l'interface graphique tels que les multiples listes déroulantes dans la zone 2 qui disposent de plusieurs éléments et qui représentent les deux fonctionnalités *'ajouter des ressources'* et *'ajouter des activités'*. L'analyse fonctionnelle de ces listes a permis l'identification de nouvelles fonctionnalités en relation telles que *'ajouter une page'*, *'ajouter un url'*, *'ajouter un fichier'*, *'ajouter un forum'*, *'ajouter une leçon'*, etc. Nous avons également identifié d'autres fonctionnalités, représentées par d'autres composants, proposant à titre d'exemple de *'modifier le résumé d'une section'*, *'cacher/afficher une section'*, etc.
4. Identifier les fonctionnalités pédagogiques et les représenter sur le modèle d'analyse fonctionnelle : La figure 14 représente un extrait du modèle fonctionnel résultant de l'analyse fonctionnelle de l'interface du cours sur Moodle.

Le choix d'une fonctionnalité particulière, par exemple *'ajouter un forum'*, parmi les fonctionnalités identifiées nous amène à une nouvelle interface dédiée (au forum dans ce cas). L'analyse fonctionnelle de cette dernière permet l'identification des nouvelles fonctionnalités, telles que *'ajouter une discussion'*, qui sont déjà des sous-fonctionnalités de la fonctionnalité principale (*'ajouter un forum'*). Elles sont représentées par un nouveau diagramme de cas d'utilisation avant d'être associé au diagramme principal (figure 13).

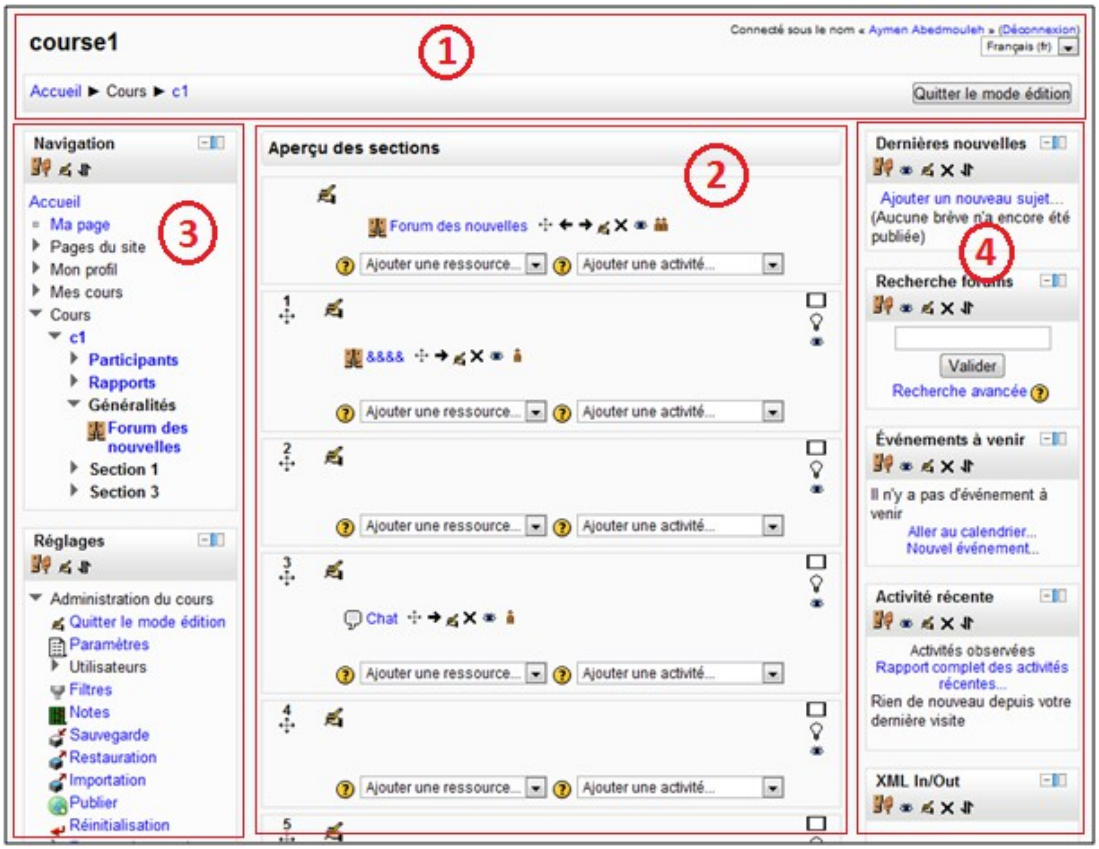


Illustration 13: Interface du cours sur Moodle

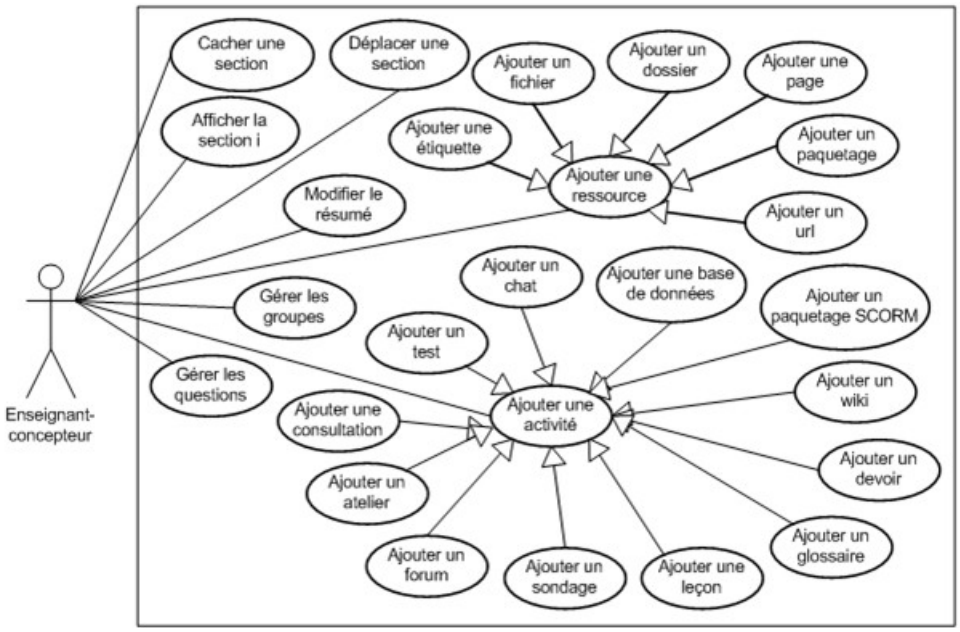


Illustration 14: Extrait du modèle fonctionnel de l'interface de cours sur Moodle

6 Analyse micro

L'analyse micro se base sur les modèles macro et fonctionnel issus des analyses IHM macro et fonctionnelle. Il consiste à identifier les caractéristiques des éléments de conception pédagogique selon deux points de vue différents : point de vue IHM micro et point de vue technique. Nous proposons une confrontation des modèles IHM-micro et technique afin de formaliser le modèle final (Figure 15).

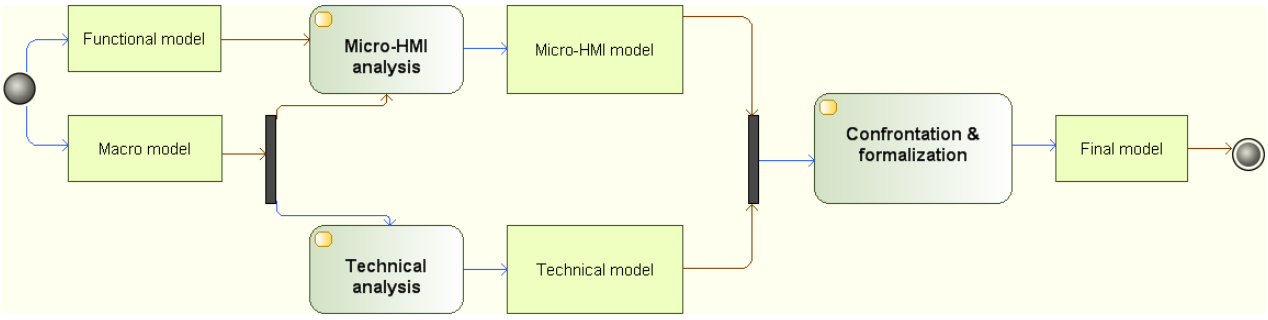


Illustration 15: Analyse micro

6.1 Analyse IHM micro

L'objectif participial de l'analyse IHM micro est d'étudier les interfaces identifiées à une échelle plus fine afin d'identifier les caractéristiques de leurs éléments micro relevant de la conception pédagogique. Nous avons modélisé cette analyse par le biais du digramme d'activité du langage UML (figure 16). Dans cette analyse, nous avons suivi les étapes suivantes :

1. Choisir un élément du modèle macro.
2. Décomposer l'interface de cet élément en plusieurs zones. La décomposition effectuée au sein de l'analyse fonctionnelle peut être adoptée.
3. Écarter les zones ne disposant pas des caractéristiques des éléments de conception pédagogique. Ces zones sont définies pour des finalités techniques ou administratifs plutôt que pédagogiques.
4. Procéder à une analyse plus fine de chaque zone relevant de la conception pédagogique pour identifier les propriétés des éléments concernés.

