Approche basée sur des patrons pour concevoir des logiciels d'enseignement adaptés aux technologies du Web

Thèse

Tonguim Ferdinand GUINKO

Doctorat en informatique
Philosophiæ doctor (Ph.D.)

Québec, Canada

Approche basée sur des patrons pour concevoir des logiciels d'enseignement adaptés aux technologies du Web

Thèse

Tonguim Ferdinand GUINKO

Sous la direction de:

Laurence Capus, Directrice de recherche
Hatem Ben Sta, Co-directeur de recherche

Résumé

Les applications Web en général ont connu d'importantes évolutions technologiques au cours des deux dernières décennies et avec elles les habitudes et les attentes de la génération de femmes et d'hommes dite *numérique*. Paradoxalement à ces bouleversements technologiques et comportementaux, les logiciels d'enseignement et d'apprentissage (LEA) n'ont pas tout à fait suivi la même courbe d'évolution technologique. En effet, leur modèle de conception est demeuré si statique que leur utilité pédagogique est remise en cause par les experts en pédagogie selon lesquels les LEA actuels ne tiennent pas suffisamment compte des aspects théoriques pédagogiques. Mais comment améliorer la prise en compte de ces aspects dans le processus de conception des LEA ? Plusieurs approches permettent de concevoir des LEA robustes. Cependant, un intérêt particulier existe pour l'utilisation du concept *patron* dans ce processus de conception tant par les experts en pédagogie que par les experts en génie logiciel. En effet, ce concept permet de capitaliser l'expérience des experts et permet aussi de simplifier de belle manière le processus de conception et de ce fait son coût. Une comparaison des travaux utilisant des patrons pour concevoir des LEA a montré qu'il n'existe pas de cadre de synergie entre les différents acteurs de l'équipe de conception, les experts en pédagogie d'un côté et les experts en génie logiciel de l'autre. De plus, les cycles de vie proposés dans ces travaux ne sont pas complets, ni rigoureusement décrits afin de permettre de développer des LEA efficients. Enfin, les travaux comparés ne montrent pas comment faire coexister les exigences pédagogiques avec les exigences logicielles. Le concept patron peut-il aider à construire des LEA robustes satisfaisant aux exigences pédagogiques ? Comme solution, cette thèse propose une approche de conception basée sur des patrons pour concevoir des LEA adaptés aux technologies du Web. Plus spécifiquement, l'approche méthodique proposée montre quelles doivent être les étapes séquentielles à prévoir pour concevoir un LEA répondant aux exigences pédagogiques. De plus, un répertoire est présenté et contient 110 patrons recensés et organisés en paquetages. Ces patrons peuvent être facilement retrouvés à l'aide du guide de recherche décrit pour être utilisés dans le processus de conception. L'approche de conception a été validée avec deux exemples d'application, permettant de conclure d'une part que l'approche de conception des LEA est réaliste et d'autre part que les patrons sont bien valides et fonctionnels. L'approche de conception de LEA proposée est originale et se démarque de celles que l'on trouve dans la littérature car elle est entièrement basée sur le concept patron. L'approche permet égale-

ment de prendre en compte les exigences pédagogiques. Elle est générique car indépendante de toute plateforme logicielle ou matérielle. Toutefois, le processus de traduction des exigences pédagogiques n'est pas encore très intuitif, ni très linéaire. D'autres travaux doivent être réalisés pour compléter les résultats obtenus afin de pouvoir traduire en artéfacts exploitables par les ingénieurs logiciels les exigences pédagogiques les plus complexes et les plus abstraites. Pour la suite de cette thèse, une instanciation des patrons proposés serait intéressante ainsi que la définition d'un métamodèle basé sur des patrons qui pourrait permettre la spécification d'un langage de modélisation typique des LEA. L'ajout de patrons permettant d'ajouter une couche sémantique au niveau des LEA pourrait être envisagée. Cette couche sémantique permettra non seulement d'adapter les scénarios pédagogiques, mais aussi d'automatiser le processus d'adaptation au besoin d'un apprenant en particulier. Il peut être aussi envisagé la transformation des patrons proposés en ontologies pouvant permettre de faciliter l'évaluation des connaissances de l'apprenant, de lui communiquer des informations structurées et utiles pour son apprentissage et correspondant à son besoin d'apprentissage.

Abstract

Web applications in general have experienced significant technological developments over the last two decades and with them the habits and expectations of the generation of men and women called *The Digital Generation*. Paradoxically to these technological and behavioral changes, e-learning software (ELS) does not quite follow the same curve of technological change. Indeed, its design model remained so static that its pedagogical usefulness is questioned by pedagogical experts who say that current ELS does not take sufficient account of educational theory. So how to improve the inclusion of pedagogical requirements in the ELS design process? There are several approaches to designing robust ELS. However the use of *pattern* concept in this design process is of great interest to both educational experts as well as experts in software engineering. The *pattern* concept allows ones to capitalize on the experience of ELS design experts and also simplifies the software design process, thus also reducing the design process cost. A comparison of patterns-based ELS design processes in the litterature has shown that there is no collaboration framework for the ELS design team, that is to say, educational specialists and software engineers. There is also a lack of important steps in the proposed software life cycles which may not be rigorously described to allow the design of efficient ELS. Finally patterns used in the design process of ELS meet either educational or software requirements but not both. As a solution, this thesis proposes a design approach to designing pattern-based ELS suited to Web technologies. More specifically, this thesis primarily proposes a pattern-based systematic approach, showing what should be the sequential steps for designing an ELS that meets pedagogical requirements. Furthermore this thesis also proposes a repository of 110 patterns that are used in the approach. These patterns can easily be found using the pattern search guide proposed in this thesis. The design approach was validated with two application examples to conclude that firstly the ELS design approach is realistic and secondly, that the patterns are valid and functional. The proposed ELS design approach is original and differs from those found in the literature as it is entirely based on the *pattern* concept. The approach also allows the ELS engineer to take into account the educational requirements. It is generic because it is independent of any hardware or software platform. However, the process of educational requirements translation is still not very intuitive nor very straight forward. Further work must be done to complete the results obtained, in order to bring usable artifacts from pedagogical requirements to software engineers. A

pattern-based metamodel for ELS design that will allow the definition of a typical modeling language for ELS design, or the development of a more intelligent method for patterns search in a large directory is also considered. Adding patterns that will help adding a semantic layer at the ELS could be considered. This semantic layer will not only adapt pedagogical scenarios but will also automates the process of adaptation to the needs of a particular student. Finally another futherwork that can be addressed is how the transformation of proposed patterns in ontologies that can help facilitate the assessment learner's knowledge in order to provide him structured and useful information for his learning process.

Table des matières

Liste des tableaux

Liste des figures

Remerciements

Le Drocessus des études doctorales comporte une complexité de facteurs tel qu'il nécessite la direction de personnes avisées. Aussi dans notre cas plusieurs personnes nous ont apportées leur contribution. C'est ainsi que nous tenons à remercier particulièrement nos deux directeurs de thèse, Dre Laurence CAPUS et Dr Ben Sta HATEM qui ont oeuvré sans relâche pour nous accompagner tout au long de ce Drocessus.

Nous remercions également :

— la Dre Nicole TOURIGNY et le Dr Minh Duc BUI membres de notre comité d'encadrement pour leur accompagnement pédagogique ;

— le Drésident et les membres du jury de la soutenance de notre thèse qui sont respectivement Dre Laurence CAPUS, Dr Ben Sta HATEM, Dr Brahim CHAIB-DRAA, Dr Ronald BEAUBRUN, Dr Mamoudou BOUSSO et Dr Minh Duc BUI ;

— la direction du département, et l'ensemble du personnel enseignant, du département d'informatique de l'Université Laval, pour toutes les offres d'auxiliaires d'enseignement mises à la disposition des étudiants du département, dont j'ai beaucoup bénéficié ;

— les professeurs du département d'informatique de l'Université Laval qui ont contribué au succès de ma thèse. Ils ont toujours été disponibles selon leur compétence respective ;

— la Dre Thèrese LAFERRIERE du département des études sur l'enseignement et l'apprentissage de l'Université Laval pour ses apDréciations pertinentes sur les idées et les questions à lui soumises ;

— le Dr Pierre MARCHAND qui, bien qu'étant sous le régime de retraite professionnelle, n'a menagé aucun effort pour nous soutenir ;

— les secrétaires du département d'informatique de l'Université Laval qui assurent le rôle de l'interface administration-étudiants pour permettre une meilleure collaboration entre les deux parties ;

— les étudiants et anciens étudiants de l'Équipe de Recherche en Ingénierie des ConnAissancEs (ERICAE), Imen SALLEM, Fatima Ezzahra CHOUKAIRY, Kenza Sakout ANDALOUSSI, Romaric Yao BONI, François VAUDRIN, pour leur contribution à l'élaboration de notre thèse ;

— le groupe de croissance des étudiants Adventistes de l'Université Laval pour leurs encouragements et leur soutien moral, spirituel et matériel.

Nous tenons enfin à remercier sincèrement nos parents pour leur appui inconditionnel, notre soeur Line et son époux Stéphane, notre frère Noé et son épouse Salomé, ainsi que notre cousin François et son épouse Nadège ainsi notre amie Laurentine.

Chapitre 1

Introduction

Les logiciels d'enseignement et d'apprentissage (LEA) ont suscité un réel engouement dans le milieu éducatif. Certains spécialistes y sont allés de leur enthousiasme estimant que ces logiciels contenaient en eux la motivation nécessaire pour susciter l'innovation pédagogique tant rêvée. D'autres (Seffah & Grogono, 2002; Downes, 2005; Hadjerrouit, 2005, 2007; Wilson *et al.* , 2007; Lu *et al.* , 2015) plus prudents ont vu dans les logiciels d'enseignement actuels, certes, des outils innovateurs qui permettent aux enseignants et aux apprenants de s'affranchir de bien des contraintes, mais qui peuvent aussi intégrer en leur sein, des failles souvent d'origine conceptuelle qui font douter les experts en pédagogie de leur efficacité en matière d'enseignement et d'apprentissage.

Cette thèse tente de répondre au besoin de conception des LEA selon les attentes des experts en pédagogie en proposant une solution de conception intégrant des éléments réutilisables et génériques répondant au besoin d'évolution du modèle de développement des LEA. Les notions clés de réutilisabilité et de généricité rappellent sans aucun doute le concept de *patron*, très souvent utilisé dans le domaine de l'ingénierie pédagogique ou dans le cadre de la conception et du développement des logiciels, qui se retrouvera lui aussi au centre de cette thèse.

Ce chapitre présente un résumé du travail réalisé dans le cadre de cette thèse. Nous présentons d'abord le contexte et les motivations ayant conduit à cette thèse. Nous partons du constat de l'engouement pour les LEA dans les milieux éducatifs en admettant le fait que cet engouement ne doit pas être l'arbre qui cache la forêt. Afin de combler certaines des limites des LEA, nous présentons ensuite les questions de recherche et objectifs qui ont guidé notre recherche. Nous avons énoncé à cet effet deux questions de recherche. Nous expliquons ensuite la méthodologie suivie pour la conduite des travaux réalisés. Enfin nous présentons le processus de validation, les résultats et contributions obtenus puis nous terminons ce chapitre par la présentation de l'organisation de cette thèse.

1.1 Contexte

L'Internet et le Web ont connu des évolutions notables au cours des deux dernières décennies à tel point qu'ils se sont immiscés dans les habitudes comportementales quotidiennes des êtres humains. De nos jours, à l'ère du Web 3.0, la génération de femmes et d'hommes dite *numérique* naît pratiquement, vit, travaille et joue avec Internet (Downes, 2005), autant dire donc que les habitudes et les attentes de la *génération numérique* ont aussi changé.

Paradoxalement à ces bouleversements technologiques et comportementaux, les logiciels d'enseignement et d'apprentissage (LEA) n'ont pas eux, tout à fait suivi la même courbe d'évolution technologique. Certains chercheurs (Downes, 2005; Hadjerrouit, 2005, 2007; Wilson *et al.* , 2007; Lu *et al.* , 2015) affirment que le problème en cause est le modèle de développement technologique des LEA : celui-ci est demeuré si statique que l'utilité pédagogique des LEA est remise en cause. Les chercheurs enfoncent le clou en affirmant que ces logiciels sont en déphasage avec les besoins de la génération dite numérique, ce qui réduit l'efficacité de l'apprentissage par les apprenants. De façon générale, les experts en pédagogie tendent à remettre en cause l'efficacité pédagogique des LEA développés par les experts en génie logiciel tandis que ces derniers tendent à s'accorder sur le fait que les méthodes de développement utilisées pour la conception des LEA ne conviennent pas à la conception des LEA (Seffah & Grogono, 2002; Hadjerrouit, 2007; Kompen *et al.* , 2008). En effet les LEA sont à la croisée de plusieurs domaines d'expertise, c'est-à-dire la pédagogie, le multimédia, les sciences cognitives, l'informatique et le marketing ; en conséquence leur processus de conception doit évoluer pour intégrer d'une part les compétences, et d'autre part les connaissances issues de ces domaines d'expertise.

Dans le cadre de cette thèse, nous nous sommes intéressés de façon exclusive aux LEA compatibles avec le Web, car du fait de leur vulgarité et de leur très grande accessibilité au grand public, nous estimons être en phase avec les chercheurs (Seffah & Grogono, 2002; Downes, 2005; Hadjerrouit, 2005, 2007; Wilson *et al.* , 2007) qui pensent qu'une plus grande attention devrait être portée à ces logiciels, afin d'en accroître l'efficacité tant en matière d'enseignement qu'en matière d'apprentissage. L'expression LEA, ou l'expression plateforme logicielle d'enseignement et d'apprentissage, s'il y a lieu, fait référence à tout logiciel conçu dans une perspective d'aide à l'acquisition de connaissances ou de compétences.

1.2 Motivations

Les LEA ont suscité un réel engouement dans le milieu éducatif. Certains spécialistes y sont allés de leur enthousiasme estimant que ces logiciels contenaient en eux la motivation nécessaire pour susciter l'innovation pédagogique tant rêvée. D'autres (Seffah & Grogono, 2002; Downes, 2005; Hadjerrouit, 2005, 2007; Wilson *et al.* , 2007; Lu *et al.* , 2015) plus prudents, ont vus dans les LEA actuels, certes, des outils innovateurs qui permettent aux enseignants et aux

apprenants de s'affranchir de bien des contraintes qui empêchent parfois les uns d'être des enseignants à plus grande échelle, et les autres de bénéficier de formations adaptées à leur profil, mais qui peuvent aussi intégrer en leur sein, du fait de la manière dont ils ont été conçus, une cause qui peut atténuer sinon remettre en question leur efficacité pédagogique.

La première limite évoquée par la littérature (Hadjerrouit, 2005, 2007) au sujet des LEA actuels est leur manque réel d'adaptation aux contraintes pédagogiques des enseignements. Les LEA actuels ignorent, ou ne tiennent pas suffisamment compte des aspects théoriques pédagogiques, que tout outil technologique destiné à être utilisé à des fins d'enseignement ou d'apprentissage devrait prioritairement intégrer. En effet, les applications Web en général, et les LEA en particulier, sont devenus des outils de plus en plus complexes. Paradoxalement le processus de développement de ces logiciels spécifiques n'a pas évolué. Pendant longtemps, les développeurs d'applications pour l'enseignement se sont contentés de s'inspirer du modèle de développement des logiciels dits classiques, qui ignore simplement les questions d'ordre pédagogique. Il en résulte une pauvre analyse et une mauvaise conception pour des applications devant servir à l'enseignement et à l'apprentissage.

En plus des aspects pédagogiques, le fait que les LEA soient situés à la croisée de plusieurs domaines de savoirs (pédagogie, multimédia, sciences cognitives, informatique, marketing, etc.) nécessite pour eux, à l'inverse des logiciels dits classiques, une approche de conception multidisciplinaire (Seffah & Grogono, 2002; Kompen *et al.* , 2008).

Une autre limite qui vient renforcer la thèse du manque d'adaptation des LEA aux fins d'enseignement et d'apprentissage est le fait que certains types de LEA, en particulier les *systèmes de gestion de l'apprentissage*[1], se sont imposés en tant que *modèles dominants* (Wilson *et al.* , 2007). Or depuis quelques années, d'autres technologies innovantes (pair à pair, weblogs, wikis, réseaux sociaux, etc.) sont apparues, et ont été très bien assimilées par les utilisateurs, potentiels apprenants, qui en font une utilisation personnelle dans leur vie quotidienne. Malgré l'adhésion de spécialistes tel que Stephen Downes encourageant ces nouvelles technologies aux fins d'enseignement et d'apprentissage, ces technologies demeurent marginalisées à cause de l'effet de bord des modèles dits dominants.

Les principales caractéristiques des modèles dominants sont les suivantes (Wilson *et al.* , 2007) :

- Ils se concentrent sur les outils en rapport direct avec l'enseignement des cours (forums de discussion, quiz, etc.).
- La relation enseignant/apprenant est asymétrique, c'est-à-dire que le rapport des pouvoirs entre l'enseignant et l'apprenant sur les fonctionnalités du système est inégal.
- Ils offrent un environnement uniforme d'apprentissage pour les apprenants, ce qui constitue un obstacle à l'idéal de la formation personnalisée.