Les éléments de chaque zone sont modélisés par le biais des composants de l'interface graphique (des formulaires, des graphiques, des icônes, etc.). Ils doivent être testés et analysés afin de déterminer leurs caractéristiques pédagogiques. L'analyse porte également sur l'identification des attributs et des propriétés des éléments identifiés. Lors des expérimentations menées sur les plateformes d'étude, nous avons constaté que certains attributs possèdent des domaines de valeurs spécifiques et des initialisations par défaut. Ces derniers, représentant des caractéristiques importantes de la conception pédagogique d'une plateforme, doivent être identifiés. Enfin, tous les éléments identifiés ainsi que leurs attributs et leurs propriétés doivent être représentés au sein d'un modèle IHM-micro. A noter que cette analyse est itérative dans le sens que les différentes étapes sont appliquées sur tous les éléments du modèle macro.

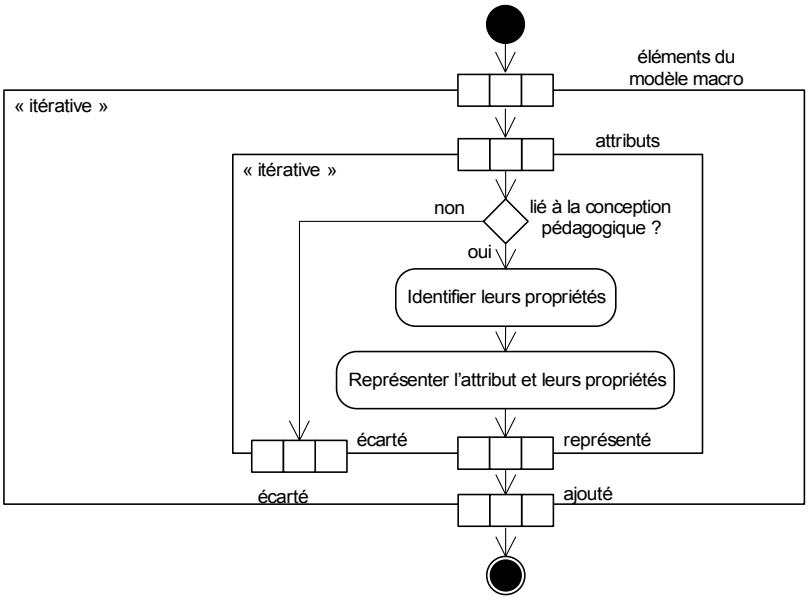


Illustration 16: Diagramme d'activité de l'analyse IHM-micro

Dans les sous-sections suivantes, nous présentons l'analyse fine des zones des interfaces ainsi que le formalisme du modèle IHM-micro.

6.1.1 Analyse fine des zones des interfaces

L'analyse fine des zones des interfaces consiste à analyser les titres, les formulaires, les attributs et les propriétés des éléments.

Analyse des titres

Les parties ou les zones principales des interfaces sont souvent limitées par des cadres et menées par des titres qui soulignent souvent un thème ou un concept spécifique. Ces titres sont généralement spécifiés dans un format spécifique (taille plus grande, souligné, en italique, couleur différente, etc.). Les titres des menus, des blocs, des formulaires, etc. peuvent être également mis en relief. L'analyse sémantique des titres au sein d'une interface ou d'une zone particulière permet l'identification des éléments du langage de conception pédagogique.

Par exemple, dans la plateforme MOODLE, l'interface d'ajout d'un forum (figure 17) est formée de trois zones (encadrées en rouge). Chaque zone est menée d'un ou plusieurs titre(s) spécifié(s) dans un format spécifique (encadrés en rouge et mis en gras à l'entête des cadres). L'analyse des titres a permis d'écarter certains éléments n'intervenant pas à la conception d'un cours tels que *'navigation'*, etc. Cependant, l'analyse du titre *'Ajouter Forum'* de la troisième zone renseigne sur l'élément forum : élément de conception pédagogique d'un cours. Les éléments/attributs dédiés au forum sont ensuite analysés via l'analyse du formulaire proposé. Cette partie sera détaillée au sein de la section suivante.

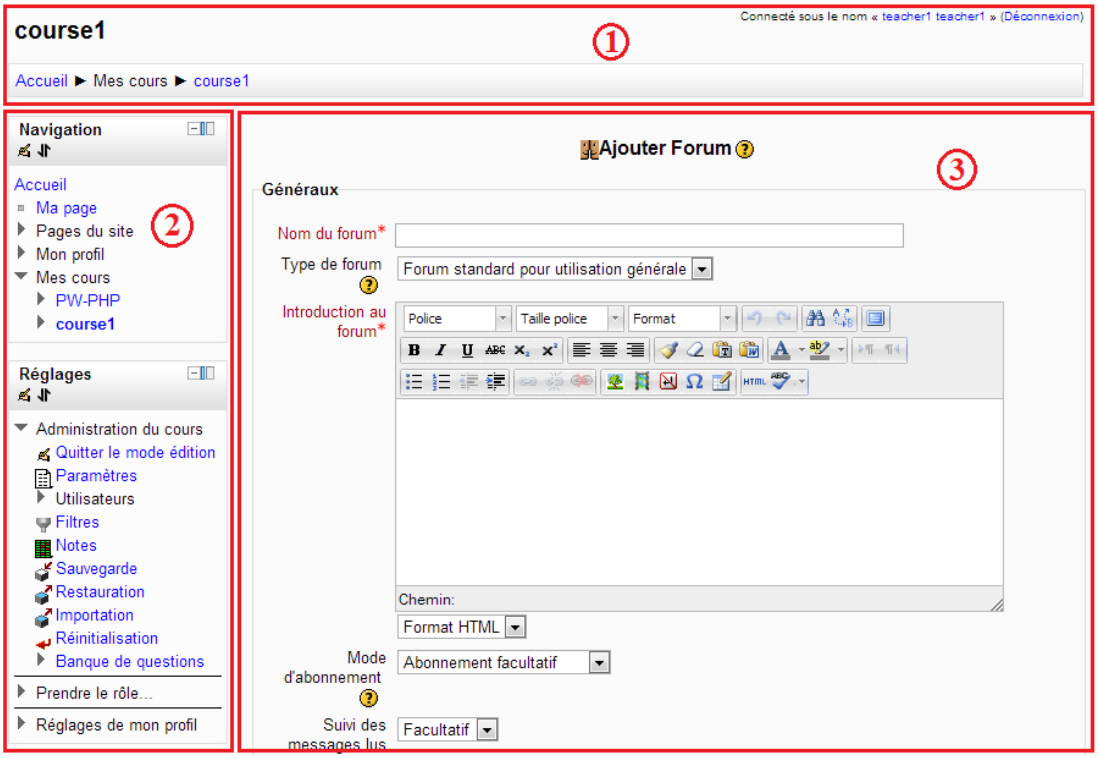


Illustration 17: Interface d'ajout d'un forum

Analyse des formulaires

Basées sur les technologies web, les interfaces des plateformes de formation disposent des formulaires qui permettent de faciliter les interactions entre les utilisateurs et la machine. Ces formulaires décrivent un élément ou un concept en termes d'attributs, des champs de valeurs, des initialisations et des valeurs par défaut, etc.

Comme le montre la figure 17, la zone numéro 3, est composée d'un formulaire spécifique pour l'ajout de l'activité forum. Le formulaire décrit, dans la section 'Généraux', les attributs du forum tels que *'nom du forum'*, *'type du forum'*, *'introduction au forum'*, *'suivi des messages lus dans ce forum'*, etc. Nous avons constaté par le biais des expérimentations que nous avons mené sur plusieurs plateformes que les formulaires disposent généralement de deux catégories d'éléments/attributs :

- des éléments/attributs requis par les interfaces : ils sont obligatoires et primordiaux pour le fonctionnement du système. Ils sont généralement spécifiés dans un format spécifique (colorés différemment, par exemple en rouge), mis dans un style particulier, marqués avec des caractères spéciaux, de font différent (taille plus grande, souligné, italique, etc.). Ils constituent une partie du langage de conception pédagogique d'une plateforme. Par exemple sur l'interface de la figure 17, les éléments requis sont marqués en rouge et par des étoiles '*'.
- des éléments/attributs optionnels ou secondaires : certains sont exploitables pour la conception pédagogique et certains représentent des détails techniques à bas niveau.

Identification des attributs et des propriétés des éléments

L'analyse IHM-micro consiste à identifier les caractéristiques des éléments du langage de conception pédagogique d'une plateforme. Les attributs de ces éléments ainsi que leurs propriétés telles que leurs types doivent être identifiés ; plusieurs types sont possibles tels qu'entier, booléen, chaîne de caractères, une liste de choix, etc. Certains attributs possèdent des champs de valeur spécifiques ainsi que des initialisations par défaut. Ces derniers représentent également une partie du métier de conception pédagogique et doivent être identifiés.

Basées davantage sur les technologies et les techniques favorisant les aspects graphiques, certains éléments, attributs ou propriétés sont désignés par les composants de l'interface graphiques (des widgets, des icônes, des figures, etc.). Souvent, des composants graphiques sont de la même forme, mais ils sont associés à des éléments différents. L'analyse IHM-micro consiste également à identifier les attributs/caractéristiques résidents sous ces composants et à les attribuer aux éléments concernés. Toutefois, certains composants peuvent être écartés parce qu'ils ne sont pas liés à la conception pédagogique ou définis pour des finalités techniques (d'affichage, d'administration, d'organisation, etc.).

Par exemple, la figure 18 illustre l'interface de conception de cours sur la plateforme Moodle où plusieurs éléments/attributs sont représentés par des composants graphiques (widgets). Les widgets (1 et 2) et (4 et 5) ont la même forme, mais ils sont associés à deux éléments différents. Le widget (1) permet d'afficher/cacher un élément du cours (tel qu'un forum ou atelier, etc.). Le widget (2) permet d'afficher/cacher une section entière du cours et également les éléments qu'elle contienne. Le widget (3) permet de déplacer horizontalement un élément . Les widgets (4) et (5) permettent de déplacer (verticalement) respectivement une section ou un de ses éléments. Nous avons tenu compte de ces widgets parce qu'ils interviennent à la conception des éléments du cours.

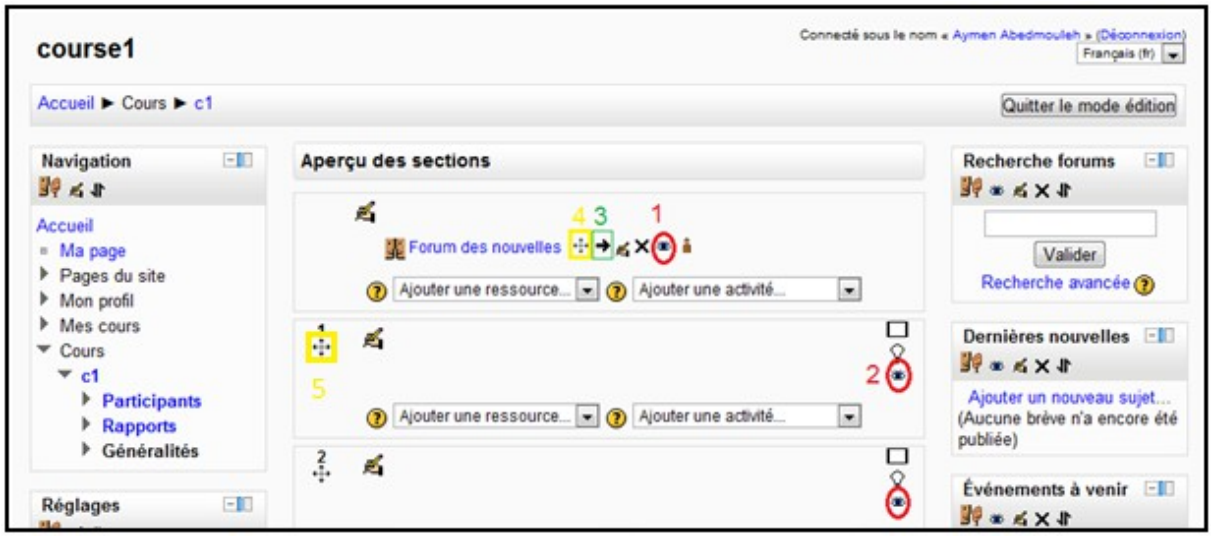


Illustration 18: Widgets graphiques dans une interface

6.1.2 Formalisme du modèle IHM-micro

Pour capitaliser les caractéristiques des éléments de la conception pédagogique au sein des modèles IHM-micro, nous avons choisi le formalisme de diagramme de classe. Ce diagramme vient compléter le diagramme de classe du modèle macro. Il permet la représentation et la description des éléments du modèle macro en termes d'attributs, de types, de propriétés, de domaines de valeur, de contraintes et de relations.

De l'analyse IHM-micro résulte plusieurs modèles dont chacun représente un élément spécifique ainsi que ses attributs et ses caractéristiques. La figure 19 montre un extrait du modèle IHM-micro du concept forum sur la plateforme Moodle.

Une activité/ ressource a plusieurs attributs comme par exemple son ordre dans une section (attribut *A/R order*), sa visibilité (attribut *visible*), sa position (attribut *indent*)... Nous pouvons restreindre la disponibilité d'une activité/resource selon l'achèvement d'une autre activité/ressource (classe *ActivityCompletionCondition*). Nous pouvons aussi restreindre la disponibilité d'une activité / resource selon la note d'une autre activité de type activité notée (Classe *GradeCondition*). Chaque activité a un nom et une description (attributs *name* et *description* de la classe activity). D'autres attributs sont spécifiques pour une activité donnée. Par exemple pour l'activité choix, le concepteur peut choisir qu'elle soit faite en mode groupe par les étudiants (attribut *type: groupmode* de l'activité *choice*). Cette activité peut être composée d'option(s) avec un nombre prédéfini de réponses maximales.