1. En anglais : Learning Management Systems (LMS)

- Les normes dominantes (SCORM [2], IMS [3], etc.) sont privilégiées au détriment d'autres normes, ce qui illustre le caractère fermé des modèles dominants.

A l'opposé, les modèles alternatifs présentent les caractéristiques suivantes :

- La relation est symétrique entre enseignants et apprenants, c'est-à-dire que le rapport des pouvoirs entre l'enseignant et l'apprenant sur les fonctionnalités du système est également réparti.
- L'offre d'environnement est personnalisé et le contexte est individualisé, pour chaque apprenant.
- Ils sont ouverts aux technologies non standards (googleMaps [4], IETF [5] etc.).

Les caractéristiques des modèles dits alternatifs, comme par exemple la possibilité d'offrir un environnement personnalisé et un contexte individualisé à chaque apprenant, intéressent bien la communauté des experts en pédagogie. Cependant pour des raisons stratégiques ces caractéristiques ne font l'objet que de peu d'intérêt de la part de la communauté des chercheurs qui s'intéressent au processus de conception des LEA.

Une autre insuffisance des LEA est en rapport avec le fait que, malgré les évolutions technologiques du Web, les LEA actuels en sont restés aux standards du Web 1.0 et ne peuvent pas par conséquent porter les atouts avantageux du Web 2.0 (Downes, 2005) voire du Web 3.0. Pourtant les habitudes et comportements des apprenants potentiels actuels, dits de la *génération numérique*, susceptibles d'utiliser ces LEA, ont suivi l'évolution du Web 1.0 vers le Web 2.0. La *génération numérique* est pratiquement *née* avec Internet, a grandi avec, joue et travaille avec Internet [6]. C'est une génération qui, à l'heure du Web 2.0 et des réseaux sociaux électroniques tels que Facebook [7] et Twitter [8], a vu ses comportements et ses habitudes fortement influencés par ces nouvelles technologies.

Il est donc nécessaire que les LEA s'adaptent à ces nouvelles tendances afin d'améliorer l'efficacité de l'apprentissage par les apprenants. Ces changements attendus des LEA vont au-delà, par exemple, de la possibilité pour les apprenants de personaliser l'interface graphique de leur environnement d'apprentissage. Il s'agit de faire en sorte que les LEA soient conçus de telle sorte qu'ils soient centrés sur les besoins des apprenants.

2. http://scorm.com/scorm-explained/technical-scorm/
3. http://www.imsglobal.org/learningdesign/
4. http://goo.gl/maps/Dvv0y
5. http://www.ietf.org/rfc/rfc4287.txt?number=4287
6. http://www.grownupdigital.com/archive/
7. http://www.facebook.com
8. http://www.twitter.com

1.3 Questions et objectifs de recherche

Le contexte général défini précédemment exprime le fait que les LEA, du fait qu'ils sont à la croisée de plusieurs domaines d'expertise, devraient évoluer pour prendre en compte d'une part les compétences, et d'autre part les connaissances issues des différents domaines d'expertise pendant le processus de conception. C'est pourquoi nous nous posons la question générale suivante : ***Comment faire pour prendre en compte les exigences pédagogiques dans le processus de conception des LEA ?***

De façon plus spécifique, nous nous sommes posés les questions suivantes :

1. ***Quelle approche de conception des LEA permettra de construire des LEA robustes satisfaisant aux exigences pédagogiques ?*** En effet à travers la revue de la littérature effectuée, nous avons observé un manque de synergie, voire de collaboration entre les acteurs oeuvrant pour améliorer la qualité des systèmes d'enseignement. D'un côté les experts en pédagogie travaillent et élaborent des modèles d'ingénierie mais ne peuvent pas projeter leurs perspectives au delà des aspects purement pédagogiques, tandis qu'en face les experts en ingénierie logicielle procèdent de façon similaire, en concevant des plateformes robustes sur le plan technologique mais que les experts en pédagogie trouvent inadaptées aux exigences pédagogiques.

   Plusieurs approches permettent de concevoir des LEA robustes. Cependant la revue de la littérature nous a permis de mettre en évidence un intérêt particulier pour l'utilisation du concept patron dans le processus de conception des LEA tant par les experts en pédagogie que par les experts en génie logiciel. Cela nous a conduit alors à la deuxième question subsidiaire.

2. ***Le concept* patron *peut-il aider à construire des LEA robustes satisfaisant aux exigences pédagogiques ?*** Nous avons constaté que ce concept est fort intéressant, car les patrons permettent de capitaliser la connaissance et l'expérience forgées au fil des années par les anciens experts, ce qui empêche les nouveaux de commettre les mêmes erreurs que leurs prédécesseurs et de perdre ainsi un temps précieux. Les patrons facilitent aussi le processus de développement, en réduisent les coûts, et facilitent davantage le processus de rétro-ingénierie d'un système, en cas de besoin. Nous avons comparé les principaux travaux qui relatent des utilisations de patrons pour la conception d'un LEA dans une perspective d'ingénierie logicielle. Cette comparaison nous a permis d'identifier dans la littérature des travaux de recherche dont l'analyse a révélé les leviers utilisés pour mettre en évidence la problématique de cette thèse. En particulier, nous avons constaté qu'il n'existe pas de cadre de synergie entre les différents acteurs de l'équipe de conception du LEA, les experts en pédagogie d'un côté et les experts en génie logiciel de l'autre. De plus les cycles de vie proposés dans ces travaux ne sont pas complets, ni rigoureusement décrits afin de permettre de développer des LEA efficients.

Enfin, la coexistence des exigences pédagogiques et des exigences logicielles n'est pas clairement définies dans les travaux comparés.

Les objectifs de cette thèse qui déclinent du contexte général et des questions de recherche présentés précédemment sont les suivants :

1. Proposer une démarche méthodique basée sur le concept de *patron*, pour concevoir un LEA. Il s'agira de montrer quelles doivent être les étapes séquentielles de conception d'un LEA prenant en compte les exigences pédagogiques.
2. Définir le répertoire de patrons qui seront intégrés dans la démarche.
3. Proposer une interface multipartite des experts en pédagogie, en technologie éducative et en ingénierie logicielle leur permettant de communiquer dans le but de développer un LEA. Il s'agira de façon plus spécifique de proposer un processus de conversion des artéfacts de l'ingénierie pédagogique en artéfacts pouvant être exploités par les ingénieurs logiciels et permettant de capitaliser l'expérience des experts des domaines de pratique et de connaissance connexes.

1.4 Méthodologie

Le travail effectué dans le cadre de cette thèse repose sur une démarche empirique caractérisée par une revue de la littérature rigoureuse, l'observation de situations d'enseignement et d'apprentissage et notre expérience de pédagogue, d'utilisateurs de LEA et d'analyste-concepteur-programmeur.

Tout d'abord dans le cadre de la revue de la littérature, nous avons dressé une liste, non exhaustive, des principaux modèles d'ingénierie pédagogique dont l'étude nous a permis de positionner notre travail dans le processus d'ingénierie pédagogique. En effet le LEA est un outil qui s'inscrit dans la liste des outils pédagogiques dont disposent les pédagogues pour exercer leurs tâches. Il s'agissait donc dans cette première partie de positionner notre travail dans le processus d'ingénierie pédagogique et de situer la place du concept patron dans ce même processus.

Nous avons ensuite étudié les différentes manières d'utilisation du concept patron pour concevoir des LEA. Le concept patron permet de capitaliser l'expérience des concepteurs plus expérimentés au profit des concepteurs moins expérimentés, d'où son intérêt dans notre approche. Cette étude nous a permis de déterminer le point de départ de notre contribution.

Par la suite, nous avons proposé une nouvelle approche séquentielle et itérative de conception des LEA basée sur les patrons et l'ingénierie dirigée par les modèles (MDE). Nous avons pour cela utilisé le langage de modélisation des systèmes SysML[9]. Pour chaque paquetage

9. http://www.omg.org/spec/SysML/

de notre approche, nous avons successivement représenté ses exigences de conception à l'aide d'un diagramme d'exigences et ses éléments structurels dans un diagramme de définition de blocs. Les aspects structurels ainsi mis en évidence au niveau de chaque paquetage de notre approche, à travers tous ces formalismes successifs, leurs propriétés et les exigences auxquels ils doivent répondre ont été recensés et ont constitué notre répertoire de patrons. Les différents formalismes du langage SysML ont été graphiquement représentés à l'aide de l'outil de modélisation Visual paradigm version 11.2 édition standard [10]. En plus des formalismes précédemment décrits, le catalogue de patrons de chaque phase est aussi décrit à l'aide d'un format de description textuelle des patrons.

Enfin nous avons évalué l'approche de développement de LEA basée sur les patrons, et les patrons proposés. Deux exemples ont servi de cadre d'application de l'approche de conception des LEA et d'utilisation des patrons proposés. Ce processus de validation a permis de conclure d'une part au réalisme de notre approche de conception des LEA et d'autre part que les patrons proposés sont bien valides et fonctionnels.

1.5 Résultats et contributions

A l'issue de cette thèse, nous avons mis en évidence une approche de conception de LEA basée sur les patrons, séquentielle et itérative. Il s'agit d'une approche qui se démarque des approches de conception logicielle classiques par le fait qu'elle est entièrement basée sur le concept patron. Cela permet aux concepteurs peu expérimentés de bénéficier de l'expérience de leurs pairs plus expérimentés, et aussi à des concepteurs n'ayant pas des notions très avancées en matière d'ingénierie logicielle de s'investir dans un projet de conception de LEA.

Nous avons aussi obtenu un répertoire de 110 patrons adéquats. Ces patrons permettent de capitaliser l'expérience des concepteurs de LEA expérimentés au profit des moins expérimentés. Compte tenu du grand nombre de patrons dans le répertoire, nous avons développé à cet effet un guide de recherche des patrons.

Ce répertoire de patrons facilitera le processus de conception logicielle au développeur de LEA peu expérimenté. Il sera guidé dans ce processus à l'aide de l'approche de conception séquentielle que nous avons proposée.

1.6 Plan de la thèse

Le **chapitre 2 - Etat de l'art** présente les principaux modèles, méthodes, approches issues du domaine des sciences de l'éducation et des modèles issus du domaine de l'informatique aux fins d'améliorer les systèmes d'enseignement et d'apprentissage. Nous avons comparé ces travaux qui relatent des utilisations de patrons pour la conception d'un LEA, dans une perspective

10. http://www.visual-paradigm.com/

d'ingénierie logicielle. Cela nous a permis de mettre en évidence l'utilité du concept et sa place dans le processus de conception des LEA. Cet état de l'art nous a aussi conduit vers la problématique et les objectifs traités dans le cadre de cette thèse.

Le **chapitre 3 - Approche de conception des LEA basée sur les patrons** présente l'approche séquentielle et itérative de conception d'un LEA constituée de huit étapes que nous avons proposée en guise de réponse à notre première question de recherche.

Le **chapitre 4 - Répertoire des patrons regroupés en paquetages** présente le répertoire comprenant 110 patrons organisés en paquetages.

Le **chapitre 5 - Cadre de validation de l'approche de conception des LEA basée sur les patrons** présente le processus de validation de l'approche de conception des LEA, et des patrons proposés, à travers deux cas de validation.

Cette thèse s'achève par une conclusion présentant une discussion sur les résultats obtenus et quelques perspectives de recherche.

Une vue d'ensemble de la séquentialisation du plan du document est représentée à la figure 1.1. Cette figure résume de façon tabulaire, d'une part, les éléments clés de chacun des chapitres de la thèse et montre les liens et interactions entre ces différents éléments d'un chapitre à l'autre, et d'autre part montre aussi la méthodologie suivie dans le contexte de cette thèse. Ce tableau est inspiré du cadre de la méthodologie de recherche définie par Holz et ses collègues (Holz *et al.* , 2006; Kohls, 2014) et axé autour de quatre questions :

A) Que voulons-nous faire ?
B) D'où proviennent les données ?
C) Que faire des données ?
D) Avons-nous atteint notre objectif ?

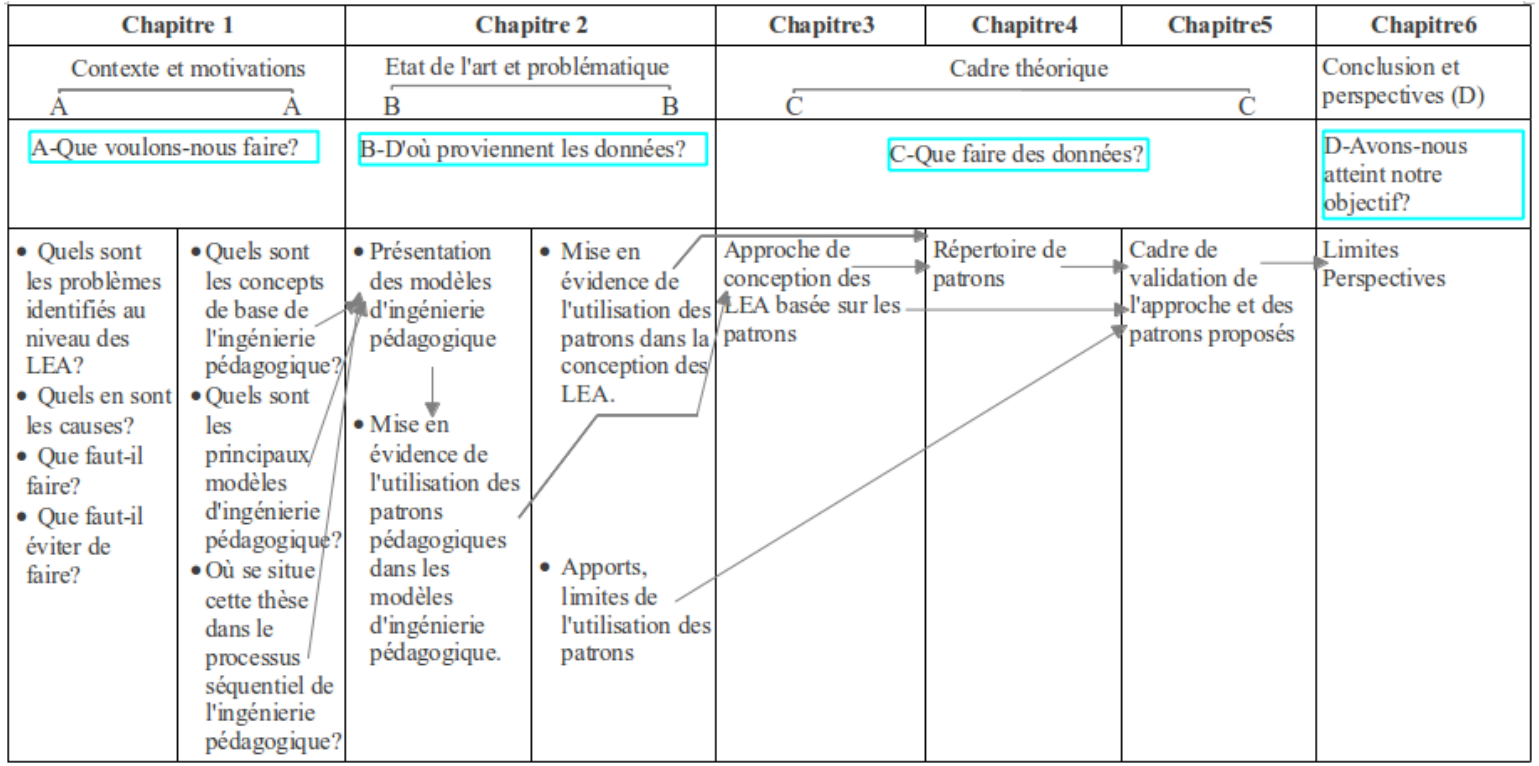


FIGURE 1.1 – Vue d'ensemble de la méthodologie utilisée. *Figure adaptée de (Holz* et al. *, 2006; Kohls, 2014)*

Cette thèse s'adresse aux experts du génie logiciel, c'est-à-dire aux analystes, concepteurs, développeurs, en tant que maîtres d'oeuvre du projet de conception d'un LEA. Ceux-ci ont en effet la responsabilité de la coordination du processus de conception du LEA conformément aux exigences pédagogiques définies par les experts en pédagogie. Toutefois, la lecture de cette thèse ne requiert pas au préalable des connaissances très avancées en génie logiciel. Le lecteur devra tout de même, pour la compréhension du chapitre 3 savoir ce qu'est le *cycle de vie d'un logiciel* et être familier avec le concept de *patron*. Cette thèse s'adresse aussi aux experts en pédagogie en leur qualité de maîtres d'ouvrage d'un projet de conception d'un LEA. Les experts en pédagogie constituent en effet l'entité qui définit et exprime le besoin, qui définit l'objectif et les exigences pédagogiques et qui choisit le maître d'oeuvre. Les experts en pédagogie représentent de ce fait le troisième type d'acteur concerné par le LEA, les apprenants.