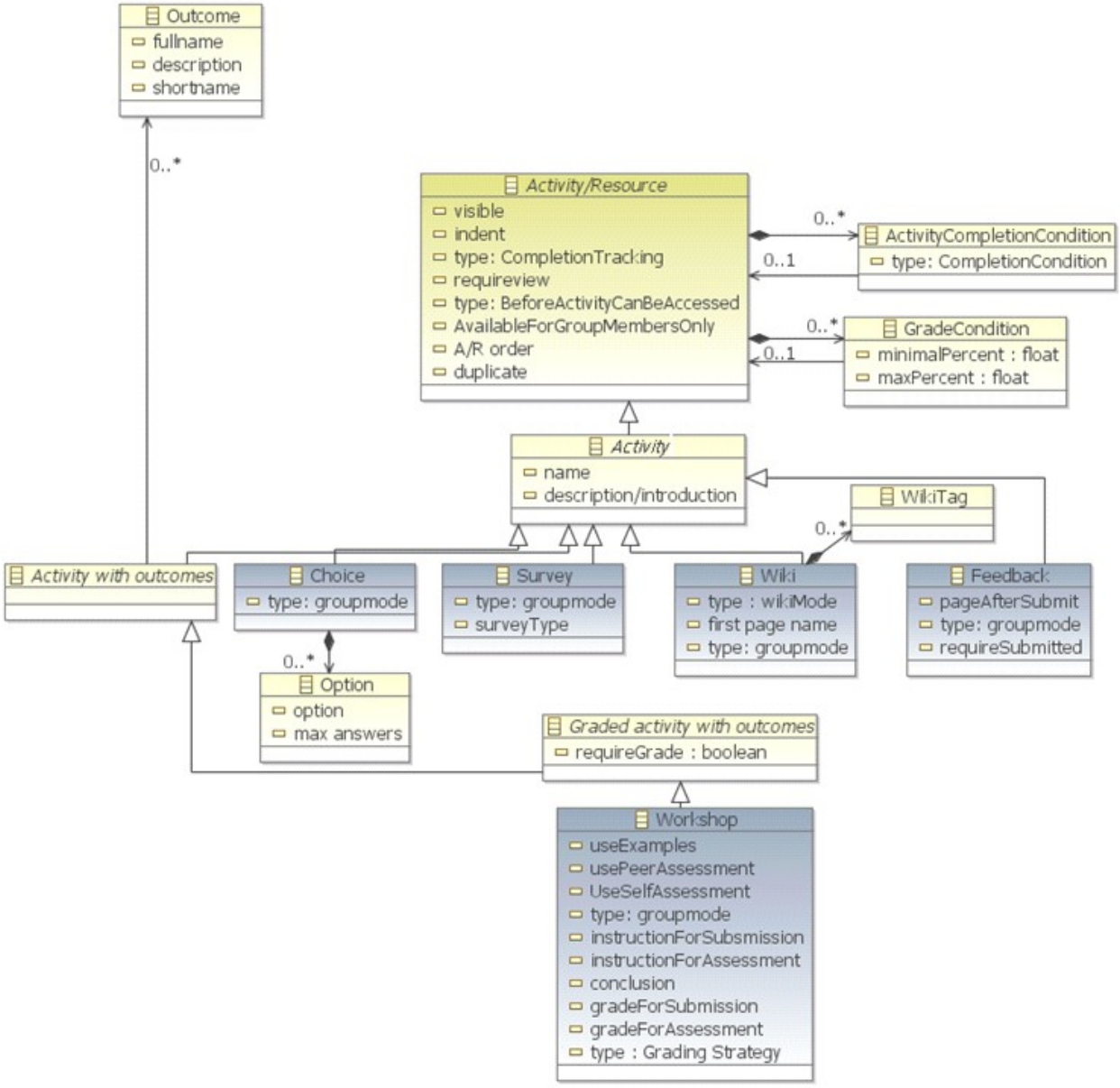


Illustration 19: Extrait du modèle IHM-micro d'un forum de Moodle

6.2 Analyse centrée technique

Cette analyse concerne plusieurs points techniques de la plateforme à étudier : les bases de données, le code source, la sauvegarde/restauration des cours (si elle existe), etc. Nous avons choisi de procéder à une analyse technique pour identifier le langage de conception pédagogique de point de vue technique, en complément au point de vue IHM, afin de valider la persistance des données spécifiques à ce langage. Dans cette étape, la source principale d'information pour identifier le langage de conception pédagogique est la base de données de la plateforme

concernée. Les autres analyses techniques sont réalisées au sein de l'étape de confrontation.

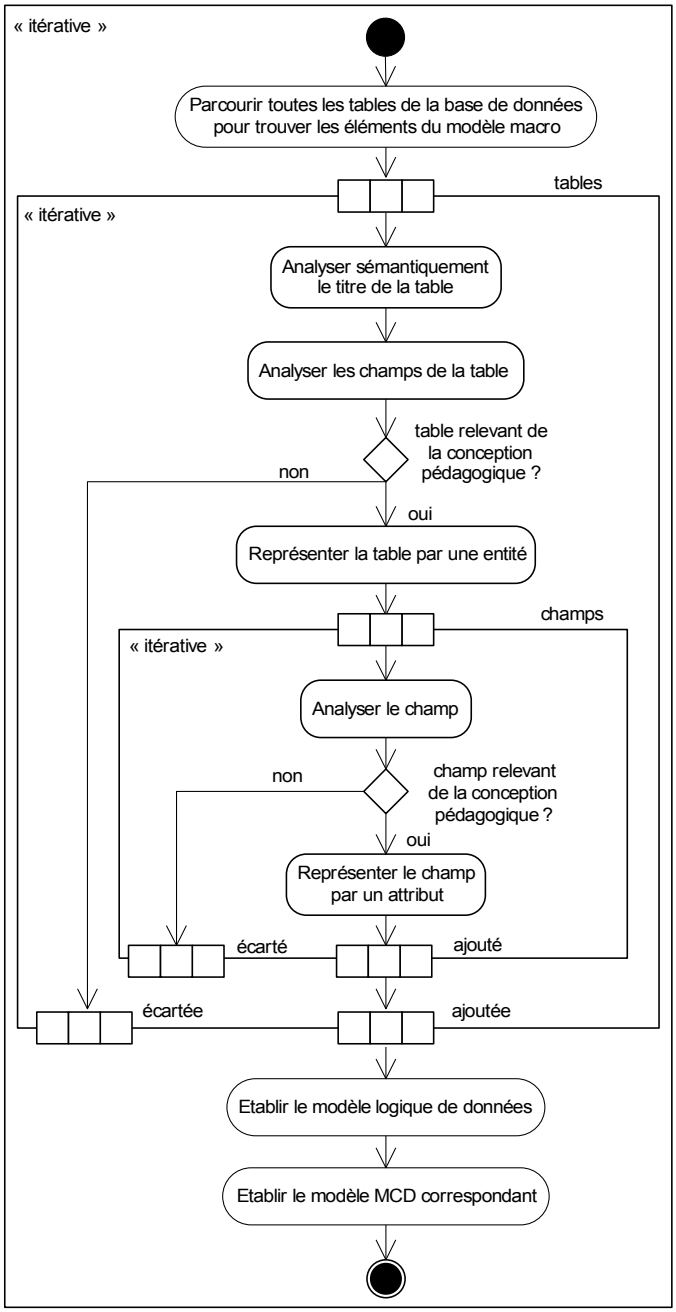


Illustration 20: Diagramme d'activité de l'analyse de la base de données

L'analyse de la base de données correspond en partie à la rétro-ingénierie (*reverse-engineering*) de la base de données.

Nous rappelons que la rétro-ingénierie est un processus d'analyse de la version opérationnelle d'un composant logiciel qui vise à en reconstruire les spécifications techniques et

fonctionnelles [Henrard et al. 1996]. Les objectifs principaux de la rétro-ingénierie sont la ré-documentation, la conversion, la maintenance ou l'évolution d'anciennes applications, ainsi que la modélisation d'un système d'information existant. Cette modélisation décrit les concepts qui ont conduit à l'implémentation de ce système. En fonction du contexte du projet, diverses sources d'information disponibles peuvent être exploitées telles que les sources formelles (les bases de données, les programmes sources de l'application, le dictionnaire de données s'il existe, etc.).

Dans notre approche, l'analyse de la base de données consiste à spécifier le modèle conceptuel de données réduit. Lorsque le modèle conceptuel complet existe déjà ou bien fourni par les créateurs des plateformes, il est possible de l'étudier afin d'identifier la partie relevant de la conception pédagogique. Les parties administratives et techniques doivent être écartées de ce modèle. Plusieurs outils de rétro-conception des bases de données peuvent être utilisés pour générer leurs modèles relationnels de données tels que *MyEclipse, PowerAMC, Win'Design, DBDesigner,* etc. Cependant, la majorité de ces outils ne sont pas gratuits et leurs résultats dépendent directement de la taille de la base de données. Notre approche consiste à restreindre l'analyse de la base de données aux tables/colonnes en relation avec la conception pédagogique. Ces tables sont spécifiques pour la sauvegarde des paramètres et du contenu du cours. Elles représentent aussi les attributs des éléments pédagogiques (activité, ressource, etc.). Sur la base de cette hypothèse, l'analyse a consisté dans un premier temps à la spécification du modèle logique de données réduit. Nous avons constaté que ce modèle ne permettait pas de représenter certains détails et contraintes entre les tables. Pour cela, nous avons choisi de formaliser le modèle technique par le biais du modèle conceptuel de données de la méthode Merise. L'obstacle principal concerne l'identification de ces tables spécifiques. Les informations issues de l'analyse centrée IHM semblent utiles pour ce cas.

Nous avons proposé notre propre méthodologie d'analyse pour capitaliser le langage de conception pédagogique sous la forme d'un modèle conceptuel de données (MCD).

La figure 20 décrit, par le biais du diagramme d'activité UML, les étapes de la méthodologie :

1. Parcourir toutes les tables afin d'esquisser une première ébauche du modèle tout en se focalisant sur les tables référençant des éléments de conception pédagogique.
2. Identifier les tables via l'analyse sémantique de leurs titres ou de leurs champs d'enregistrement. D'autres tables peuvent être également identifiées selon leurs dépendances avec d'autres ou bien selon les clés étrangères proposées. Cependant, certaines plateformes définissent les relations entre certaines tables par des tables spécifiques. Ces tables doivent être identifiées afin de spécifier le modèle logique de données de ces artefacts.
3. Spécifier le schéma réduit de la base de données par le biais des règles de *reverse engineering* des bases de données. Le modèle conceptuel de données peut être finalement spécifié à partir de ce schéma. Ce modèle est important pour représenter le modèle métier selon un point de vue technique. Il permet par construction de cacher les détails techniques de la structure des bases de données et d'éviter les redondances et les mauvaises conceptions. Par exemple, la spécification du modèle technique par le format du modèle conceptuel de données permet d'écarter les tables dédiées à la définition des relations entre d'autres tables.

6.2.1 Parcours des tables

La première étape consiste à parcourir toutes les tables de la base de données de la plateforme. Elle a pour objectif d'avoir une vision globale sur les tables afin d'esquisser une première ébauche du modèle. Elle porte sur la lecture et l'analyse sémantique des titres des tables et de leurs champs. Malgré la diversité des plateformes et des systèmes LMS, il n'existe pas une liste exhaustive des termes et des concepts utilisés en conception pédagogique. Les nominations de certains éléments pédagogiques peuvent être différentes d'une plateforme à une autre. Par exemple, un cours dans Ganesha est composé de séquences alors qu'un cours dans Moodle est composé de sections. Pendant cette étape, des tables spécifiques à des nouveaux concepts pédagogiques, autres que les pré-requis indispensables à l'analyse, peuvent être identifiées.

6.2.2 Identification des tables/champs relavant de la conception pédagogique

La deuxième étape consiste à identifier les tables susceptibles d'avoir des données relevant de la conception pédagogique. L'analyse sémantique des titres des tables, de ses champs/colonnes ainsi que les pré-requis pédagogiques peuvent guider à l'identification de ces tables. Souvent, les tables de la base de données sont nommées par le concept principal qu'elles représentent. Par exemple, les tables de la base de données de Moodle sont nommées soit par l'élément principal de la table (exemple forum, quiz, workshop, etc.), soit par la composée du nom de l'élément principal et d'une spécification de la table telles que les tables *forum_discussion*, *forum_posts*, etc.

Ces tables sont dédiées à la conception pédagogique lorsqu'une partie de leurs titres porte sur un élément de la conception pédagogique. Cette nomination peut aider l'analyste à identifier le concept pédagogique principal de la table ainsi que des relations entre ces tables. Outre les pré-requis pédagogiques que doit posséder l'analyste, d'autres moyens et méthodes permettront d'étendre l'identification des tables spécifiques à des éléments dédiés à la conception pédagogique. Par exemple, les clés étrangères et également les tables spécifiques à la définition des relations (dépendances fonctionnelles) permettent d'indiquer de nouvelles tables liées à la conception pédagogique.

L'identification des tables est plus complexe lorsqu'elles sont intitulées par des nominations non significatives (des titres ou codes spécifiques). Il est ainsi nécessaire d'analyser les enregistrements des tables et souvent d'interpréter l'ajout des éléments pédagogiques à partir des interfaces sur la table. Lors de l'analyse de la base de données, plusieurs tables sont écartées. Elles ne sont ni analysées ni représentées sur le modèle final parce qu'elles ont des finalités techniques plutôt que pédagogiques. Elles sont spécifiques à des usages de gestion, d'administration, etc. Toutefois, nous ne prenons pas en compte tous les champs des tables identifiées. Les champs considérés comme techniques plutôt que pédagogiques doivent être écartés. Par exemple, les champs représentant des détails techniques ou définis à null ou ayant toujours la même valeur sont écartés. Nous prenons les champs portant sur la définition des éléments tels que *name* et *introduction* et les attributs pédagogiques de description des éléments.

6.2.3 Spécification du modèle logique de données réduit de la base de données

La troisième étape consiste à établir le modèle logique réduit de la base de données à partir des tables identifiées dans l'étape précédente. Pour des raisons de simplicité et de clarté, la spécification de ce modèle doit répondre à certaines règles.

- Règle 1 : représenter chaque table par un modèle dont son titre est identique au celui de la table.
- Règle 2 : représenter chaque champ/colonne identifié(e) par un attribut.
- Règle 3 : déterminer les domaines de valeurs des attributs. Les champs définis à "null" ou ayant une valeur par défaut (0 par exemple) peuvent être ignorés.
- Règle 4 : relier les tables qui possèdent des clés étrangères aux tables concernées (celles dont leurs identifiants correspondent aux clés étrangères).
- Règle 5 : relier les tables qui possèdent des identifiants composés aux tables concernées (celles dont leurs identifiants correspondent à une partie de leur identifiant).

6.2.4 Spécification du modèle MCD réduit

La quatrième étape consiste à élaborer / définir ou construire le Modèle Conceptuel de Données réduit de la base de données à partir du modèle logique de données, résultat de l'étape précédente. Cette étape est importante parce qu'elle permet de mettre en évidence les contraintes qui existent entre les entités et de cacher plusieurs détails techniques non liés à la conception pédagogique. Le modèle logique de données de la base de données n'impose pas certaines contraintes existantes dans le monde réel (dont les plus populaires sont les dépendances fonctionnelles et les dépendances d'inclusion). Également, plusieurs tables au sein de ce modèle sont spécifiques seulement à la définition des relations entre d'autres tables. Ces tables, ne portant pas sur la conception pédagogique, seront disparues automatiquement pendant la génération du modèle final (le MCD réduit).

Le modèle conceptuel de données est généré grâce à l'application des règles de transformation de rétro-conception ou *reverse-engineering* sur le modèle logique de données. Ces règles correspondent aux règles de transformation du modèle logique en modèle conceptuel de données définis par la méthode Merise, mais appliquées dans le sens inverse [Gruau 2006] [Panet et al. 1994] [Tardieu et al. 2000]:

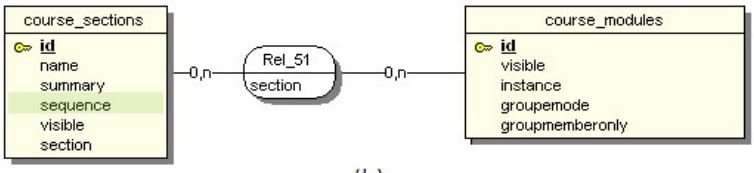
- Règle 1 : chaque table dont la clé primaire ne contient pas de clés étrangères devient une entité dont l'identifiant est la clé primaire de la table et dont les attributs sont les champs de la table à l'exception des clés étrangères.
- Règle 2 : chaque table dont la clé primaire est composée exclusivement de clés étrangères qui référencent une seule clé primaire, devient une sous-entité ou une sous-association (les autres colonnes non clés étrangères de la table deviennent des attributs de cette sous-entité).
- Règle 3 : chaque table dont la clé primaire est composée exclusivement des clés étrangères qui référencent plusieurs clés primaires, devient une association autour de laquelle toutes les cardinalités maximales valent n, c'est-à-dire soit une association binaire

de type (n,m) soit une association ternaire ou plus (les autres colonnes non clés étrangères de la table deviennent des attributs de l'association).

- Règle 4 : chaque table dont la clé primaire est composée partiellement de clés étrangères provient soit d'une optimisation soit d'un identifiant relatif d'une entité (auquel cas les autres colonnes non clés étrangères de la table deviennent des attributs de cette entité).
- Règle 5 : les colonnes clés étrangères restantes deviennent des associations binaires de type (1, n) s'il n'y a pas de contrainte d'unicité ou de type (1,1) s'il y a une contrainte d'unicité (il faut trouver un nom à cette association).
- Règle 6 : la cardinalité minimale vaut 1 pour les clés étrangères qui font partie d'une clé primaire ou qui possèdent une contrainte (non vide), sinon elle vaut 0.
- Règle 7 : Certains attributs doivent changer d'emplacement. Par exemple, l'élément '*sequence*' représentant l'ordre des activités/ressources est une colonne de la table *cours_sections* (Figure 21).

(a)

id	course	section	name	summary	summaryformat	sequence
1	2	0	*NULL*		1	1,17,18
2	2	2	Faire de l'analyse qualitative de texte		1	13,14,52
3	2	3	Faire passer un entretien		1	15,16,55



(b)

Illustration 21: (a) Extrait de la table course_sections, (b) Extrait du modèle logique de données

Nous avons fait le choix de considérer cet élément comme attribut de la classe *course_modules*. Notre métamodèle final va alimenter l'éditeur externe pour la spécification de scénarios en dehors des plateformes, nous pensons que c'est plus pertinent que chaque activité/ressource porte son ordre dans une section (Figure 22).

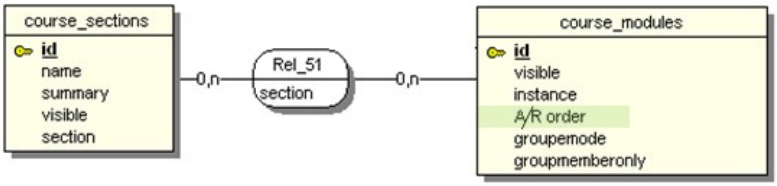


Illustration 22: Extrait du modèle MCD réduit

Dans la section suivante, nous passons à la phase de confrontation et de formalisation du langage de conception.