Dans ce chapitre, nous avons présenté les concepts fondamentaux de cette thèse et ses motivations. Nous avons ensuite délimité le champ de notre thèse en spécifiant les questions auxquelles nous tenterons d'apporter des réponses. Nous avons terminé cette introduction en proposant un résumé par chapitre de cette thèse.

Puisqu'il est question dans le cadre de cette thèse de la conception de LEA devant satisfaire aux exigences pédagogiques, nous nous attèlerons dans le prochain chapitre à présenter une description des principaux modèles d'ingénierie pédagogique qui nous ont permis de comprendre les concepts de base de l'ingénierie pédagogique, et de situer cette thèse dans le processus séquentiel de l'ingénierie pédagogique.

Chapitre 2

Etat de l'art

Les experts en pédagogie et en ingénierie logicielle s'accordent sur le fait que les systèmes d'enseignement et d'apprentissage ont des failles d'origine conceptuelle et sur la nécessité de développer des approches de conception adaptées à leur contexte et en particulier à celui des LEA. Dans ce cadre, divers processus d'ingénierie ont été proposées tant du côté des experts en pédagogie que de celui des experts en génie logiciel en vue d'améliorer ces systèmes. Du côté des experts en pédagogie notamment, des approches, des modèles et des méthodes ont été proposés dans le but d'améliorer notamment les scénarios pédagogiques, tandis que du côté des experts en génie logiciel des approches de conception de logiciels ont été proposées, des approches utilisant l'ingénierie des ontologies à celles utilisant les systèmes multi-agents, en passant par les approches de conception par aspects et les approches orientées-objet. Ces approches utilisent en général un processus de conception qui part de rien et nécessite la réinvention perpétuelle des parties de logiciels. Or la complexité de plus en plus croissante des LEA suggère plutôt pour leur conception, l'utilisation d'approches permettant la réutilisation de composants logiciels, tel que les approches basées notamment sur les patrons.

Nous nous sommes particulièrement intéressés dans le cadre de ce chapitre à l'étude des processus d'ingénierie, tant pédagogique que logiciel, utilisant le concept *patron* du fait de la capacité des patrons à pouvoir capitaliser la connaissance et le savoir-faire et parce qu'ils permettent également la réutilisation de connaissances. Nous présentons donc dans ce chapitre l'état de l'art de ces différents processus.

Ainsi, dans la première partie du chapitre, nous présentons une liste non exhaustive des principaux modèles, méthodes et approches issus de l'ingénierie pédagogique en matière de conception des systèmes d'enseignement et d'apprentissage. Cette différenciation en modèles, méthodes et approches d'ingénierie ne résulte pas d'une volonté de catégorisation quelconque de ces éléments mais résulte naturellement de la lecture des noms que portent ces outils. Dans la deuxième partie du chapitre nous mettons en évidence la place du concept patron dans les processus d'ingénierie pédagogique et présentons aussi un état de l'art de la conception des

LEA basés sur le concept *patron*. Dans la troisième partie nous présentons une analyse comparative des travaux recensés et basés sur les patrons. Les résultats de cette analyse comparative nous ont permis de mettre en évidence l'importance de l'utilisation du concept *patron* dans le processus de conception des LEA, et nous ont aussi permis de définir la problématique, ainsi que l'objectif de recherche, de cette thèse, présentés dans la quatrième partie. Enfin nous terminons ce chapitre par une conclusion.

2.1 Modèles, méthodes et approches issus de l'ingénierie pédagogique

L'expression *ingénierie pédagogique*, autrefois *design pédagogique* est la traduction en langue française de l'expression anglaise *Instructional Engineering* (Basque, 2011).

Des experts en pédagogie ont défini le *design pédagogique* comme étant une discipline éducationnelle qui concerne la conception de stratégies définissant des situations pédagogiques (Sauvé, 1992) ou comme étant un processus systématique permettant la création d'activités d'apprentissage et de matériel d'enseignement (Smith & Ragan, 1999). L'expression *design pédagogique* fait aussi souvent référence au processus de conception des systèmes d'enseignement et d'apprentissage ainsi qu'au modèle descriptif des composantes faisant partie d'un système d'enseignement et d'apprentissage (Willis, 1995). Cette expression est aussi définie comme étant ce qui concerne l'élaboration de solutions d'apprentissage (Basque, 2011).

D'une part l'avènement et l'intégration des technologies de l'information et de la communication dans le processus du *design pédagogique* et d'autre part la prise en compte d'une démarche scientifique et systémique et l'emprunt d'outils au domaine de l'ingénierie logicielle entraîne la requalification du *design pédagogique* en *ingénierie pédagogique*.

Nous retenons de cet exercice de définition que l'expression *ingénierie pédagogique* fait référence à l'ensemble des méthodes permettant de savoir représenter les situations pédagogiques par des formalismes de telle sorte qu'elles puissent être réutilisées dans d'autres contextes.

Cet effort de formalisation a permis la définition de modèles, de méthodes et d'approches d'ingénierie pédagogique essentiellement centrés sur l'enseignant et son contexte d'enseignement, qui constituent des ensembles de principes permettant de produire des outils pédagogiques (Culatta, 2005; Quintin, 2011; Simmons, 2012) ou des méthodes permettant d'évaluer les besoins des étudiants (Jones & Davis, 2011) ou encore des représentations visuelles du processus d'ingénierie pédagogique (Chen, 2008).

Dans les paragraphes suivants, nous présentons les principaux modèles, méthodes et approches d'ingénierie pédagogique en matière de conception des systèmes d'enseignement et d'apprentissage.

2.1.1 Modèle ADDIE

L'accronyme ADDIE signifie en anglais *Analysis, Design, Development, Implementation, and Evaluation.* Le modèle ADDIE, apparu en 1975 (Don, 2010) est le modèle de conception pédagogique le plus répandu. Sa paternité est attribuée à plusieurs auteurs, mais malgré l'imprécision concernant son origine, il suscite un large consensus au sein de la communauté scientifique si bien qu'aujourd'hui le modèle ADDIE s'est imposé en modèle dominant parmi les modèles de conception pédagogique existants.

Le modèle ADDIE, comprend les cinq différentes étapes de conception suivantes :

1. *Analyse ou étude préalable de la situation :* lors de cette étape, le problème d'ordre instructionnel est cerné et défini. Les besoins ou les demandes du public sont aussi cernés. Les contraintes d'ordre instructionnelle sont définies également. Les objectifs et buts instructionnels sont délimités. L'environnement d'apprentissage et le niveau de l'apprenant sont identifiés. Les questions les plus pertinentes à se poser sont :
   a) quel est l'audience et ses caractéristiques ?
   b) quel est le temps imparti à l'activité ?
   c) quelles sont les contraintes pouvant influencer l'apprentissage des apprenants ?
2. *Conception :* lors de cette étape, les objectifs et buts d'apprentissage sont définis.
3. *Développement :* ici le contenu conçu à l'étape précédente est assemblé puis médiatisé.
4. *Implémentation ou implantation :* la formation des facilitateurs et des apprenants est mise en oeuvre à cette étape.
5. *Evaluation :* enfin la dernière étape constitue celle des tests et des évaluations des retours des utilisateurs.

La succession des étapes lors du processus de conception pédagogique avec le modèle ADDIE peut être non linéaire, ainsi qu'on peut le voir à la figure 2.1.

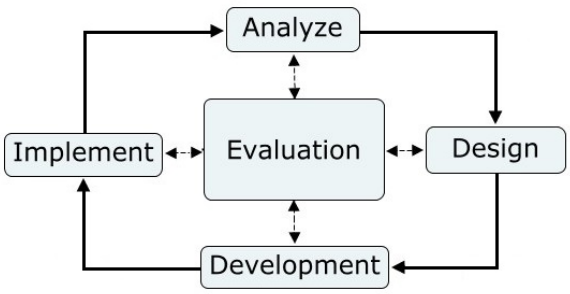


FIGURE 2.1 – Les 5 étapes du modèle ADDIE. (Forest, 2014)

Cependant, le modèle ADDIE s'avère être trop générique et contraignant à la fois et ne permet pas de définir un résultat avec une séquence chronologique rigoureuse. L'étape d'évaluation du modèle se situe à la fin du processus d'ingénierie, ce qui entraîne souvent une détection tardive des erreurs de conception.

2.1.2 Modèle proposé par Walter Dick et Lou Carey

Ce modèle, illustré à la figure 2.2, proposé par Dick et Carey en 1978, est linéaire et itératif, ce qui lui confère une certaine allure cyclique. Il est le modèle le plus répandu après le modèle ADDIE. Gustafson et Branch (Gustafson & Branch, 2002) ne tarissent pas d'éloges à l'endroit de ce modèle car il offre des outils de mesure des objectifs et des buts. Il est itératif, centré sur l'apprenant.

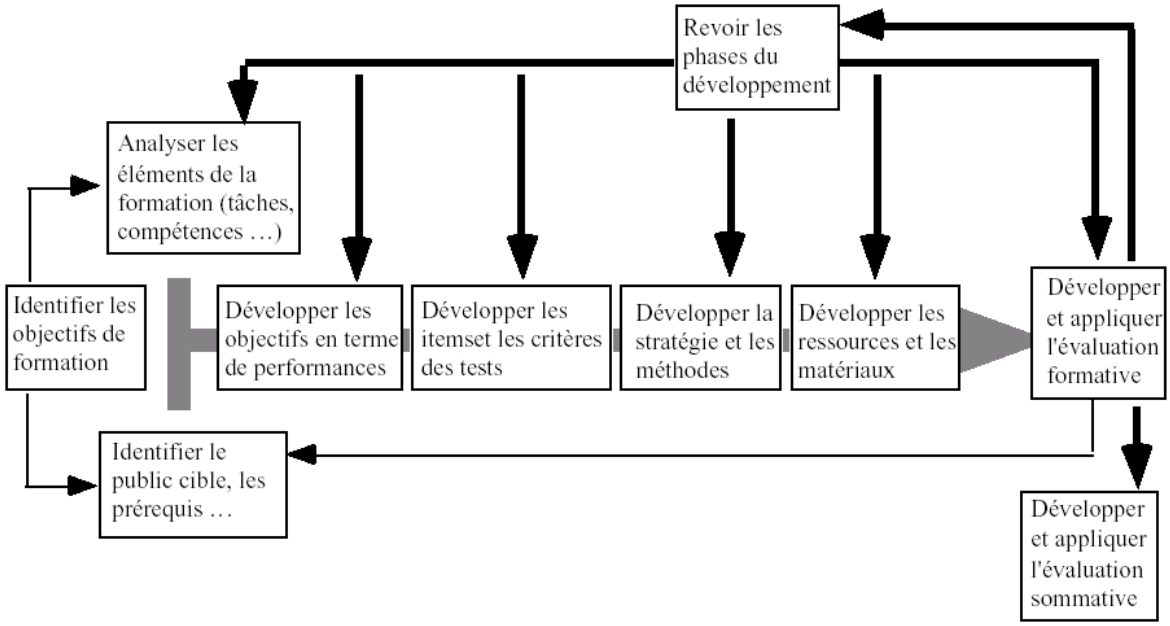


FIGURE 2.2 – Les dix étapes du modèle proposé par Dick et Carey. (Lebrun, 2002)

Ce modèle est plutôt destiné à la conception de programmes de formation et comprend dix grandes étapes majeures de nature plus complexes que celles du modèle ADDIE dont il reprend d'ailleurs les étapes. Toutefois, certaines étapes du modèle ADDIE ont été scindées en étapes à part entière (Branch & Gustafson, 1998). Ce modèle comprend de ce fait dix phases au total (Chen, 2008), ainsi qu'il suit :

1. identifier les objectifs de formation ;
2. analyser les éléments de formation ;
3. analyser les apprenants et leur contexte ;
4. définir les objectifs de performance ;

5. développer les outils (tests, critères de réussite, etc.) d'évaluation ;
6. développer les stratégies et les méthodes pédagogiques ;
7. développer des stratégies d'enseignement (ressources et matériels) ;
8. concevoir et mettre en oeuvre une stratégie d'évalutation formative ;
9. réviser le processus de formation ;
10. mettre en oeuvre une stratégie d'évalutation sommative.

Le modèle proposé par Dick et Carey est critiqué (Willis, 1995; Winn, 1992) entre autres pour sa linéarité et pour le fait que la méthodologie de planification utilisée est descendante et aussi parce qu'il s'inspire des méthodes de conception utilisées en génie logiciel. Ce dernier facteur rend le modèle proposé par Dick et Carey de peu d'intérêt pour les experts en pédagogie peu familiers avec les concepts du génie logiciel repris dans ce modèle.

2.1.3 Modèle RAPID

Le modèle RAPID, représenté à la figure 2.3, a été proposé par Tripp et Bichelmeyer en 1990 pour résoudre les problèmes de complexité et de lourdeur que leurs auteurs trouvaient dans les modèles existants (Tripp & Bichelmeyer, 1990).

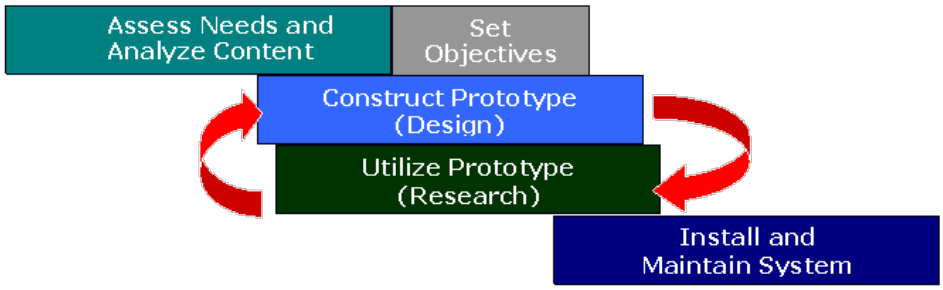


FIGURE 2.3 – Les 4 étapes du modèle RAPID. (Mitchell & Tashjian, 2014)

Ce modèle a pour but en particulier d'accélérer le processus de création des scénarios pédagogiques pour enseignants solitaires. Le modèle de Tripp et Bichelmeyer est un modèle iératif à 4 étapes ainsi qu'il suit :

1. analyse des besoins ;
2. construction d'un prototype ;
3. utilisation du prototype ;
4. installation du système.

Toutefois le modèle RAPID est victime de sa simplicité car certains estiment que plusieurs étapes jugées fondamentales du processus d'ingénierie ont été ignorées dans ce modèle ou négligées.

2.1.4 Modèle proposé par Kemp, Morrison et Ross

Ce modèle, schématisé à la figure 2.4, a été proposé en 1994 (Qureshi, 2004). Il est itératif et est aussi centré sur l'individu en particulier sur les besoins de l'apprenant.

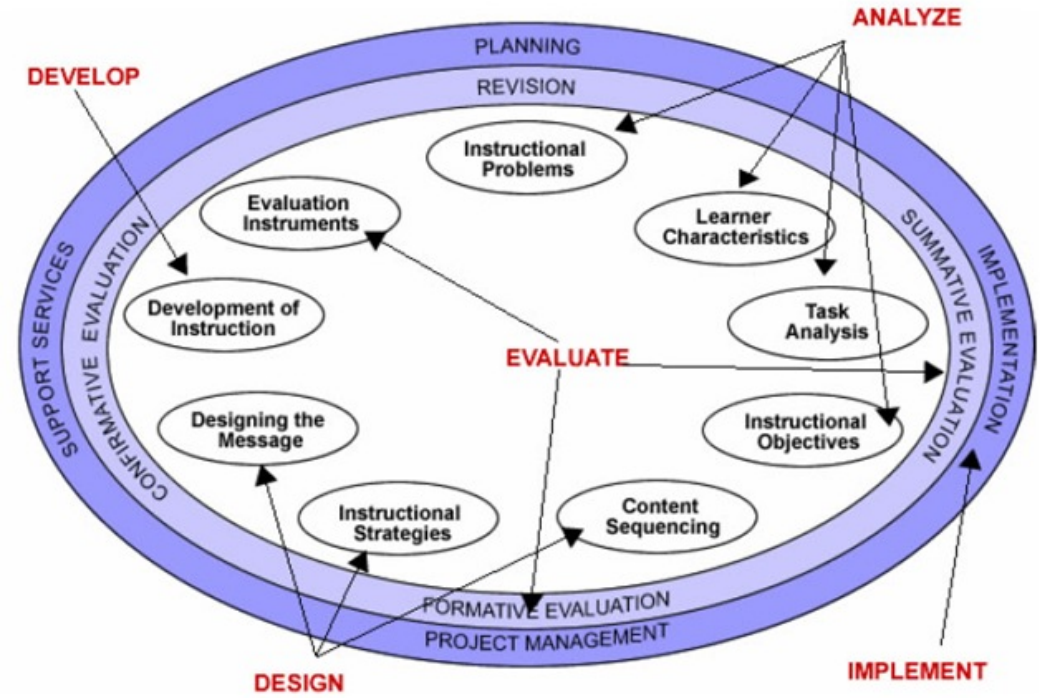


FIGURE 2.4 – Modèle proposé par Kemp, Morrison et Ross. (Morrison *et al.* , 2010)

Il met surtout l'emphase sur la phase de conception, et comprend les neuf étapes suivantes (Morrison *et al.* , 2010) :

1. identifier les problèmes pédagogiques et définir les objectifs d'un programme pédagogique ;
2. analyser les caractéristiques de l'apprenant, devant être l'objet d'une attention particulère ;
3. définir le contenu pédagogique et analyser les tâches liées aux objectifs et buts définis ;
4. définir des objectifs d'apprentissage pour l'apprenant ;
5. définir une séquence du contenu selon les unités pédagogiques ;
6. concevoir des stratégies pédagogiques permettant à chaque apprenant de se les approprier ;
7. planifier le message pédagogique et le mode de livraison ;
8. développer des outils d'évaluation pour évaluer les objectifs définis ;
9. choisir les ressources permettant de supporter l'enseignement et les activités d'apprentissage.