6.3 Confrontation et formalisation du langage de conception pédagogique final

Les deux analyses centrées IHM micro et techniques mènent à la spécification de deux modèles selon deux points de vue différents et dont leurs représentations ne sont pas identiques. La dernière étape du processus concerne ainsi la confrontation entre ces deux modèles et la formalisation du langage de conception final. Son objectif est de croiser le métier de conception pédagogique manipulé par des enseignants-concepteurs au travers de l'IHM avec le métier de conception persistant stocké dans la base de données afin de parvenir à un modèle unique, correct et représentatif. La confrontation est basée principalement sur la comparaison entre les éléments des deux modèles IHM micro et technique. D'autres méthodes d'analyses techniques (analyse du code sources analyse du modèle des fichiers de sauvegarde, etc.) peuvent être adoptées lorsque des différences ou des ambiguïtés entre les données des modèles sont identifiées.

Dans cette partie, nous allons tout d'abord justifier pourquoi cette phase est importante dans ce processus en se basant sur les inconvénients de chaque méthode d'analyse. Ensuite, nous allons décrire le processus de confrontation et de formalisation. Au cours de ce processus, plusieurs opérations de vérifications peuvent être menées. Elles font l'objet de la section 6.3.2.

Intérêt de la confrontation et de la formalisation

Notre approche d'identification et de formalisation du langage de conception pédagogique porte sur l'exploitation de plusieurs méthodes d'analyse pour plusieurs raisons. Chaque méthode possède ses propres avantages et inconvénients. Par exemple, l'analyse IHM permet l'identification du langage de spécification des situations d'apprentissage visible de point de vue enseignant-concepteur. Cependant, elle dépend de la perception humaine et de la capacité d'interprétation. L'utilisation de plusieurs méthodes permet d'aboutir à des résultats similaires, complémentaires et parfois différents. Certaines méthodes peuvent cacher/montrer certains concepts, relations, contraintes, etc. Certains éléments sont définis par les développeurs de manière implicite. Dans cette section, nous justifions davantage notre choix consistant à :

1. utiliser plusieurs méthodes d'analyse (analyses IHM micro et technique)
2. mener une étape de confrontation entre les modèles résultant afin de détecter certaines erreurs.

Ce choix se justifie par le biais de l'hypothèse suivante : l'utilisation d'une seule méthode d'analyse peut relever plusieurs points négatifs. Nous avons identifié certains inconvénients pour les analyses IHM et techniques (base de données, fichiers de sauvegarde de cours, codes sources).

Inconvénients issus de l'analyse IHM micro

- Le langage identifié dépend directement de la compétence de l'analyste (sa capacité d'identifier les éléments, les relations, les contraintes, etc.).
- Risque de manque d'attributs (non identifiés pendant l'analyse).
- Le langage identifié ne garantit pas la persistance des futurs modèles opérationnels.

Inconvénients issus de l'analyse technique, en particulier l'analyse de la base de données

- Plusieurs contraintes et structures de données ne sont pas déclarées explicitement lors de la création de la base de données parce qu'elles sont définies dans le code.
- Plusieurs éléments peuvent être définis différemment dans la base de données par rapport à leurs affichages dans l'interface.
- L'identification de certaines tables ou certains attributs, représentés(es) par des codes spécifiques, peuvent être très complexes.
- La base de données ne montre pas tous les détails du savoir-faire du système (en termes de conception pédagogique par exemple).
- Les modèles issus de l'analyse de la base de données ou bien de la rétro-conception de la base de données ne sont pas capables de modéliser la sémantique de la plateforme pendant son utilisation.

Inconvénients issus de l'analyse des autres méthodes techniques

Analyse des fichiers de sauvegarde d'un cours

- Plusieurs concepts/fonctionnalités ne sont pas toujours existants(es).
- Le format du fichier de sauvegarde est souvent non éditable, non consultable, etc.
- Les éléments de la sauvegarde sont réduits aux concepts utilisés dans le cours.
- Plusieurs détails peuvent être écartés : tout dépend du contenu et de la configuration du cours et des paramètres de la sauvegarde.

Analyse des codes sources

- L'analyse du code source nécessite de bonnes compétences en informatique telle que la maîtrise des langages de programmation utilisés.
- La compréhension et l'interprétation des codes sources ne sont pas une tâche aisée.

6.3.1 Description du processus de confrontation

La phase de confrontation consiste à comparer les éléments des modèles IHM micro et technique afin de spécifier le modèle métier final du système en termes de conception pédagogique, en particulier le métamodèle du langage de conception pédagogique de la plateforme concernée. Ces deux modèles (IHM micro et technique) sont confrontés afin d'assurer la bonne formalisation du langage de conception pédagogique et sa cohérence globale. Notre démarche consiste à mener des vérifications concernant les éléments des deux modèles et portant sur plusieurs points tels que la définition d'éléments similaires, la non-existence de certains attributs, la divergence des types d'attributs, etc.

Nous avons modélisé le processus de confrontation par le biais d'un diagramme d'activité (figure 23). Le processus considère chaque élément du modèle macro et le présente par un méta-classe dans le métamodèle. Il identifie ses attributs sur le modèle technique. Ensuite, il vérifie l'existence de ces attributs et leurs types dans le modèle IHM. En cas de non-existence d'un

attribut, la méthode d'analyse du code source est adoptée afin de prendre en compte ou de rejeter cet attribut. L'analyse du code révèle plusieurs détails que le développeur a choisi d'implémenter pour des raisons de portabilité et d'efficacité. Elle concerne principalement le code de la représentation des données sur les IHMs concernées et les requêtes d'interrogation de la base de données, en particulier les requêtes d'insertion/sélection des données. En cas de différence de type d'un attribut, la méthode d'analyse du code source est adoptée afin de vérifier le type. Le résultat de cette analyse est généralement conforme à un des résultats des analyses antécédentes. Le type définitif de cet élément peut être ainsi déterminé.

Lorsqu'un attribut et son type sont bien définis, il est modélisé par un méta-attribut dans le métamodèle final. Nous avons choisi pour la nomination de la méta-classe le nom de l'élément du modèle technique. Ce choix stratégique répond aux exigences du métamodèle utilisé pour le développement de nouveaux outils et langages. Les futurs modèles spécifiés par ces outils seront opérationnalisables sur les artefacts concernés. Enfin, les relations entre les méta-classes doivent être définies en se basant sur les relations existantes dans les modèles IHM et techniques. Ensuite, elles sont représentées sur le métamodèle final. La figure 23 illustre les étapes de confrontation et de formalisation du métamodèle du langage de conception pédagogique.

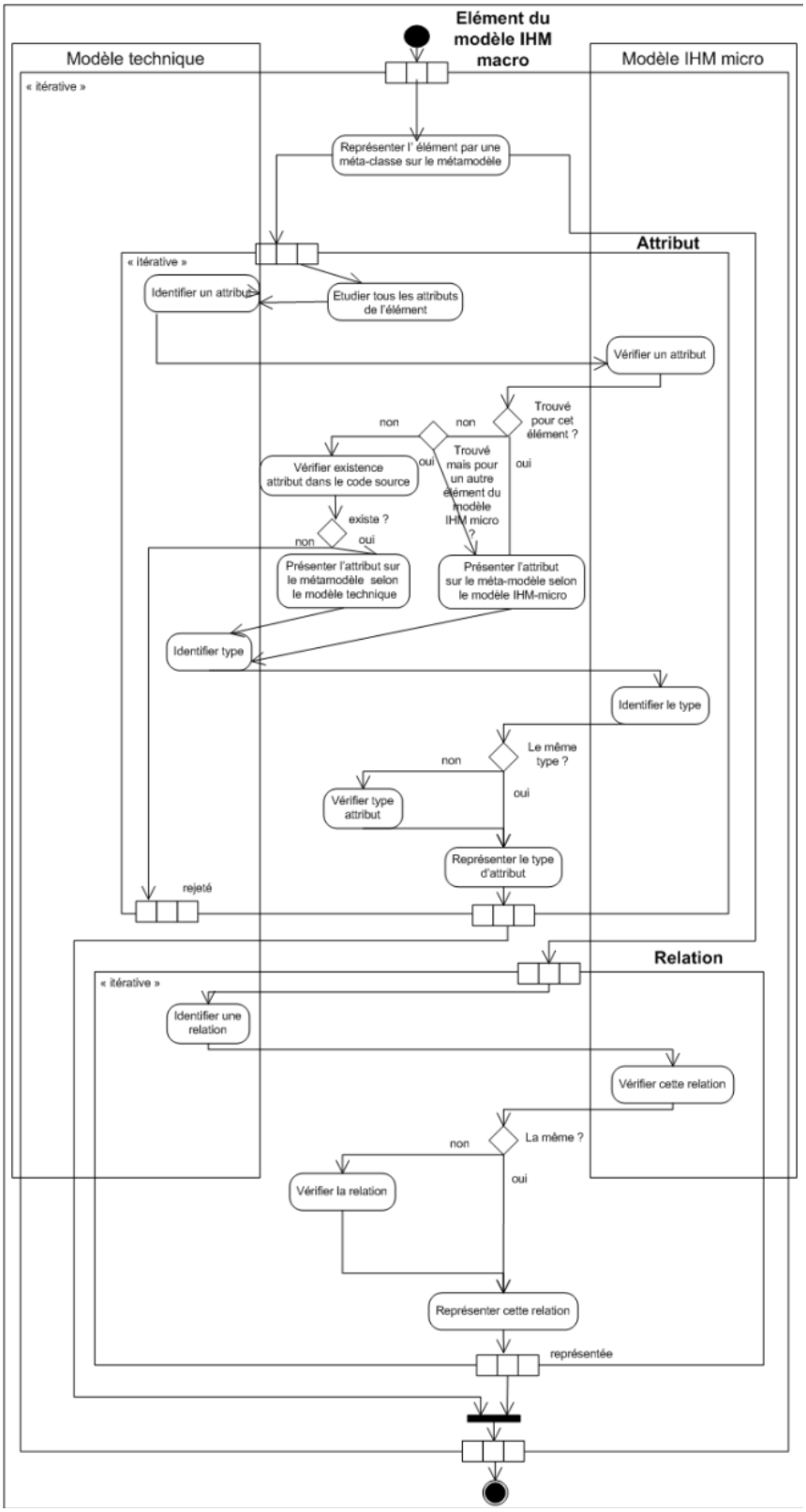


Illustration 23: Diagramme d'activité de la confrontation et formalisation

6.3.2 Processus de vérification

Le processus de vérification consiste à vérifier l'existence d'un attribut, son type et les relations entre les éléments pour l'ajouter ou l'écarter du métamodèle final.

Vérification de l'existence d'un attribut

En ce qui concerne la vérification de l'existence d'un attribut d'un élément, nous considérons que l'attribut est identifié selon les deux méthodes d'analyse centrées IHM micro et technique. Cependant, certains attributs ne sont pas identifiés via la méthode IHM micro. Des vérifications, restreintes à cet attribut, sont ainsi nécessaires.

Le processus de vérification consiste à utiliser une nouvelle méthode d'analyse, il s'agit de l'analyse du code source. Cette analyse consiste à étudier les instructions de définition de l'élément sur l'IHM ainsi que le code d'interrogation des tables de la base de données dédiées à cet élément. Plus spécifiquement, nous analysons les instructions (les requêtes), d'insertion, de mise à jour ou de sélection de l'attribut concerné. Ces instructions peuvent expliquer davantage certaines transformations, codées par le développeur, pendant l'interrogation de la base de données. Le résultat de cette analyse est comparé avec le résultat de l'analyse technique précédant afin de déterminer la prise en compte de cet attribut. La figure 24 illustre un exemple de processus de vérification de l'attribut « sectionOrder » de l'élément « Section » pour la plateforme MOODLE. En effet, cet attribut est identifié dans le modèle technique, mais il n'est pas trouvé au niveau du modèle IHM micro. Pour cela, l'analyse du code source est nécessaire pour vérifier l'existence de l'attribut. L'analyse du code source du fichier course/editsection.php prouve la présence de cet attribut. Donc « sectionOrder » sera présenté comme attribut de l'élément « Section » dans le modèle final (Figure 25).

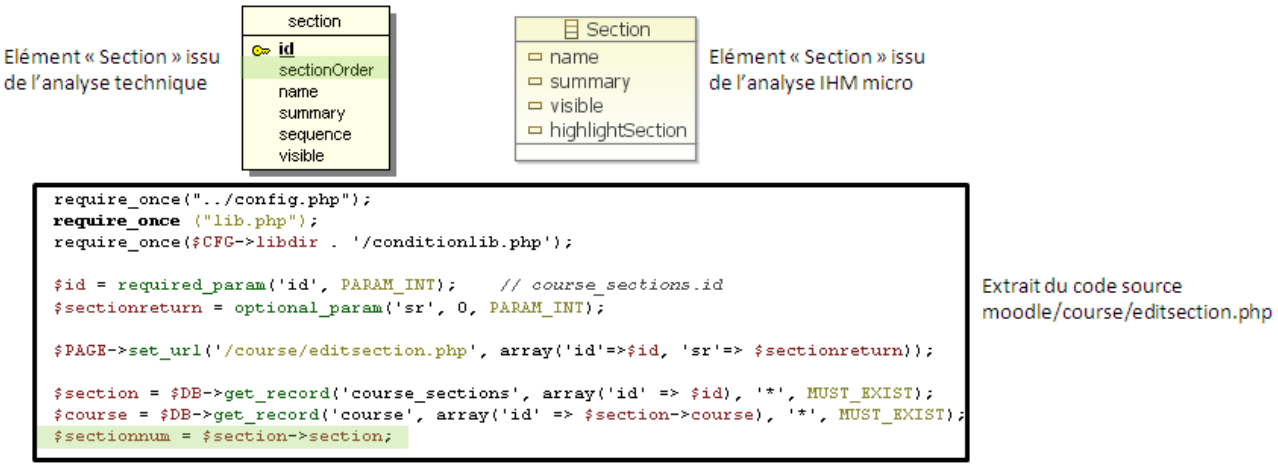


Illustration 24: Exemple de la vérification de l'attribut "sectionOrder" de l'élément "Section"

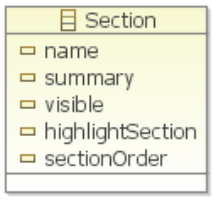


Illustration 25: Présentation de l'attribut "sectionOrder" de l'élément "Section" dans le modèle final

Vérification du type d'un attribut

En ce qui concerne la vérification du type d'un élément, nous considérons que l'attribut est déjà identifié dans les deux modèles IHM micro et technique. Cependant, son type est défini différemment. Lorsque les deux analyses mènent à deux types différents, le processus de vérification doit procéder par une autre méthode d'analyse. Nous choisissons l'analyse du code source comme méthode d'analyse complémentaire.

Similairement au cas de vérification de l'existence d'un attribut, l'analyse du code source consiste à analyser le code de définition de l'élément sur l'IHM ainsi que le code d'interrogation des tables de la base de données dédiées à cet élément, en termes d'insertion, de mise à jour ou de sélection. Certaines transformations, codées par le développeur, peuvent être identifiées pendant (1) son enregistrement dans la base de données ou (2) son affichage sur l'interface associée. Le résultat de cette analyse est généralement conforme à un des résultats des analyses antécédentes. Le type définitif de cet élément peut être ainsi déterminé.

La figure 26 illustre le processus de vérification du type d'un élément en adoptant la méthode d'analyse du code source comme méthode complémentaire. Par exemple dans la plate-forme MOODLE, les attributs « requirePosts », « requireDiscussions » et « requireReplies » de l'élément « Forum » ont les deux types « booléen » et « entier » selon l'analyse IHM micro. Alors que selon

l'analyse technique, ces attributs sont de type « entier » uniquement. Pour cela, l'analyse du code source est nécessaire afin de déterminer le type. L'analyse du code source du fichier forum/lib.php montre que ces attributs sont de type « entier ». Par conséquent, ces attributs sont présentés dans le modèle final avec un type « entier » (Figure 27).

Illustration 26: Exemple de vérification des types des attributs "requirePosts", "requireDiscussions" et "requireReplies" de l'élément "Forum"

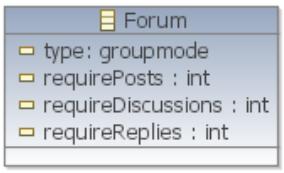


Illustration 27: Présentation des types des attributs "requirePosts", "requireDiscussions" et "requireReplies" de l'élément "Forum" dans le modèle final

Vérification d'une relation

La relation entre deux éléments particuliers peut-être différemment définit sur les deux modèles IHM micro et technique. Cette particularité nécessite l'analyse du code source pour leurs vérifications. Nous analysons le code source d'interrogation des tables de ces éléments et des tables en relation (si elles existent) et également les instructions spécifiques à l'ajout de ces éléments et la définition des relations entre eux.