2.1.5 Modèle R2D2

Le modèle R2D2, illustré à la figure 2.5, a été proposé par Willis en 1995 (Murphy *et al.* , 1994). Le sigle R2D2 signifie en anglais *Recursive, Reflective, Design and Development.*

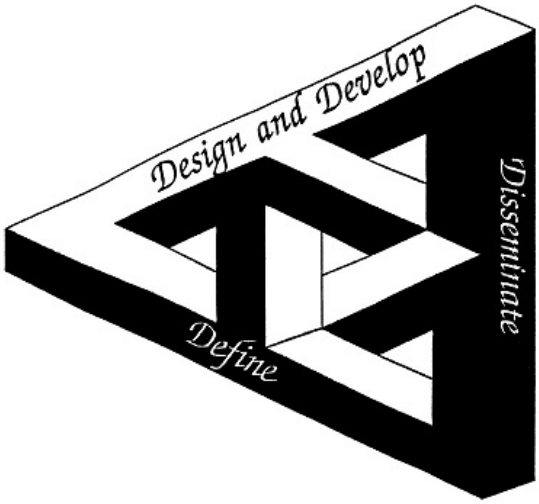


FIGURE 2.5 – Les 4 étapes du modèle R2D2, selon Dick. (Dick, 1996)

Le modèle R2D2 est proche de celui proposé par Dick et Carey, et a été construit selon la théorie constructiviste. Il est itératif et récursif, et comprend les quatre étapes suivantes :

1. définition : il s'agit de l'analyse des besoins de l'étudiant, l'analyse des tâches, et la définition des objectifs.
2. conception : elle concerne la sélection du format des média de diffusion, d'un environnement de développement et l'élaboration d'une stratégie d'évaluation.
3. développement : il s'agit de l'implémentation du modèle obtenu lors de l'étape de conception.
4. dissémination ou de diffusion des matériels.

Le modèle R2D2 se focalise sur l'enseignement en ligne et constitue un cadre de développement pour gérer et organiser le matériel pédagogique.

Il est reproché au modèle R2D2 notamment le fait qu'il met l'accent sur la conception de ressources pédagogiques destinées à l'enseignement des méthodes de recherche qualitatives.

2.1.6 L'approche MOCA

L'approche MOCA (Modélisation et Optimisation des Cycles d'Apprentissage) (Barré *et al.* , 2003; Barré, 2004; Bouabib, 2005) est une approche expérimentale de l'ingénierie des Environnements Informatiques pour l'Apprentissage Humain (EIAH). Elle s'intéresse à la réingénierie

pédagogique d'un système d'apprentissage à distance et est basée sur un système de scénarisation pédagogique.

La méthode MOCA se base sur l'hypothèse selon laquelle les EIAH laissent des traces informatiques pouvant renfermer de la connaissance d'ordre pédagogique et pouvant aider à la reconstruction du scénario pédagogique.

De ce fait, cette méthode permet d'anticiper la construction du scénario descriptif c'est-à-dire le déroulement effectif des situations d'apprentissage et exprime pour cela les besoins d'observation dans le scénario prévu par les concepteurs.

Pour valider leur hypothèse, d'une part, les auteurs ont mesuré la distance entre les activités réelles d'apprentissage des apprenants et les scénario initialement prévus, et d'autre part ont procédé à la liaison les unes aux autres des informations résultantes de l'interaction entre les agents humains.

Malgré le fait qu'elle soit présentée comme étant une approche de réingénierie pédagogique d'un système d'apprentissage, force est de constater que la méthode MOCA est surtout centrée sur les aspects liés à l'évaluation de l'apprentissage pour en extraire de la connaissance et n'offre pas à ses utilisateurs potentiels une démarche rigoureuse et systémique d'ingénierie leur permettant de développer un EIAH robuste.

2.1.7 La méthode MISA

La Méthode MISA développée par Paquette (Paquette, 2002) au Centre de recherche CIRTA de la TELUQ [1]. MISA est essentiellement basée sur la modélisation de la connaissance et comprend six phases et quatre axes ou modèles tel qu'illustré à la figure 2.6. Les phases et les axes représentent deux différentes approches de construction d'un modèle de système d'apprentissage. Le croisement des deux approches, par phases et par axes, confère à la méthode MISA une allure itérative et en spirale.

1. Constituante du réseau de l'Université du Québec au Canada et seule université francophone en Amérique du Nord à offrir tous ses cours à distance.

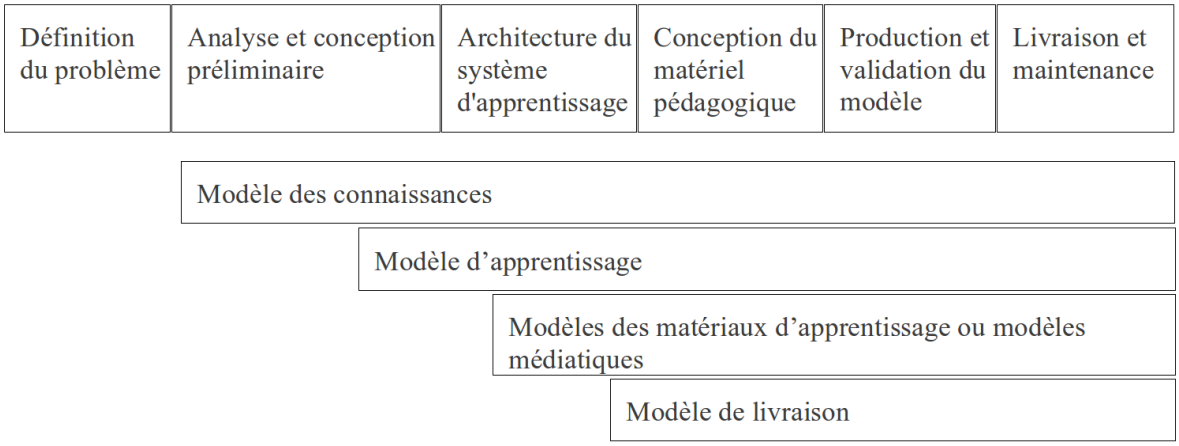


FIGURE 2.6 – Les six phases et quatre modèles de la méthode MISA (Oubahssi & Grandbastien, 2008)

L'approche par phases équivaut au processus d'ingénierie pédagogique tandis que celle par axes correspond à la situation où la conception de l'environnement d'apprentissage nécessite des compétences multidisciplinaires et servent à guider le concepteur du logiciel d'enseignement à distance durant le processus de conception.

Les quatre axes ou modèles de la méthode MISA sont :

1. l'axe de la conception du contenu ou modèle des connaissances ou encore axe de conception du contenu : il permet de modéliser la connaissance et les aptitudes. Il est une représentation graphique du domaine de connaissance étudié.
2. l'axe de la conception de scénarios et d'activités pédagogiques ou modèle d'apprentissage : il permet de décrire les unités d'apprentissage.
3. l'axe du support à la production de nouveau matérial pour l'apprentissage ou modèle des matériaux d'apprentissage ou encore modèle médiatique : il est constitué des représentations graphiques des unités d'apprentissage et permet de décrire les propriétés des différents matériels.
4. l'axe de planification de la livraison ou modèle de livraison : il correspond à l'ensemble des ressources du modèle d'apprentissage et permet de décider comment les différents livrables seront diffusés aux utilisateurs du système d'apprentissage.

La Méthode MISA est munie d'un éditeur graphique de modèles de connaissances, MOT+, permettant de représenter la connaissance sous la forme d'un graphique, d'un atelier distribué d'ingénierie de systèmes d'apprentissage avec une interface web, ADISA, et d'un système de conception et de diffusion de la formation en ligne (Explor@).

Congratulée pour son exhaustivité, la méthode MISA se retrouve victime de sa complétude. Elle comprend trop de détails qui paradoxalement n'améliorent pas son utilisabilité, bien au contraire, et qui contre toute attente rendent davantage confus les concepteurs de systèmes

pédagogiques d'enseignement à distance ! En outre, certains attributs tels que les critères de qualité, les procédures de maintenance et de mise à jour sont peu détaillés ou ne sont pas pris en compte (Oubahssi & Grandbastien, 2008). Enfin, du fait de son antériorité, cette méthode ne prend pas en compte certains aspects liés à l'interopérabilité (Barré *et al.* , 2003).

La méthode MISA est complète et de ce fait s'avère être très complexe à utiliser. De plus elle s'avère finalement plus adaptée à la scénarisation pédagogique qu'à autre chose.

2.1.8 L'approche LTSA

L'organisme IEEE [2] a développé une méthode nommée IEEE-LTSA (Learning Technology Systems Architecture) de développement de systèmes d'enseignement. La méthode LTSA constitue un cadre abstrait dont le but est de produire des spécifications et recommandations et un guide des meilleures pratiques pour le développement des systèmes d'enseignement à distance. Elle regroupe donc un ensemble de normes définissant un cadre architectural pédagogique de haut niveau pour les systèmes d'enseignement à distance et prend en compte plusieurs aspects des processus et activités d'apprentissage. Ces aspects pris en compte sont :

1. le contexte des acteurs humains ;
2. le modèle d'organisation ;
3. le cycle de développement logiciel.

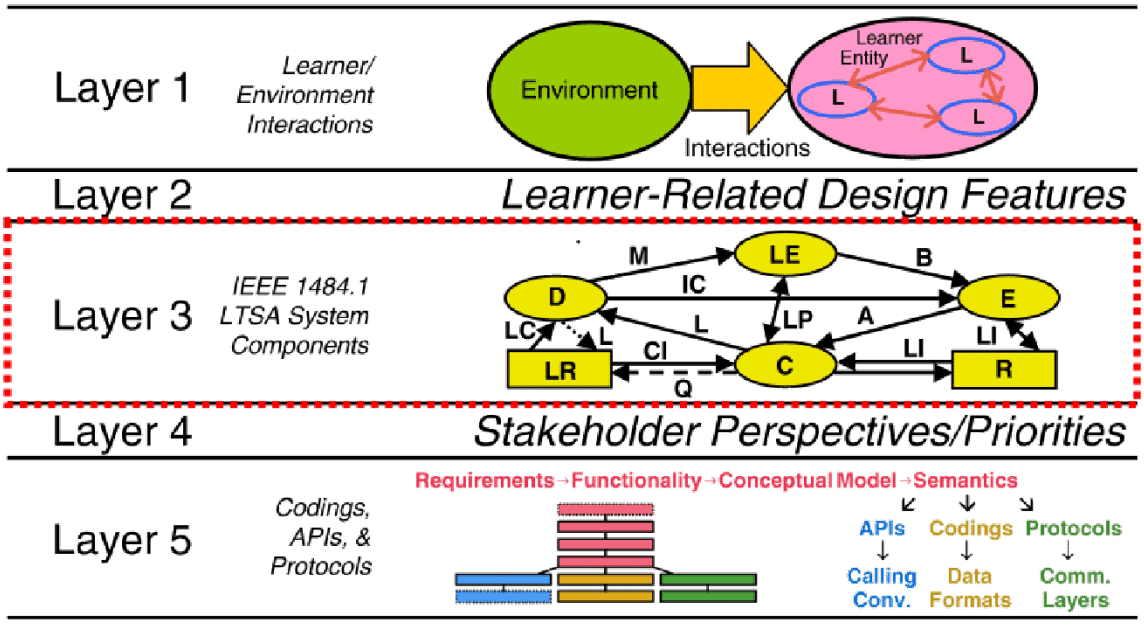


FIGURE 2.7 – Architecture de la méthode LTSA (Mannhardt, 2007)

2. Institute of Electrical and Electronics Engineers :`http://www.ieee.org/`

L'architecture de la méthode LTSA, illustrée à la figure 2.7, comprend cinq couches ainsi qu'il suit :

1. La couche 1 est la couche la plus abstraite. Elle définit les tâches d'acquisition, de transfert.

2. La couche 2 définit la réaction de l'apprenant face à son environnement.

3. La couche 3 définit l'organisation du processus d'apprentissage du point de vue des données et des flux de contrôle. Trois types de composants sont représentés au niveau de cette couche :

   - Les processus représentés par quatre rectangles ou quatre ellipses selon le cas peuvent être l'entité apprenant, la fonctionnalité «évaluation» ou la fonctionnalité «livraison». Ils sont décrits en termes de limites, prennent des données reçues des flux en entrée, et produisent des données en sortie.

   - L'entrepôt représenté par deux rectangles peut stocker des informations sur l'apprenant, sur les ressources d'apprentisage, etc. Il est décrit par le type d'information stockée et par les méthodes de recherche et de mise à jour.

   - Les flux représentés par les flèches décrivent en termes de connectivité (uni ou bi-directionnelles, connexions statiques ou dynamiques, etc.) le type d'information échangé dans le flux tel que les préférences de l'apprenant, les comportements, des informations sur l'apprenant, des informations sur le contexte multimédia et d'interaction, etc.

4. La couche 4 permet de formaliser les contraintes de conception technologiques et d'identifier les activités du système durant le processus d'apprentissage.

5. La couche 5 définit les phases abstraites du processus de dévelppement du logiciel selon une approche par composants.

Le modèle LTSA n'a pas connu un grand succès, victime des critiques dont il a fait l'objet (O'Droma *et al.* , 2003; Corbière & Choquet, 2004; Oubahssi & Grandbastien, 2008). Il lui est notamment reproché le fait d'être centré uniquement sur les aspects techniques (Oubahssi & Grandbastien, 2008), et le fait de ne pas prendre en compte la possibilité de création ou de mise à jour des ressources d'enseignement, (Oubahssi & Grandbastien, 2008) ou d'apprentissage (Lindner, 2006). Enfin le modèle LTSA prend très peu en compte la possibilité d'accompagnement par conseil des étudiants (Kumar *et al.* , 2010) durant le processus d'enseignement ou d'apprentissage.

2.1.9 L'approche IMS Learning Design

La spécification IMS [3] est un système d'apprentissage issu de l'implémentation des standards du groupe de travail international dénommé *IMS Global Learning Consortium* [4] et présenté dans plusieurs travaux (Lindner, 2006; Stracke, 2006; Moreno Ger *et al.* , 2007).

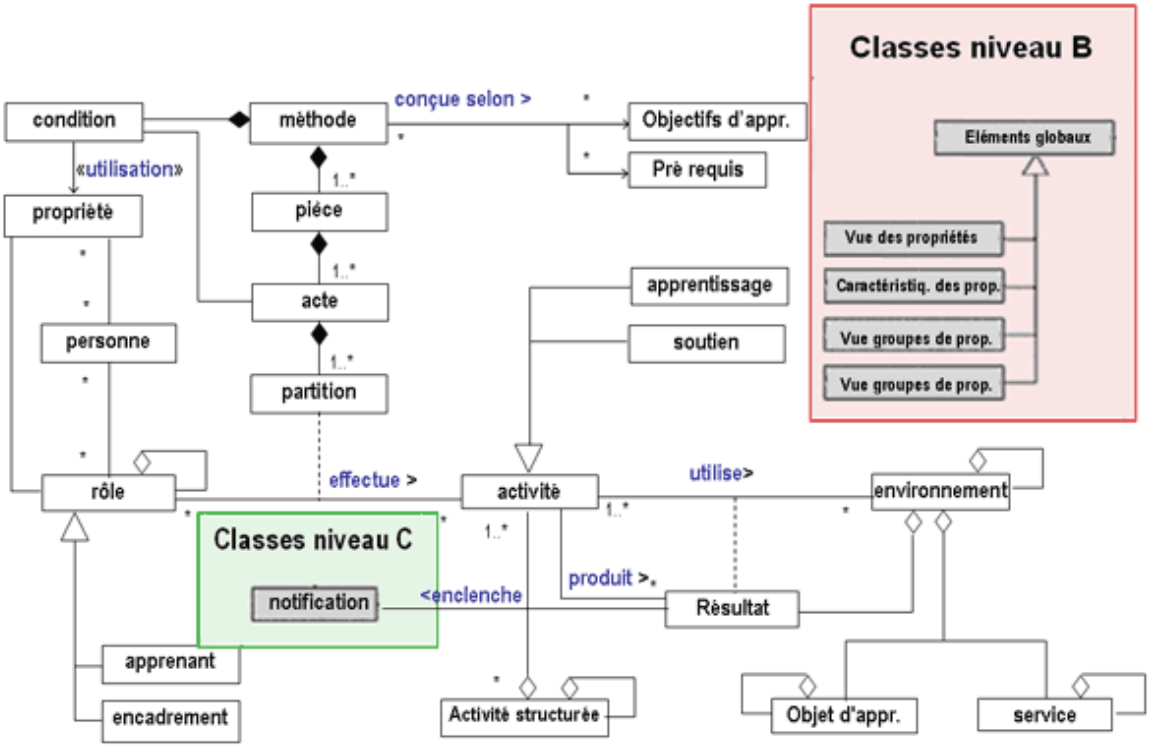


FIGURE 2.8 – Architecture de la méthode IMS Learning Design (Burgos *et al.* , 2006)

Dans cet outil, dont l'architecture est illustrée à la figure 2.8, l'aspect technologique semble avoir été privilégié au détriment des aspects pédagogiques, fait qui constitue une faiblesse qui ne passe pas inaperçue. Sa grande capacité à s'adapter à plusieurs cas d'utilisation fait d'IMS un outil très apprécié dans les domaines techniques, ce qui par la même occasion fait oublier aux spécialistes de l'éducation que l'outil a été conçu pour le domaine de l'éducation (Lindner, 2006).

2.1.10 Place du concept *patron* dans les processus d'ingénierie pédagogique

Les résultats de l'analyse des neuf modèles, méthodes et approches d'ingénierie pédagogique présentés précédemment permettent de constater le fait que les experts en pédagogie défi-

3. Le nom initial du projet était en 1997, *Instructional Management Systems*, d'où l'acronyme actuel IMS. Ce nom évolua par la suite pour éviter d'une part des quiproquos et d'autre part pour résoudre maintes ambiguïtés, mais le projet conserva jusqu'au moment de la rédaction de cette thèse en 2016 son acronyme initial. Pour plus de détails voir la page suivante : `http://www.imsproject.org/background.html`

4. `http://www.imsproject.org/`

nissent l'ingénierie pédagogique comme étant un processus constitué d'étapes permettant de concevoir des systèmes d'enseignement et d'apprentissage et prévoient même pour la plupart une étape dite de conception logicielle, lorsqu'une solution logicielle est préconisée. La contribution scientifique de cette thèse se situe précisément au niveau de cette étape de conception logicielle faisant partie intégrante du processus d'ingénierie pédagogique.

Il existe plusieurs façons de concevoir des logiciels parmi lesquelles les approches dites *traditionnelles* et les approches dites *émergentes* (Hachani, 2006; Agarwal *et al.* , 2012; Enard, 2013; Vyatkin, 2013). Les approches dites traditionnelles sont composées des approches par conception, par aspects, orientées-objet, et orientées-agent, dans le cadre desquelles le processus de conception part du néant et nécessite la réinvention perpétuelle des parties de logiciels (Hachani, 2006). Les approches par réutilisation de composants, quant à elles, sont basées notamment sur les patrons et permettent de réutiliser des composants d'analyse, architecture ou de code existants.

Le but de notre travail n'était pas d'étudier toutes ces façons de concevoir mais plutôt de faire un choix en fonction du contexte étudié. Parmi les différentes façons de concevoir des logiciels, nous avons choisi de baser notre contribution sur une approche basée sur les patrons, du fait de la capacité des patrons à capitaliser la connaissance et le savoir-faire et parce qu'ils permettent également la réutilisation de connaissances. De ce fait, les concepteurs bénéficient, héritent aussi de l'expérience de leurs pairs plus expérimentés, ce qui entraîne en conséquence l'amélioration des logiciels produits.

D'ailleurs dans notre analyse transversale, nous avons constaté que cette approche par patron est privilégiée. En effet, nous avons observé que le concept de base autour duquel les processus décrits sont conçus est le *scénario pédagogique*. Ils partent tous de la définition ou de l'observation d'un scénario pédagogique initial et aboutissent à la définition d'un scénario pédagogique prédit, ou mieux à la modélisation du scénario pédagogique de base.

C'est le cas, par exemple, de l'approche MOCA qui part de l'hypothèse selon laquelle il existe un scénario pédagogique initial préconçu c'est-à-dire idéal, et un scénario pédagogique d'arrivée correspondant aux activités réelles des étudiants, et essaie ensuite d'évaluer la distance entre les deux types de scénarios. Elle utilise ensuite le langage EML [5] pour modéliser les scénarios pédagogiques considérés.

Le concept de scénario pédagogique se retrouve aussi au coeur de la méthode MISA et se décline en deux composantes : le scénario d'apprentissage et le scénario d'assistance (Paquette *et al.* , 2001). Le scénario pédagogique est ensuite érigé en un modèle pédagogique mettant en évidence toutes les activités, ressources et consignes requises par les acteurs du système d'apprentissage.

5. Educational Modeling Language

Le modèle IEEE-LTSA considère lui, comme unité de base de son architecture, le scénario de l'activité d'apprentissage (Karampiperis & Sampson, 2005). Les différents cas de scénario d'apprentissage sont ensuite structurés et modélisés et les interactions entre les différents acteurs explicitées et représentées.

L'approche IMS-Learning Design quant à elle part d'un plan de cours qu'il modélise ensuite grâce à la description des différents rôles, activités, environnements (Burgos *et al.* , 2006).

Le scénario pédagogique permet donc de décrire d'une part une situation d'enseignement, c'est-à-dire qu'il décrit en détail le déroulement des activités d'enseignement mais apporte aussi des solutions aux problèmes récurrents que peuvent rencontrer les enseignants dans l'exercice de leurs tâches. D'autre part le scénario pédagogique permet aussi de décrire le déroulement d'une séquence d'activités d'apprentissage. Ces observations permettent de reconnaître ici au concept de scénario pédagogique les propriétés qui définissent le concept de *patron pédagogique*, décrit comme étant une solution visant à résoudre un problème récurrent bien cerné (Derntl, 2005; Finlay *et al.* , 2009).

Étant donné l'importance de ce concept de patron dans le processus d'ingénierie pédagogique, nous avons décidé d'explorer la manière dont ce concept pouvait être mis à profit pour concevoir des LEA. Nous avons donc étudié différents travaux qui utilisent des patrons pour cette étape de conception logicielle. Nous les avons comparé selon trois groupes de critères afin de mettre en évidence leurs points forts et leurs points faibles. Les leçons apprises de cette comparaison nous ont ensuite aidé à déterminer une problématique de recherche et des objectifs qui lui sont reliés.

2.2 Utilisations du concept *patron* pour concevoir des LEA

Cette section présente le concept de *patron* tel qu'il est perçu et utilisé dans les processus pour concevoir un LEA issus du domaine du génie. Nous présentons les principaux travaux recensés dans ce cadre.

2.2.1 Les patrons proposés par Randriamalaka

Randriamalaka dans ses travaux vise deux objectifs : améliorer les scénarios pédagogiques (Randriamalaka, 2006; Randriamalaka & Iksal, 2006) et aider à la réingénierie des systèmes d'enseignement à distance (Randriamalaka, 2005).

Sur le plan de l'ingénierie logicielle, Randriamalaka, comme Avgeriou et ses collègues (Avgeriou *et al.* , 2003), constate le fait que malgré la longue expérience de l'industrie en matière de conception des LEA, ces logiciels sont encore aujourd'hui conçus sans une véritable approche systématique capitalisant et documentant l'expérience accumulée durant les années antérieures et celles des anciens développeurs de logiciels.

Dans le cadre de l'amélioration des scénarios pédagogiques, il propose deux types de patrons dédiés à la réingénierie d'un scénario pédagogique. Le premier type de patrons constitue les *patrons d'analyse* et permet de décrire la solution au problème lié à l'utilisation des scénarios pédagogiques. Le second type de patrons est constitué de *patrons de conception* et décrivent eux, la solution au problème de conception du scénario pédagogique. L'objectif ultime est de fournir aux concepteurs pédagogiques des techniques qui leur permettront de produire des scénarios pédagogiques améliorés et donc plus efficients.

Pour trouver les patrons qu'il propose, Randriamalaka choisit de détecter, puis capitaliser la connaissance ambiante présente dans les fichiers journaux des systèmes d'apprentissage à distance en vue de la reingénierie de ces systèmes (Randriamalaka & Iksal, 2006; Randriamalaka *et al.* , 2008). De façon pragmatique, il procède à l'analyse des traces d'usage, c'est-à-dire à la collecte puis l'analyse du comportement des utilisateurs du système durant les sessions d'apprentissage en vue de définir des patrons de conception selon le modèle conceptuel illustré à la figure 2.9.

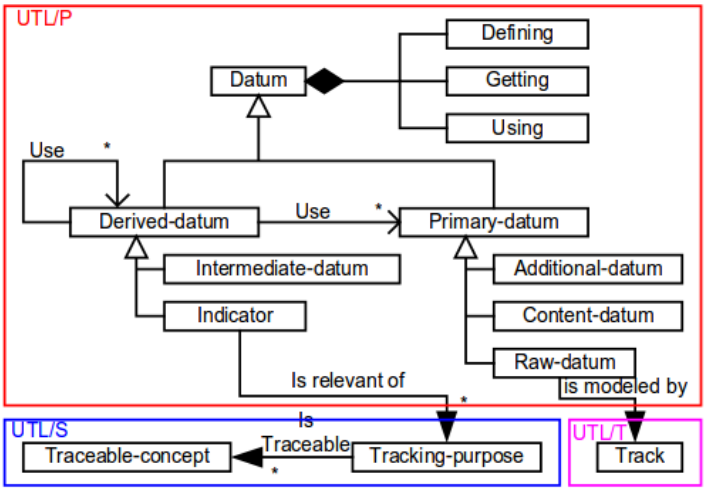


FIGURE 2.9 – Modèle conceptuel du système d'analyse des traces d'usage basé sur les patrons de conception. (Randriamalaka *et al.* , 2008) .

L'auteur définit son propre format de description de patron inspiré de celui de Bochers (Borchers, 2001). Son format répond aux questions suivantes :

1. Quels comportements observer pour une analyse objective ?
2. Quels outils utiliser pour une analyse objective ?
3. De quelle méthode d'enseignement faut-il analyser les comportements des apprenants ?
4. Quels sont les patrons de conception proches de ces considérations ?

Cependant, la méthode de capitalisation de la connaissance proposée par Randriamalaka s'intéresse seulement aux traces produites par les utilisateurs pour en déduire des patrons de

réingénierie de scénarios pédagogiques mais ne s'illustre pas comme une démarche de conception des LEA.

2.2.2 Les patrons proposés par Derntl

Les patrons pédagogiques proposés par Derntl (Derntl, 2005) ont pour objectif, entre autres, d'aider les concepteurs pédagogiques à concevoir des cours en ligne et documenter et partager leur expérience. Le souci de la capitalisation de la connaissance est aussi latent. Il propose un répertoire de patrons pédagogiques, dont une synthèse est illustrée à la figure 2.10 constituée de scénarios d'enseignement et d'apprentissage centré sur l'individu. Ces scénarios concernent par exemple les situations de gestion des ressources d'apprentissage, des aspects organisationnels des activités pédagogiques, et de l'apprentissage coopératif.

D'autres travaux ont aussi consisté en la création d'un répertoire de patrons dans le but de fournir une aide à la conception des cours. Ces patrons s'appliquent à n'importe quelle discipline (Bergin, 2000, 2004), ou sont spécifiques. Par exemple, certains de ces patrons concernent en particulier les sciences informatiques (Bergin, 2000) ou les mathématiques (Mor *et al.* , 2006), ou concernent le cadre d'activités d'encadrement académiques, de thèses par exemple (Schmolitzky & Schümmer, 2008; Köppe, 2012). D'autres encore concernent le contexte de l'enseignement à travers les plateformes d'enseignement et d'apprentissage à distance (Shi *et al.* , 2013; Littlejohn & Pegler, 2014; Mor *et al.* , 2014).

Malgré la richesse de leur diversité, les patrons proposés par Derntl sont des modèles de scénarios pédagogiques qui comme leur nom l'indique s'adressent essentiellement aux concepteurs pédagogiques. Bien qu'étant proposés aux fins de conception d'un LEA, ces patrons, du fait de leur nature péadagogique, ont une expressivité limitée et ne peuvent pas être appliqués en l'état par les développeurs de logiciels pour concevoir un LEA.

2.2.3 Les patrons proposés par Goodyear, Avgeriou, Baggetun, Bartoluzzi, Retalis, Ronteltap et Rusman

Goodyear et ses collègues (Goodyear *et al.* , 2004; Goodyear, 2005) quant à eux visent deux objectifs : d'abord mieux définir les problèmes auxquels font face les pédagogues et ensuite proposer des patrons permettant d'apporter des solutions aux problèmes repertoriés. Ces problèmes sont par exemple la non maîtrise des outils technologiques par les enseignants, l'insuffisance des compétences en matière d'analyse et de conception de ressources pédagogiques, ou encore la difficulté pour les enseignants d'organiser des groupes de conversations ou de débats à des fins pédagogiques. Le concept de patron de conception est alors introduit dans ses travaux pour définir une approche pouvant aider de façon méthodique les enseignants à développer leurs cours. Cette approche se base sur la volonté de réutilisation de la connaissance acquise par d'autres enseignants plus expérimentés en la matière.

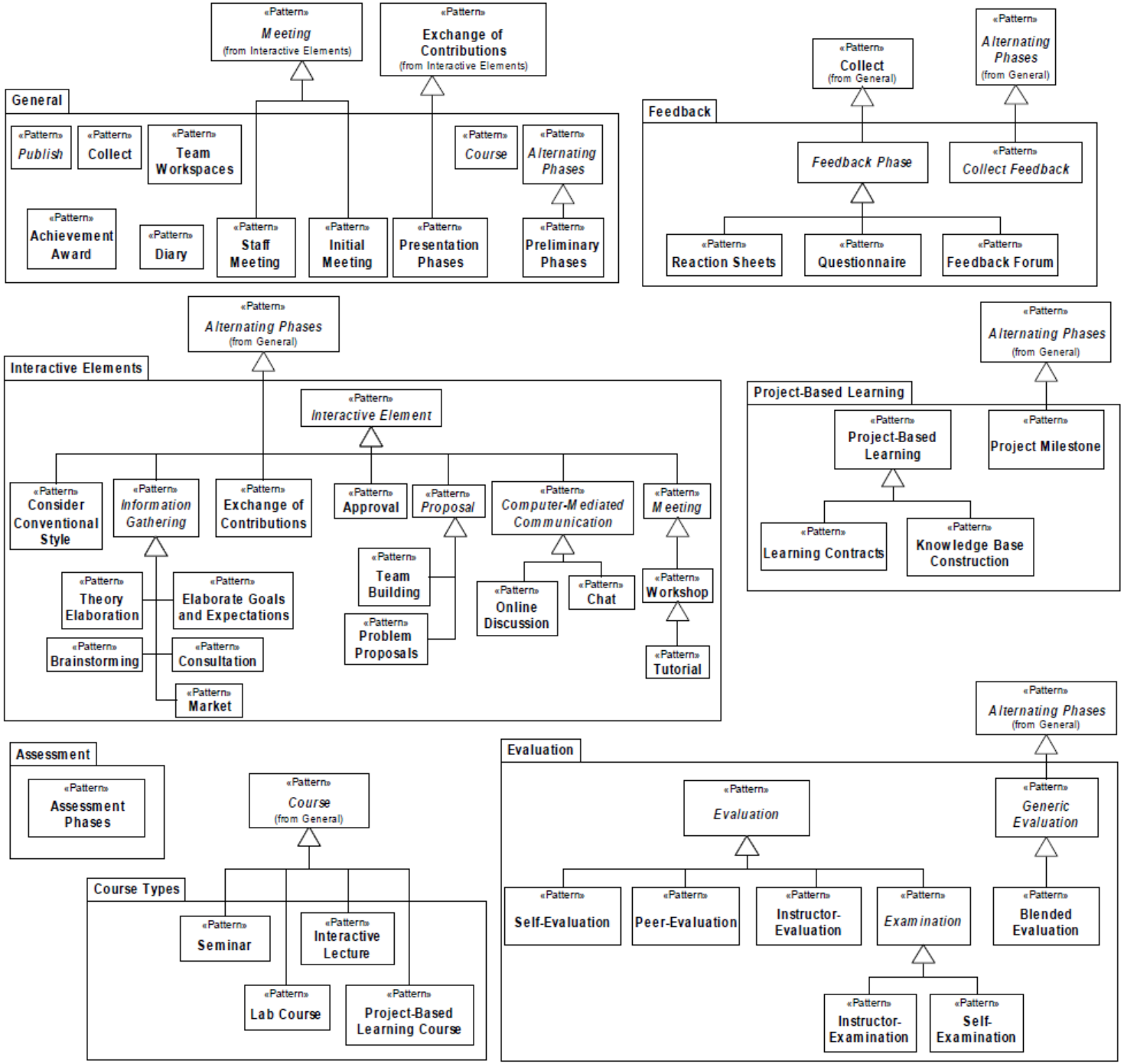


FIGURE 2.10 – Patrons pédagogiques proposés par Michael Derntl. (Derntl, 2004)

Mais les patrons proposés par Goodyear et ses collègues dont l'objectif est l'amélioration des LEA se limitent eux aussi aux processus pédagogiques. L'exemple de langage de patrons que proposent ces auteurs, et représenté à la figure 2.11, sert à gérer une activité pédagogique de débat.