Les instructions du code source peuvent exprimer certains détails, codés implicitement par le développeur. Le résultat de cette analyse est généralement conforme à un des résultats précédents. La prise en compte des multiplicités finales peut être ainsi déterminée. La figure 28 illustre, dans la plateforme Moodle, le processus de vérification de la relation d'association par référence entre les deux éléments « Activity/Resource » et « GradeCondition ». Cette relation n'est pas la même au niveau du modèle IHM micro et le modèle technique. L'analyse IHM micro montre que « GradeCondition » fait référence à « Activity/Resource » alors qu'en se basant sur le modèle technique, « GradeCondition » fait référence à une activité notée. L'analyse du code source est nécessaire pour vérifier cette relation. L'analyse du code source du fichier lib/conditionlib.php montre que « GradeCondition » fait référence à une activité notée. Par conséquent, cette relation de référence est présentée entre « GradeCondition » et une activité notée dans le modèle final (figure 29).

Illustration 28: Exemple de vérification de la relation entre les éléments "Activity/Resource" et "GradeCondition"

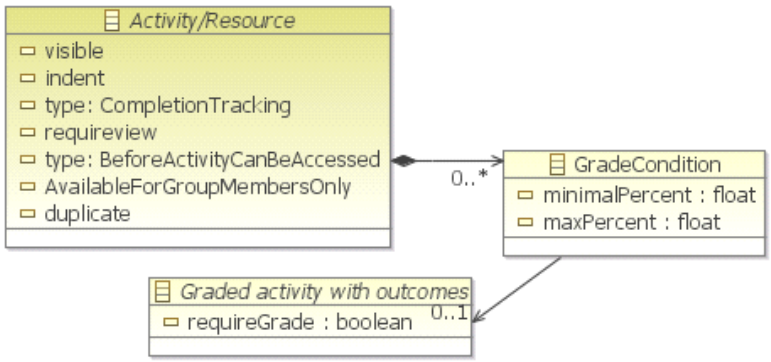


Illustration 29: Présentation de la relation de référence entre "GradeCondition" et "Graded activity with outcomes" dans le modèle final

7 Conclusion

Dans ce document, nous avons présenté notre processus d'analyse, d'identification et de formalisation du langage de conception pédagogique d'une plateforme de formation.

L'objectif de ce processus est de formaliser ce langage à partir de trois points de vue différents et complémentaires : le point de vue IHM macro qui assure la partie du langage visible pour un utilisateur, le point de vue fonctionnel qui porte sur l'dentification des fonctions déjà existantes sur le système et l'analyse micro du langage de conception pédagogique de la plateforme.

Les usagers de ce processus (analystes, ingénieurs de recherche ou pédagogiques, experts des plateformes, etc.) doivent avoir certaines compétences informatiques. La spécification du modèle technique (modèle conceptuel de données) n'est pas une tâche aisée. Elle nécessite (1) l'identification des tables relevant de la conception pédagogique d'une plateforme, (2) la génération du modèle logique de données et (3) la génération du modèle conceptuel de données par application des règles de rétro-conception. Au cours de cette analyse, plusieurs difficultés techniques peuvent être rencontrées telles que l'identification des clés étrangères.

Le processus d'identification et de formalisation présenté dans ce document se base sur les travaux de thèse d'Abedmouleh [Abedmouleh et al. 2012C]. Dans sa proposition d'identification, nous avons remarqué :

1. Les éléments du métamodèle final présentent des attributs en commun : nous pouvons attribuer des objectifs et/ou des notes à certaines activités, pour cela nous avons ajouté l'étape de factorisation dans le processus.
2. Le métamodèle final ne prend pas en considération des paramètres qui nous semble pédagogiquement pertinente comme la restriction de disponibilité d'une activité/ressource selon l'achèvement et/ou la note dans une autre activité/ressource, l'achèvement d'une activité, les objectifs d'une activité, la possibilité de noter certaines activités. Pour cela nous avons étudié la plateforme Moodle d'une façon plus fine afin d'ajouter les éléments reflétant de la conception pédagogique.
3. Le processus de confrontation et de formalisation manque de clarté et de précision, nous avons retravaillé cette partie afin de bien préciser quel est l'élément à vérifier et comment procéder à sa vérification.

Dans le livrable D4.2, nous allons expérimenter ce processus sur les quatre plateformes de formation Moodle (2.0 et 2.4), Ganesha, Dokeos et Sakai.

8 Références

Abdallah F. (2009). Méta-modélisation pour décrire et instrumenter une situation d'apprentissage de Pédagogie par Projet Collectif. Doctorat de l'Université du Maine.

Abedmouleh A. (2012, 23-24 mai 2012). Formalisation du langage de conception pédagogique implicite d'un LMS : motivations et processus. In : Quatrièmes Rencontres Jeunes Chercheurs en EIAH, Amiens, France, p. 3-9.

Abedmouleh A., Laforcade P., Oubahssi L. et Choquet C. (2012 a, 4-6 juillet 2012). Identification of LMSs Instuctional Languages: an Analysis Process. In: IEEE International Conference on Advanced Learning Technologies, Rome, Italie, p. 367-368.

Abedmouleh A., Oubahssi L., Laforcade P. et Choquet C. (2012 b, 25-27 juin 2012). Expressing the implicit instructional design language embedded in an LMS: motivations and process. In: Computers and Advanced Technology in Education, Naples, Italie, p. 774-064.

Abedmouleh A., Oubahssi L., Laforcade P. et Choquet C. (2012 c, 24-27 juillet 2012). An analysis process for identifying and formalizing LMS instructional language. In: International Conference on Software and Data Technologies, Rome, Italie, p. 218-223.

Boehm B.W. (2007). Software design and structuring. In: Software engineering, lifetime contributions to software development, management and research, edition 9 Addison-Wesley, 13 mars 2010, 792 pages, ISBN : 978-0137035151.

Caron P.A. (2007). Bricoles : une approche dispositive des applications web 2.0 utilisables pour enseigner. In: Actes de la conférence Environnements Informatiques pour l'Apprentissage Humain (EIAH 2007), Lausanne, Suisse, 27-29 Juin 2007, p. 137–142.

Choi Y., Kim k. et Kim C. (2005). A design chain collaboration framework using reference models. In: the International Journal of Advanced Manufacturing Technology, Vol. 26, issue 1, p. 183-190.

Gruau C. (2006). Conception d'une base de données. Document technique. Accessible à : ftp://ftp-developpez.com/cyril-gruau/ConceptionBD.pdf (consulté en janvier 2013).

Henrard J., Hick J.M., Roland D., Englebert V. et Hainaut J.L. (1996, 4-7 juin 1996). Techniques d'analyse de programmes pour la rétro-ingénierie de bases de données. In: INFORSID'96, Bordeaux, France, p. 215-232.

John I. et Muthig D. (2002). Tailoring Use Cases for Product Line Modeling. In : International Workshop on Requirements Engineering for Product Lines (REPL), Essen, Germany. ISBN: 0-9724277-0-8.

Kulaka O., Cebib S. et Kahraman C. (2010, september 2010). Applications of axiomatic design principles: A literature review. In: Expert Systems with Applications, vol. 37, issue 9, p. 6705–6717.

Marca D.A., et McGowan C.L. (1987). SADT: structured analysis and design technique, New York, USA, ISBN:0-07-040235-3.

Muehlen M.Z. et Recker J. (2008, June 16-20 2008). How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In: the 20th international conference on Advanced Information Systems Engineering (CAISE 2008), Montpellier, France, p. 465-479.

Oubahssi L. (2005). Conception de plates-formes logicielles pour la formation à distance, présentant des propriétés d'adaptabilité à différentes catégories d'usagers et d'interopérabilité avec d'autres environnements logiciels. Doctorat de l'Université René Descartes – Paris V.

Panet G., Letouche R. et Tardieu H. (1994). Merise 2 : Modèles et techniques Merise avancés. 360 pages. ISBN : 2708116533. Edité par Organisation Eds D'.

Ryu K. et Yücesan E. (2007). CPM: A collaborative process modelingfor cooperative manufacturer. In: Journal Advanced Engineering Informatics, vol. 21, issue 2, avril 2007, p. 231–239.

Tardieu H., Rochfeld A. et Coletti R. (2000). La méthode Merise. Principes et outils Edition 2000. 344 pages. ISBN : 2708124730 Edité par Organisation Eds D', juillet 2000.

Tchounikine P., Baker M., Balacheff N., Baron M., Derycke A., Guin D., Nicaud J.F. et Rabardel P. (2004). Platon-1: quelques dimensions pour l'analyse des travaux de recherche en conception d'EIAH. Rapport de l'Action Spécifique du département STIC du CNRS : Fondements théoriques et méthodologiques de la

conception des EIAH. Accessible à : http://hal.archives-ouvertes.fr/docs/00/02/69/65/PDF/Platon-1.pdf, (consulté en février 2014).