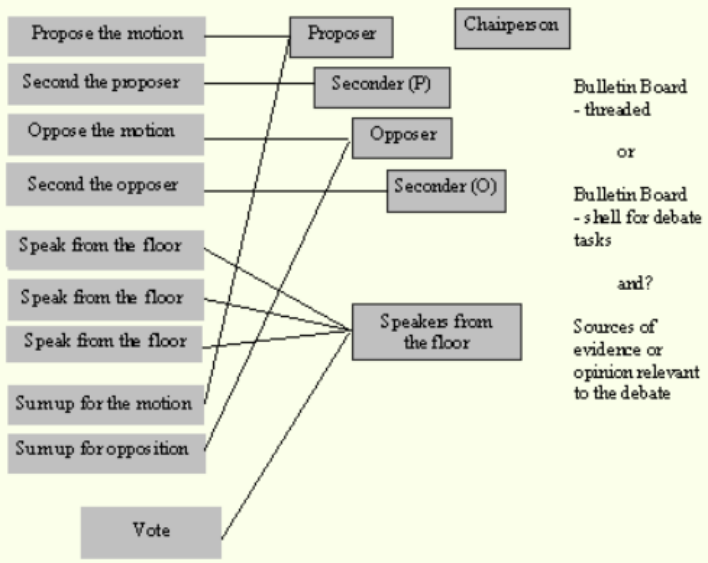


FIGURE 2.11 – Langage de patrons représentant le scénario d'un débat. (Goodyear, 2005)

2.2.4 Les patrons proposés par Avgeriou, Retalis et Papasalouros

Avgeriou et ses collègues (Avgeriou *et al.* , 2003) font le même constat que Randriamalaka (Randriamalaka, 2006), et déplorent d'abord le fait que malgré la longue expérience de l'industrie en matière de conception des LEA, ces systèmes sont encore aujourd'hui conçus sans une véritable approche systématique capitalisant et documentant l'expérience accumulée durant les années antérieures et celles des anciens développeurs. En conséquence, en matière de conception des systèmes de gestion d'apprentissage les concepteurs et développeurs ont sans cesse tendance à vouloir réinventer la roue.

Pour apporter une réponse au problème qu'ils posent, les auteurs proposent d'une part de concevoir un système de gestion d'apprentissage (Learning Management Systems) basé sur un réseau de patrons illustré à la figure 2.12, et d'autre part un répertoire de patrons pour les systèmes de gestion de l'apprentissage.

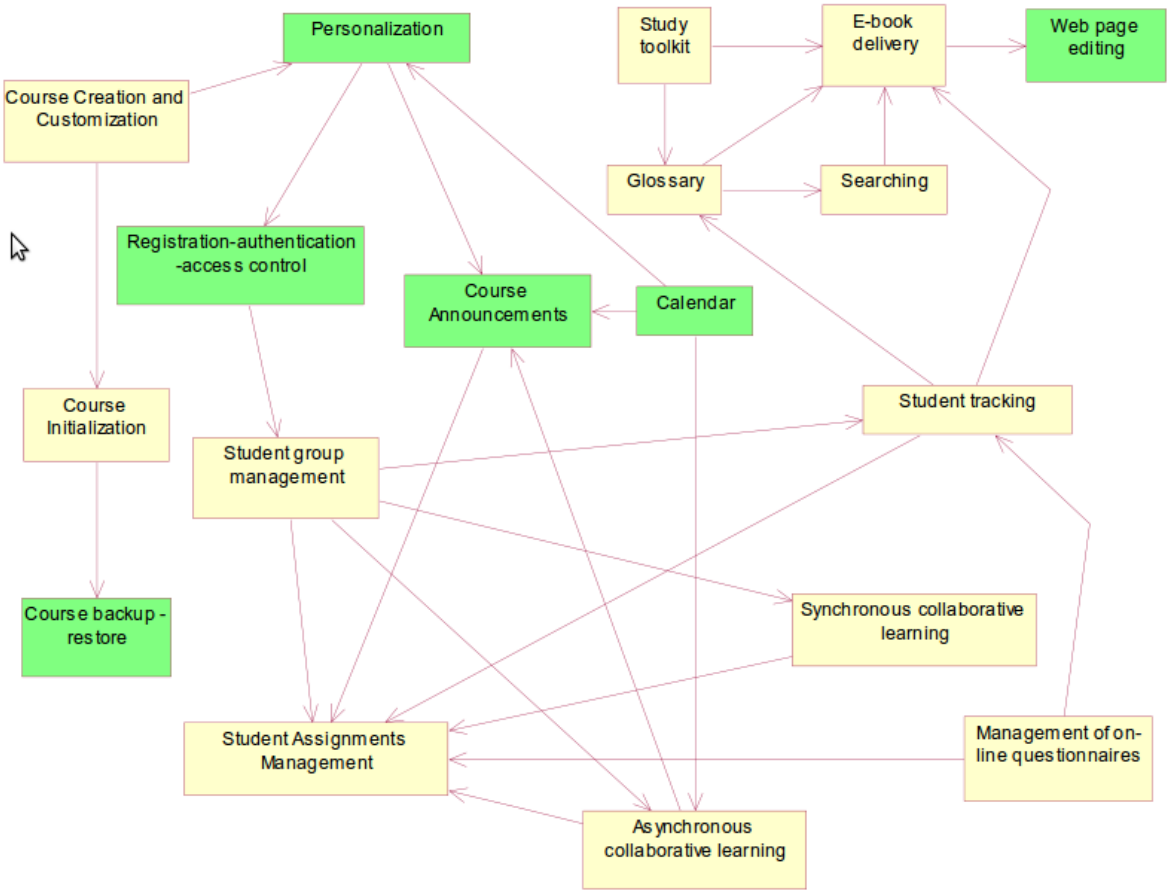


FIGURE 2.12 – Réseau de patrons pour un système de gestion de l'apprentissage. (Avgeriou *et al.* , 2003)

.

Dans ce réseau de patrons, les auteurs considèrent trois catégories d'utilisateurs sur un système de gestion d'apprentissage : les apprenants, les enseignants et les administrateurs. Ils mettent en évidence quatre types de tâches qui s'effectuent sur un tel système :

1. la distribution d'information : par exemple les informations d'aide à l'utilisation du logiciel, les informations de type calendaires, etc. ;
2. la gestion des ressources pédagogiques d'apprentissage telle que la personalisation de l'interface utilisateur ;
3. les mises à jour et le choix du mode d'enseignement, synchrone ou asynchrone ;
4. la gestion de la classe virtuelle à travers par exemple le traçage des apprenants, la gestion des profils des apprenants, etc.

Pour chaque système de gestion d'apprentissage considéré dans l'étude, ces tâches ont été décomposées en sous-tâches, qui ont à leur tour été exprimées sous la forme d'exigences fonctionnelles. Les exigences fonctionnelles obtenues réprésentent en réalité les problèmes que sont

censés résoudre les systèmes de gestion d'apprentissage. Cette démarche a permis d'identifier des patrons, répondant en guise de solutions, aux problèmes identifiés.

Cinq groupes de patrons ont été identifiés :

1. les patrons d'accès qui décrivent comment les utilisateurs accèdent aux différentes ressources du système ;
2. les patrons d'apprentissage qui décrivent comment les enseignants offrent l'aide à l'apprentissage ;
3. les patrons pédagogiques qui décrivent les différentes tâches d'enseignement ;
4. les patrons d'information qui décrivent comment les utilisateurs sont informés ;
5. les patrons d'administration qui décrivent les tâches d'administration.

Cependant la méthode proposée par Avgeriou et ses collègues (Avgeriou *et al.* , 2003; Goodyear *et al.* , 2004; Avgeriou & Zdun, 2005) consiste surtout en un répertoire de patrons mais ne précise pas la séquence chronologique des étapes à suivre par les développeurs de LEA.

2.2.5 Les patrons proposés par Coutinho Anacleto, Talarico Neto et Paula de Neris

Coutinho Anacleto et ses collègues (Coutinho Anacleto *et al.* , 2009), entre autres objectifs, ont souhaité améliorer l'utilisabilité des interfaces des ressources pédagogiques pour les étudiants et ont proposé un langage de trois groupes de patrons.

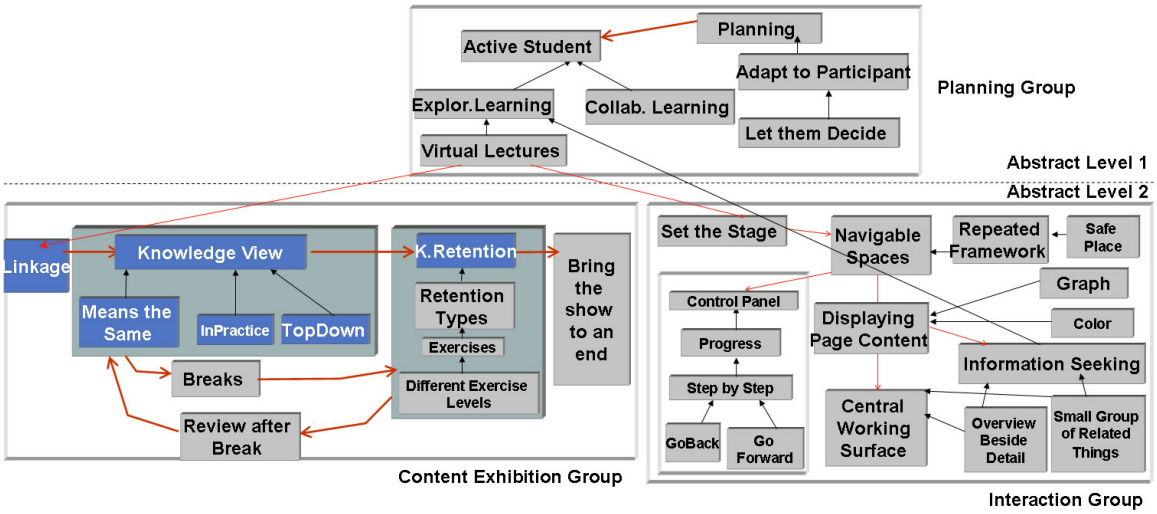


FIGURE 2.13 – Langage des trois groupes de patrons. (Coutinho Anacleto *et al.* , 2009)

Les trois groupes de patrons et leur mode de synergie apparaissent sur la figure 2.13, ainsi qu'il suit :

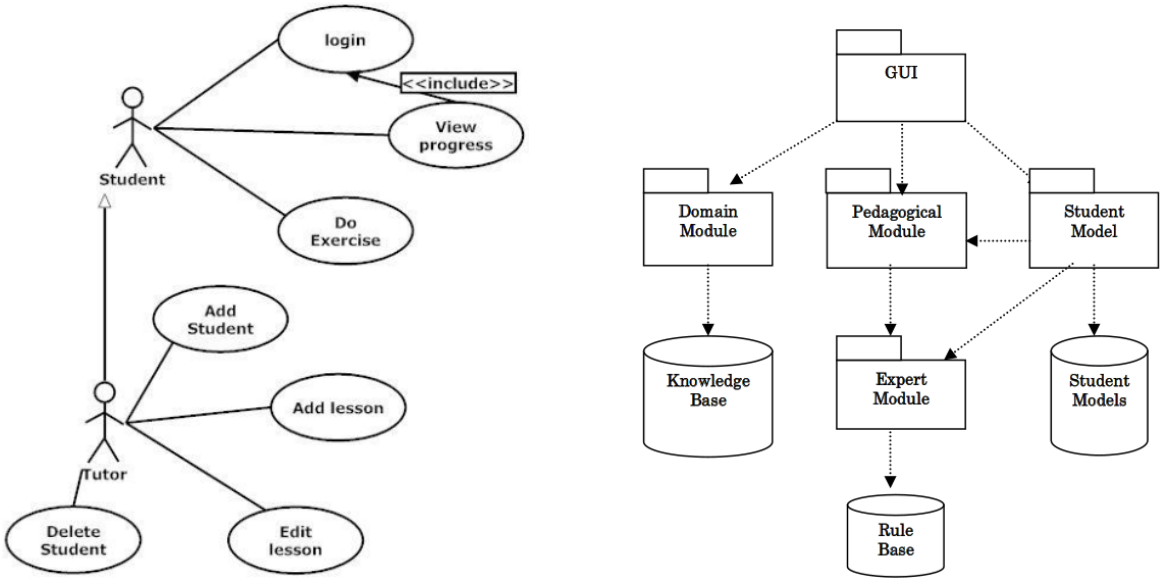
1. des patrons pédagogiques : ils servent pour la planification et la gestion des cours : ces patrons sont regroupés sur la figure, dans le rectangle nommé *planning group*.
2. des patrons d'interfaces utilisateurs : ils servent à définir l'interaction des utilisateurs avec le matériel pédagogique à travers une interface web. Ces patrons sont regroupés sur la figure, dans le rectangle nommé *Interaction group*.
3. des patrons hybrides : ils ont les caractéristiques des deux types de patrons précédents, décrivant l'approche d'intégration des stratégies du domaine cognitif dans le processus de conception du matériel pédagogique destiné à un système d'enseignement compatible avec le Web. Ces patrons sont regroupés sur la figure, dans le rectangle nommé *Content exhibition group*.

Coutinho Anacleto et ses collègues, bien que présentant leur travail comme étant une proposition d'un langage de patrons destiné à la conception d'un LEA adapté au Web, mettent surtout l'accent sur la capitalisation de la connaissance en matière de conception de ressources pédagogiques à l'intention des enseignants inexpérimentés.

2.2.6 Les patrons proposés par Zeid et Salah

Zeid et Salah (Zeid & Salah, 2010) proposent une approche de développement d'un LEA de la langue arabe, basé sur un langage de patrons.

Le système se base sur le scénario pédagogique illustré à la figure 2.14a et correspond à l'architecture représenté à la figure 2.14b.



(a) Scénario pédagogique (b) Architecture

FIGURE 2.14 – Représentation du système d'enseignement de la langue arabe. (Zeid & Salah, 2010)

Ce système se compose des modules suivants :

- un module générateur d'examen,
- un module de calcul des résultats d'examens,
- un module d'observation et de traçage des profils d'étudiants,
- un module de création et d'édition de cours,
- un module représentant l'interface du système créé.

Les auteurs utilisent un langage de patrons pour développer chacun des modules du système.

La proposition de Zeid et Salah présente les mêmes faiblesses que celle de Avgeriou et ses collègues présentée précédemment c'est-à-dire que, d'une part, elle consiste seulement en un répertoire de patrons sans guide d'utilisation et, d'autre part, elle ne précise pas la séquence chronologique des étapes à suivre par les développeurs de LEA.

2.2.7 Les patrons proposés par Turani et Calvo

Turani et Calvo (Turani & Calvo, 2006) proposent une collection de 14 patrons résultant de la modélisation de travaux sur des scénarios de discussion par logiciel de messagerie instantané, de remue-méninges, de débats, d'écriture sur tableau électronique ou intelligent, des séances de votes, etc.

Le modèle fonctionnel proposé, Beehive, offre un cadre de synergie entre les concepteurs pédagogiques et les ingénieurs logiciels, tel qu'illustré à la figure 2.15.

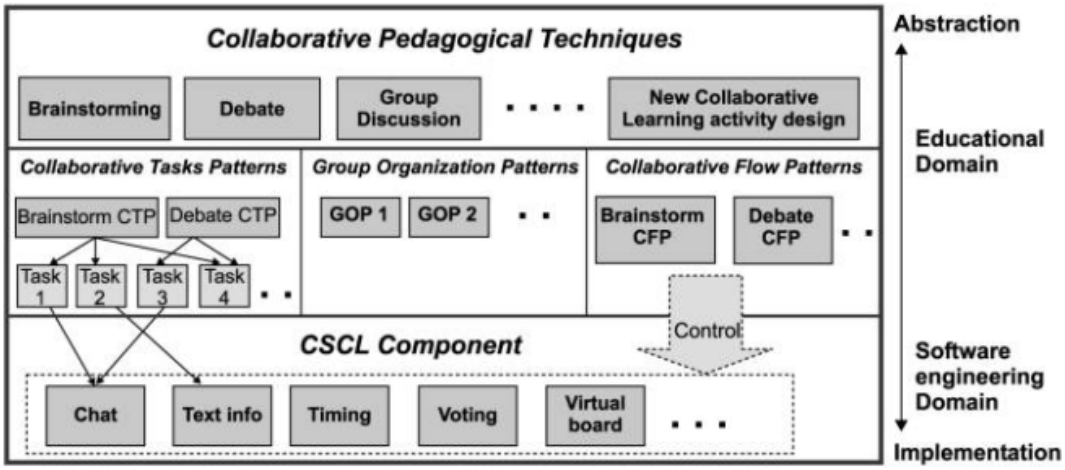


FIGURE 2.15 – Architecture fonctionnelle de Beehive. (Turani & Calvo, 2006)

Cette figure montre, à son sommet, l'ensemble des activités pédagogiques considérées pour la réalisation de leur modèle, puis ensuite les patrons qui ont été déduits de ces activités. Elle montre enfin comment ces patrons ont été implémentés dans le cadre d'un processus

d'ingénierie logicielle aux fins d'obtenir un LEA. L'idée présentée dans le cadre de ce travail est intéressante, mais les auteurs bien qu'utilisant des concepts issus du domaine de l'ingénierie logicielle s'adressent davantage aux experts en pédagogie.

2.2.8 Les patrons proposés par Retalis, Georgiakakis et Dimitriadis

Les travaux de Retalis et ses collègues (Retalis *et al.* , 2006) apportent la réponse à la question suivante : *Comment est construit un langage de patrons pour un LEA ?* Ils proposent une approche, à la fois ascendante et descendante, pour identifier les patrons de conception pour les LEA, en ayant pour objectif leur réingénierie en particulier dans le cadre de la construction d'un langage de patrons pour les systèmes collaboratifs asynchrones.

Leur méthodologie est fondée sur l'analyse des comportements et habitudes des utilisateurs du LEA à travers ses différents scénarios. Cette analyse permet d'effectuer la correspondance entre les différents attributs du système et les besoins des utilisateurs. La figure 2.16 illustre l'approche qu'ils ont utilisé. Elle comprend quatre grandes étapes que sont l'analyse fonctionnelle du système en cours de conception, la définition du scénario pédagogique, l'implémentation, et la construction du langage de patrons pour le système en cours de conception.

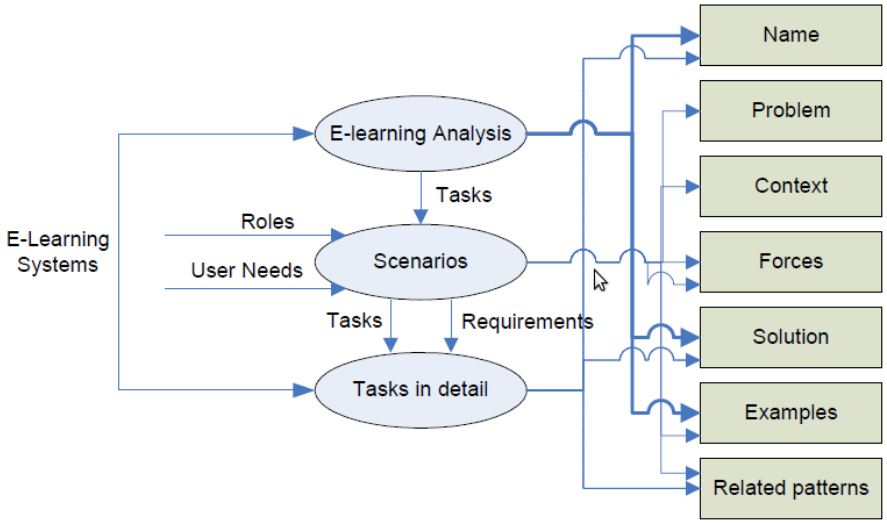


FIGURE 2.16 – Approche et format de patrons. (Retalis *et al.* , 2006)

2.2.9 Les patrons proposés par Iacob

Iacob (Iacob, 2011) propose des patrons pour la conception de systèmes collaboratifs pour activités synchrones, dont peut se servir une équipe de travail dont les membres sont distribués géographiquement. Elle identifie les meilleures pratiques de conception de systèmes collaboratifs synchrones. L'auteure a mis en évidence les patrons pouvant être identifiés pour la conception de systèmes collaboratifs synchrones. Pour cela, elle a d'abord procédé à une enquête par sondage auprès de 13 équipes de développeurs de logiciels pour recenser les défis que posent la conception de systèmes collaboratifs synchrones et leur a demandé de proposer

des instances d'interfaces utilisateur résolvant les problèmes identifiés. Ensuite elle a procédé à l'analyse de 20 logiciels collaboratifs synchrones pour tenter d'identifer les problèmes précédemment répertoriés. Les problèmes ayant le plus haut degré de récurrence et les solutions qui leur ont été trouvées ont été référencées dans le catalogue des patrons.

Iacob a ensuite proposé une ontologie pour représenter les relations d'un patron donné avec tous les autres patrons qui lui sont liés.

Iacob a aussi évalué l'impact de l'utilisation des patrons qu'elle a proposé par les développeurs de logiciels. Les données collectées dans le cadre de cette évaluation sont le nombre de fois qu'un même problème a fait l'objet d'une requête par les concepteurs, le nombre de fois qu'une solution a été appliquée, le nombre de fois que la collection de patrons a été parcourue, etc. (Iacob, 2012).

Toutefois bien que Iacob propose un langage de patrons ses travaux ne visent pas à proposer une démarche séquentielle de conception de systèmes collaboratifs. De plus les patrons qu'elle propose s'appliquent à une seule catégorie de LEA.

2.2.10 Les patrons proposés par De Moura Filho

De Moura Filho dans ses travaux (de Moura Filho, 2007; de Moura Filho & Derycke, 2007; de Moura Filho, 2009) propose une démarche de scénarisation pédagogique basée sur les patrons et inspirée de l'ingénierie dirigée par les modèles, représentée à la figure 2.17. Il a conçu un éditeur de patrons de scénarios pédagogiques à l'intention des enseignants qui s'intègre dans l'environnement de développement Eclipse. Cet éditeur respecte la syntaxe des patrons pédagogiques et les patrons sont écrits en langage naturel.

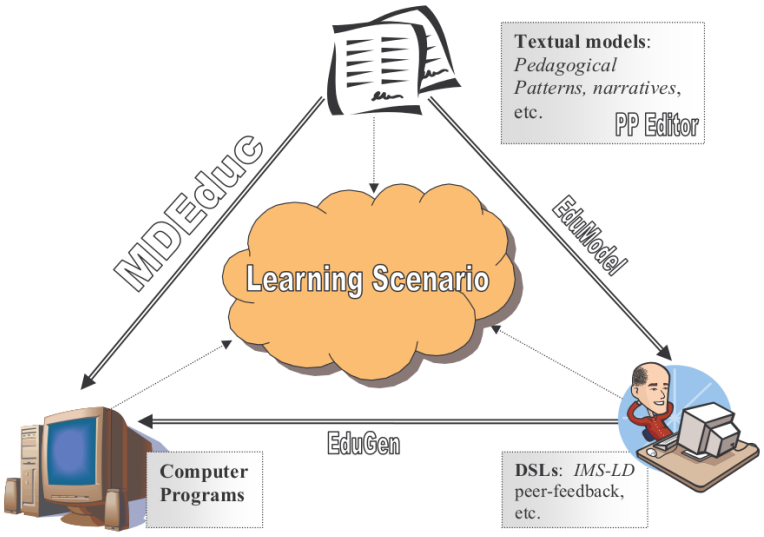


FIGURE 2.17 – Processus d'intégration du modèle MDEduc dans le processus d'ingénierie pédagogique. (de Moura Filho, 2007)

Cependant, les travaux de De Moura Filho consistent beaucoup plus en la définition d'un langage de modélisation adapté aux besoins des enseignants concepteurs qu'à ceux des ingénieurs en développement logiciel.

2.2.11 Les patrons proposés par Hadjerrouit

Le travail réalisé par Hadjerrouit (Hadjerrouit, 2007) peut être comparé à un chaînon entre le travail réalisé par les experts en pédagogie qui proposent des modèles d'ingénierie pédagogique présentés à la section 2.1 et celui des experts en ingénierie logicielle qui conçoivent des LEA.

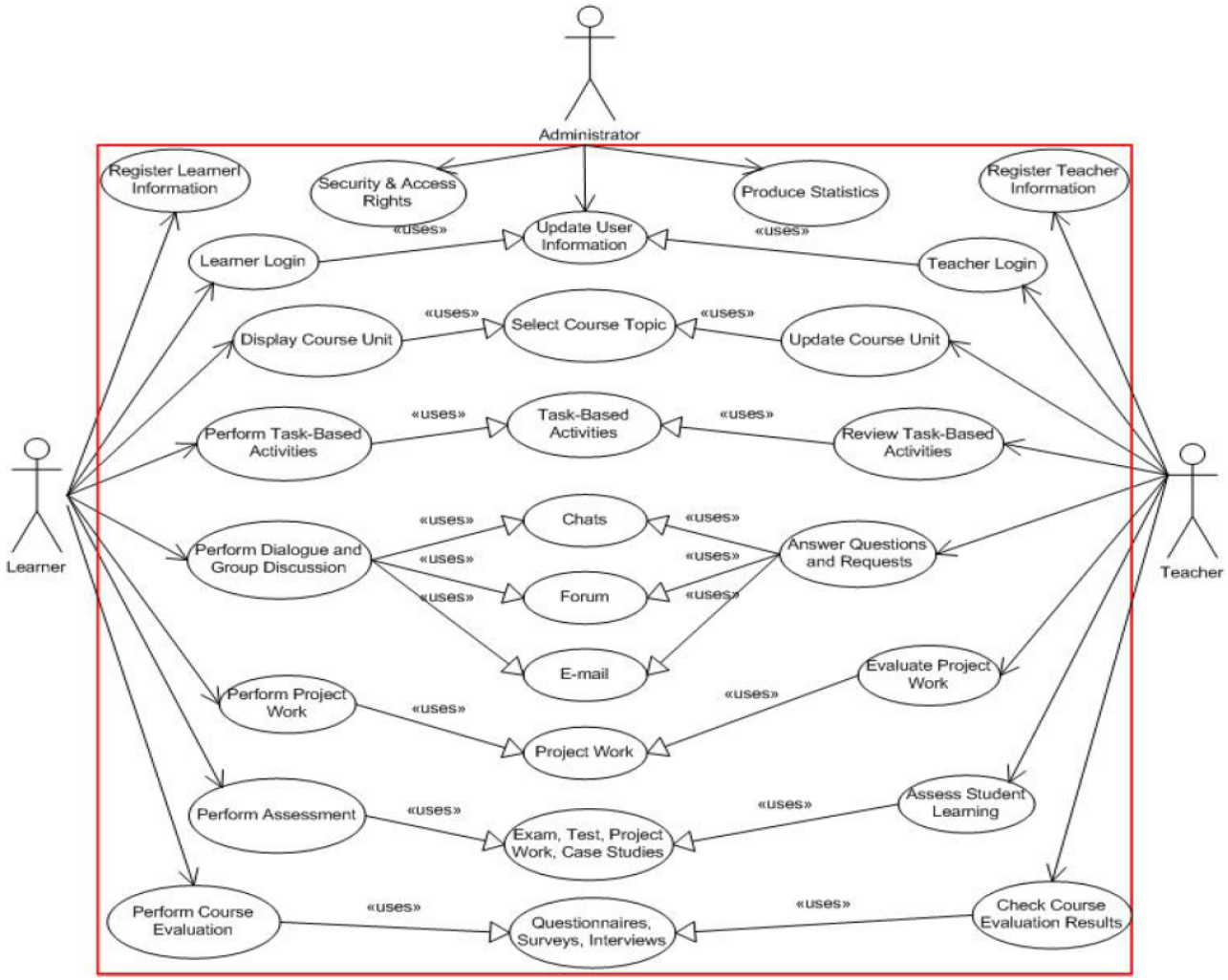


FIGURE 2.18 – Modélisation d'un scénario pédagogique. (Hadjerrouit, 2007)

Il propose une approche systématique intégrale de développement d'un LEA en proposant une démarche de traduction des exigences pédagogiques en une plateforme d'enseignement et d'apprentissage. Pour cela, Hadjerrouit part de l'observation d'un scénario pédagogique dont les exigences fonctionnelles sont illustrées à la figure 2.18, se sert des principes pédagogiques théoriques et aboutit à un LEA.

L'auteur analyse d'abord les théories psychologiques de l'enseignement et de l'apprentissage. Il propose ensuite un cycle de vie de l'apprentissage en trois étapes : conceptualisation (interaction entre apprenants), construction (conception et combinaison des concepts), dialogue (test des concepts créés). Puis il décrit un processus de développement intégré d'un LEA en superposant le cycle de vie de l'apprentissage et celui du développement d'un logiciel. Il procède ensuite à la spécification des exigences et définit les contraintes technologiques. Enfin il propose l'architecture de son système.

La solution proposée par Hadjerrouit se distingue de toutes les autres solutions présentées du fait de son intégralité puisqu'elle propose une démarche couvrant toute l'étendue du processus de développement des LEA, soit allant des exigences pédagogiques à l'implémentation d'un LEA.

Cependant, cette solution manque de finesse et de continuité car elle débute bien par une énumération des théories de l'apprentissage et s'achève par l'implémentation d'un logiciel, mais entre ces deux extrémités, nous restons grandement sur notre faim car cette solution ne propose pas de cadre de synergie entre les experts en formation à distance, les experts en technologie éducative et les experts en ingénierie logicielle. Il est en effet important de définir le cadre d'une interface fonctionnelle qui permettra à ces trois domaines d'ingénierie de communiquer de façon efficiente. Ainsi on s'assurera du fait que les exigences et la connaissance issues du cadre théorique fondamental de l'enseignement et de l'apprentissage soient converties en artefacts compréhensibles mais aussi utilisables par les ingénieurs logiciels.

Par ailleurs, la solution de Hadjerrouit manque aussi d'un cadre rigoureux définissant la séquence chronologique des étapes à suivre par les ingénieurs logiciels pour développer des logiciels d'enseignement efficients. La solution de Hadjerrouit est donc incomplète.

Les différents travaux présentés ci-dessus permettent de constater que le concept *patron* est très utilisé dans le cadre de la conception des LEA.

Nous avons comparé ces travaux selon différents critères et avons présenté les résultats de cette comparaison dans la section suivante.

2.3 Analyse comparative des modèles basés sur le concept *patron*

Nous présentons dans cette section une analyse comparative des principales utilisations de patrons dans les travaux présentés dans la section précédente. Les travaux présentés sont les plus récents sur le sujet au moment de la rédaction de cette thèse, et ne semblent pas avoir faits l'objet de publications ultérieures. Les résultats de cette analyse ont permis de valider le constat établi à la fin de la section précédente sur l'utilisation très fréquente du concept

patron dans le cadre de la conception des LEA. Ces résultats ont aussi permis de préciser les avantages et les inconvénients de chacun des travaux présentés dans la section précédente. Ils nous ont aussi permis de définir notre problématique pour pouvoir par la suite apporter les bases de la contribution de cette thèse.

Les critères de comparaison retenus pour cette analyse comparative trouvent leur inspiration dans le travail de Winn et Calder (Winn & Calder, 2002) qui estiment qu'un patron a deux composantes de telle sorte qu'il correspond à la fois à des caractéristiques fonctionnelles et à des caractéristiques non fonctionnelles. Ainsi selon eux, les caractéristiques fonctionnelles des patrons constituent un outil d'aide à la décision et répondent à la question *«Quelle décision prendre dans un contexe donné ?»* tandis que les caractéristiques non fonctionnelles des patrons constituent un outil d'évaluation de la faisabilité de la solution documentée et répondent à la question *«Pourquoi telle décision a été prise ?»*. À titre d'exemple, dans le cas d'un langage de patrons, celui-ci doit être structuré suivant les idées et principes qu'il véhicule. Le caractère fonctionnel du patron s'exprime alors par le fait qu'il documente une solution à un problème. Concernant les caractéristiques non fonctionnelles, Winn et Calder estiment qu'un patron possédant une caractéristique non fonctionnelle constitue un outil d'évaluation de la faisabilité de la solution documentée (Winn & Calder, 2002).

Dans la première partie de cette section, nous avons comparé les utilisations faites de patrons dans les travaux présentés, d'abord selon les critères fonctionnels, puis ensuite selon les critères non fonctionnels. Nous les avons comparé par la suite selon la nature de la solution proposée par chaque patron. Nous achevons cette section en effectuant une synthèse de cette comparaison.

2.3.1 Comparaison selon les critères fonctionnels

Suite à l'idée de Winn et Calder exprimée précédemment, à la question *«Quelle est la caractéristique fonctionnelle fondamentale de chaque patron étudié à la section 2.2 ?»*, nous avons répondu en déclinant le critère de comparaison *Objectif* auquel répond chaque patron, avons ensuite défini le critère de comparaison *Type* pour décrire le type de patron proposé, et pour en comprendre le contexte scientifique avons aussi décliné le critère *Méthodologie suivie par ses auteurs* ainsi que le public cible du patron considéré, exprimé à travers le critère *Destinataires*.

Le tableau 2.1 indique pour chaque critère retenu la ou les questions auxquelles on souhaite répondre.

Critères	Questions
Objectif	Quel problème résoud la proposition ?
Méthodologie	Comment les auteurs ont-ils procédé ?
Destinataires	A qui s'adressent la proposition ?
Type	Quel est le type ou la catégorie de patrons utilisée dans l'élaboration de la proposition ? Est-ce un patron unitaire, ou un langage de patrons, ou un réseau de patrons ?

TABLE 2.1 – Liste des critères fonctionnels retenus.

Le tableau 2.2 présente les résultats de la comparaison selon les critères fonctionnels. Dans ce tableau, le critère *Objectif* permet de mettre en évidence deux domaines d'application des différents patrons repertoriés dans le cadre du processus de conception des LEA : ils peuvent être utilisés soit pour concevoir les aspects purement pédagogiques du LEA ou alors pour en concevoir les aspects purement logiciels. Le critère *Destinaires* confirme cela puisqu'il permet de mettre en évidence deux groupes de destinataires des patrons repertoriés, le groupe des enseignants et celui des développeurs logiciels, correspondants respectivement aux deux domaines d'expertise mentionnés précédemment. Le destinataire enseignant ici, au regard de la fonction de pédagogue qui définit le métier d'enseignant fait référence à l'expert en pédagogie.

Malgré cette dichotomie observée de façon intrinsèque au niveau des critères *Objectif* et *Destinataires*, le tableau 2.2 nous montre le fait que peu importe le domaine d'expertise d'où sont proposés les patrons, ceux-ci demeurent similaires les uns aux autres. Cela peut s'expliquer par le caractère unitaire de l'objectif majeur poursuivi par l'ensemble des auteurs considérés, quelque soit leur domaine d'expertise : l'amélioration des LEA.

Critères / Auteurs	Objectif	Méthodologie	Destinataires		Type
			Enseig	Devel	
Randria -malaka	Réingénierie d'un scénario pédagogique, constitution d'une mémoire des indicateurs partagés.	Analyse du comportement d'apprenants et l'analyse des traces d'usage (Usage tracking analysis).	x		Patrons
Derntl	Modélisation d'un scénario pédagogique.	Observation des situations d'enseignement et d'apprentissage.	x	x	Langage de patrons
Goodyear et al.	Modélisation et capitalisation de techniques pédagogiques, modélisation de scénarios pédagogiques.	Observation des situations d'enseignement et d'apprentissage, expérience personnelle.	x		Langage de patrons
Coutinho Anacleto	Améliorer la disposition du matériel pédagogique mis à la disposition des apprenants.	Observation des situations d'enseignement et d'apprentissage, questionnaires.	x		Langage de patrons
Avgeriou et al.	Capitaliser l'expérience des développeurs de LEA.	Analyse fonctionnelle de 14 LMS puis généralisation.		x	Langage de patrons
Zeid et Salah	Développer un LEA de la langue arabe.	Analyse des systèmes similaires existants puis généralisation.		x	Langage de patrons
Turani et Calvo	Approche de conception d'un LEA synchone.	Analyse de scénario pédagogique.		x	Langage de patrons
Retalis et al.	Approche de conception d'un LEA asynchone.	Analyse des comportements et habitudes des utilisateurs.		x	Langage de patrons
Iacob	Guide d'utilisation de patrons de conception.	Analyse des habitudes de néo-développeurs.		x	Langage de patrons
De Moura Filho	Conception d'un LEA.	Ingénierie Dirigée par les Modèles et les Languages Orienté Programmation		x	Patrons
Hadjerrouit	Approche systématique de développement des LEA.	Observation des situations d'enseignement et d'apprentissage.	x	x	Langage de patrons

TABLE 2.2 – Comparaison des utilisations des patrons selon les critères fonctionnels.

2.3.2 Comparaison selon les critères non fonctionnels

Concernant les critères non fonctionnels Winn et Calder estiment que tous les patrons doivent exprimer de façon minimale certaines caractéristiques (Winn & Calder, 2002). Pour chacune des caractéristiques, nous avons suggéré un mot ou une expression clé permettant de l'exprimer simplement :

a) *Facilité d'utilisation* : cibler les problèmes importants en s'intéressant à «comment faire»

et non à «pourquoi faire» ;

b) *Généricité* : être génériques pour s'adapter, par définition, à des problèmes récurrents ;

c) *Modularité, extensibilité* : avoir la capacité d'évoluer en étant ouverts et extensibles ;

d) *Documentation* : avoir un pouvoir d'expression.

Les critères non fonctionnels que nous avons retenu pour notre comparaison sont listés dans le tableau 2.3.

Critères	**Descriptions**	**Questions**
Généricité	Dégré d'adaptabilité du patron	Jusqu'à quel dégré le patron proposé, est-il générique ?
Modularité	Facilité à décomposer le patron en modules ou fonctions plus ou moins indépendantes	Jusqu'à quel degré le patron est-il modulaire ?
Extensibilité	Degré de facilité avec lequel des modules ou des fonctions peuvent être ajoutés au patron	Jusqu'à quel degré le patron est-il extensible ?
Facilité d'utilisation	Degré d'utilisabilité du patron	Quel est le degré d'utilisabilité du patron ?
Documentation	Guide d'utilisation du patron	Le patron est-il bien documenté ?

TABLE 2.3 – Liste des critères non fonctionnels retenus.

Chacun de ces critères est évalué sur une échelle à trois niveaux. Le niveau le plus bas, représentant une faible appréciation, est symbolisé par le caractère $+$ et le niveau le plus élevé, représentant une forte appréciation, est symbolisé par l'ensemble de caractères $+++$. L'ensemble de caractères $++$ représente un niveau d'appréciation intermédiaire.

Critères / **Auteurs**	Généricité	Modularité	Extensibilité	Facilité d'utilisation	Documentation
Randriamalaka	++	+	+	+	+
Derntl	+++	+++	+++	+++	+++
Goodyear et al.	+++	++	++	+++	+++
Coutinho et al.	+++	++	++	++	+++
Avgeriou et al.	+++	+	+++	+++	+++
Zeid et Salah	+	+	+	+	+
Turani et Calvo	+++	++	++	++	+++
Retalis et al.	+++	+	+++	+++	+++
Iacob	+	++	++	+	++
De Moura Filho	+	++	++	+	++
Hadjerrouit	+++	+++	+++	+++	+++

TABLE 2.4 – Comparaison des critères non fonctionnels des différents travaux référencés sur l'utilisation des patrons.

De ce fait, le tableau 2.4 permet de constater le fait, que parmi les patrons recensés dans notre étude comparative, certains présentent une expressivité fort appréciable comme en témoigne les scores obtenus pour chacun des critères considérés, ceci étant vrai aussi bien pour les patrons servant à concevoir les aspects pédagogiques du logiciel que ceux servant à en concevoir les aspects purement logiciels.

Les patrons ayant obtenu les meilleurs scores à l'issue de la comparaison effectuée dans le tableau 2.4 suscitent particulièrement notre intérêt en particulier ceux décrits dans les travaux de Derntl en ce qui concerne leurs aspects pédagogiques, et ceux décrits dans les travaux de Hadjerouit du fait de leur appartenance au génie logiciel.

2.3.3 Comparaison selon la nature de la solution proposée

L'analyse des travaux présentés à la section 2.2 nous a permis de constater le fait que malgré le caractère unitaire de l'objectif majeur poursuivi par l'ensemble des auteurs considérés, l'amélioration des LEA, la nature des patrons proposés dans leurs travaux varie selon le domaine d'expertise dont ils sont issus. Ainsi les patrons issus des modèles proposés par les expert en pédagogie ont tendance à être des patrons pédagogiques en l'occurrence des scénarios pédagogiques tandis que ceux issus des modèles proposés par les experts en génie logiciel tendent à être des patrons d'analyse, d'architecture ou de conception, logicielle.

Dans cette section, nous avons comparé ces travaux selon la nature des patrons qu'ils proposent. Nous avons retenu un certain nombre de critères qui sont présentés dans le tableau 2.5, ainsi que les questions auxquelles ils répondent.

Critères	Questions	Commentaires
Environnement technologique	Le patron prend-il en compte l'usage d'outils technologiques, tel qu'une suite bureautique ou un logiciel divers ?	
Scénarisation pédagogique	Le patron propose-t-il une solution aux problèmes rencontrés dans le cadre de la conception d'un scénario pédagogique ?	La scénarisation pédagogique prend en compte la liste des activités pédagogiques qui ont lieu dans le cadre du LEA.
Interactions entre acteurs tiers	Le patron proposé s'intéresse-t-il aux interactions entre les différents acteurs d'un système pédagogique ?	Les patrons proposés en lien avec l'interaction entre les acteurs permettent de définir le cadre d'intervention des acteurs dans le contexte pédagogique défini, leurs rôles, les ressources dont ils ont besoin et leurs interactions.
Mode de livraison	Le patron s'intéresse-t-il au mode de livraison de l'enseignement ?	L'enseignement peut effectivement être livré de différente manière, soit de façon individualisé, ou de façon collaborative, ou encore de façon synchrone ou asynchrone, etc.
Analyse logicielle	Le patron s'intéresse-t-il au processus d'analyse logicielle du LEA et propose-t-il une solution conséquente ?	
Conception logicielle	Le patron s'intéresse-t-il au processus de conception logicielle du LEA et propose-t-il une solution conséquente ?	
Développement logiciel	Le patron s'intéresse-t-il au processus de développement logiciel dans un environnement de programmation du LEA et propose-t-il une solution conséquente ?	

TABLE 2.5 – Liste des critères choisis selon la nature de la solution proposée.

Le tableau 2.5 présente les résultats de la comparaison de la nature des patrons proposés dans les différents travaux référencés à la section 2.2. On se rend compte à travers cette comparaison que les patrons visant à améliorer les LEA et proposés par les experts en pédagogie s'intéressent surtout à définir les bonnes pratiques en matière de scénarisation pédagogique ou de conception de ressources pédagogiques et à définir les modèles d'interactions entre enseignants et étudiants notamment, tandis que les patrons visant à améliorer les LEA et proposés par les experts en génie logiciel s'intéressent surtout à définir les bonnes pratiques en matière de développement logiciel.

Auteurs \ Critères	Environnement technologique	Scénarisation pédagogique	Interactions entre acteurs tiers	Mode de livraison	Analyse logicielle	Conception logicielle	Développement logiciel
Randriamalaka	x	x	x	-	-	-	-
Derntl	x	x	x	x	-	-	-
Goodyear et al.	x	x	x	-	-	-	-
Coutinho Anacleto et al.	x	x	x	-	-	-	-
Avgeriou et al.	x	-	x	-	x	x	-
Zeid et Salah	x	-	-	-	x	x	x
Hadjerrouit	x	x	x	x	x	-	x
Turani et Calvo	x	-	-	-	x	x	x
Iacob	x	-	-	-	x	x	x
Olavo de Moura Filho	x	-	-	-	x	x	x

TABLE 2.6 – Comparaison de la nature des patrons proposés.

Nous venons d'une part d'effectuer l'étude des principaux modèles de conception des LEA issus des sciences de l'éducation. Cela a permis de mettre en évidence la place du concept *patron* dans le processus de conception d'outils aux fins pédagogiques. D'autre part nous avons aussi effectué une étude comparative des principales utilisations du concept *patron* dans le processus de conception ou d'amélioration des LEA qui nous a permis de mettre en évidence l'utilité du concept dans le processus de conception des LEA.

Cet état de l'art a permis de mettre en évidence une problématique liée à la conception des LEA à laquelle la présente thèse apporte une réponse. Nous présentons la problématique et les objectifs de cette thèse dans la section suivante.

2.4 Problématique et objectifs de recherche

L'étude effectuée dans les sections précédentes des travaux auxquels nous nous sommes intérressés a permis de mettre en évidence quelques limites des modèles étudiés et de préciser l'objectif de notre recherche. Cette section présente donc la problématique de cette thèse ainsi que son objectif de recherche.

2.4.1 Définition de la problématique

Les résultats de l'analyse comparative effectuée dans les sections précédentes nous ont permis de relever certes les apports mais aussi les insuffisances de ces travaux provenant de plusieurs disciplines ne partageant pas à priori d'intérêt commun, en l'occurrence les sciences de l'éducation et l'informatique.

D'abord cela nous a permis de positionner cette thèse dans le processus d'ingénierie pédagogique. Nous avons effectivement vu qu'en général, les modèles d'ingénierie pédagogique prévoient une étape dite de conception logicielle dans le processus d'ingénierie pédagogique, à l'image du modèle ADDIE présenté au paragraphe 2.1.1, qui possède une étape *Conception* ou du modèle IEEE-LTSA présenté au paragraphe 2.1.8 qui possède une étape *Conception logicielle*.

C'est tout naturellement donc que nous situons cette thèse, par exemple au niveau de l'étape de *Conception* du modèle ADDIE, étape qui fait référence à la conception d'outils pédagogiques tel que les LEA ou de l'étape *Conception logicielle* de l'approche IEEE-LTSA.

Le positionnement de la thèse étant fait, nous nous interrogeons sur la portée des éléments récurrents observés dans les modèles, méthodes et approches présentés.

En effet, nous avons observé que les problèmes généralement rencontrés dans le domaine de l'enseignement sont le manque de motivation des étudiants, le manque d'expérience des enseignants, la difficulté pour ces derniers d'effectuer la séquentialisation de leurs cours, de savoir sur quels concepts insister, de choisir le matériel de cours adéquat, d'évaluer leurs étudiants, etc. (Haberman, 2006). Les patrons sont donc utilisés dans le domaine de la pédagogie pour déterminer les bonnes pratiques d'enseignement et d'apprentissage dans un domaine donné, et capitaliser l'expérience des pairs en vue de les rendre réutilisables pour résoudre les problèmes sus-mentionnés (Project, 2002).

Les patrons dans le domaine de la pédagogie permettent aussi d'évaluer le déroulement des activités d'enseignement et d'apprentissage et de proposer des solutions résolvant les problèmes généralement rencontrés dans ce cadre pédagogique.

Les patrons pédagogiques permettent encore d'extraire et de formaliser de l'expertise afin de la mettre à la disposition d'autres enseignants moins expérimentés, en l'occurence les nouveaux enseignants.

L'une des limites les plus fondamentales observée est le fait que les chercheurs, malgré leur volonté de proposer une solution guidant le processus de conception des LEA ont de la difficulté à s'exprimer sur les perspectives de ce processus allant au delà du cadre de leur domaine d'expertise alors que le processus de conception des LEA est holistique c'est-à-dire qu'il comprend de manière inclusive des étapes impliquant différentes expertises. Ainsi les travaux issus des sciences de l'éducation, malgré leur volonté de proposer des modèles de processus de conception de LEA se limitent à proposer des modèles de conception de scénarios ou de ressources pédagogiques, tandis que les travaux issus du domaine de l'informatique, malgré leur volonté de proposer des modèles de processus de conception de LEA proposent certes des modèles logiciels robustes sur le plan technologique mais décriés par les premiers car ne tenant pas compte selon eux des aspects théoriques fondamentaux de l'enseignement et de l'apprentissage.

Cette situation équivoque exprime de façon naturelle une absence d'interface entre ces deux disciplines que sont les sciences de l'éducation et l'informatique, directement impliqués dans le processus de conception d'un LEA, qu'il convient de résoudre. Ce problème d'interface et de continuité apparaît bien au niveau de la proposition faite par Hadjerrouit car sa proposition débute bien par une énumération des théories de l'apprentissage et s'achève par l'implémentation d'une plateforme, mais entre ces deux extrémités, cette solution ne propose pas de cadre de synergie entre les acteurs de l'équipe de conception du LEA, c'est-à-dire les experts en pédagogie et les experts en génie logiciel. Son approche est aussi incomplète car il manque à sa chronologie d'importantes étapes du cycle de vie d'un logiciel. Il manque aussi à sa solution un cadre rigoureux définissant la séquence chronologique des étapes à suivre par les ingénieurs logiciels pour développer des LEA efficients.

Quant à la proposition de Derntl, elle pourrait être complétée afin d'offrir aux experts en ingénierie logicielle une gamme complète de patrons pour la conception des LEA.

Ces observations nous ont conduit à définir l'objectif de recherche présenté dans la section suivante.

2.4.2 Objectifs de recherche

Les observations précédentes nous ont conduit à nous poser les questions de recherche suivantes :

1. ***Quelle approche de conception des LEA permettra de construire des LEA robustes satisfaisant aux exigences pédagogiques ?*** À cette question répond l'objectif de cette thèse qui est de proposer une démarche méthodique séquentielle et itérative, basée sur le concept de *patron*, pour concevoir un LEA. Il s'agira de montrer quelles doivent être les étapes séquentielles de conception d'un LEA répondant aux exigences pédagogiques. Cette approche multidisciplinaire permettra aux experts des disciplines concernées par le processus de conception d'un LEA de communiquer et de travailler en synergie afin de concevoir des LEA efficaces et efficients, prenant en compte les exigences pédagogiques.
2. ***Le concept* patron *peut-il aider à construire des LEA robustes satisfaisant aux exigences pédagogiques ?*** À cette question répond l'objectif de cette thèse qui est de proposer une approche de conception des LEA basée sur les patrons. Le concept de *patron* offre des perspectives intéressantes à la solution que nous avons choisie et que nous souhaitons pourvue des attributs de réutilisabilité, de généricité et de portabilité. La finalité de notre solution est la possibilité de capitalisation de la somme de connaissances acquise par les experts en pédagogie, les experts en technologie éducative, et les experts en ingénierie logicielle à travers une interface robuste. Un cadre de communication entre ces experts issus de différents domaines de connaissance nécessite en effet d'être modélisé.

Les caractéristiques des patrons mis en évidence dans ce chapitre, et le fait que les patrons soient très utilisés dans les modèles d'ingénierie pédagogique nous incitent à considérer leur utilisation dans un modèle d'ingénierie pour LEA. Effectivement l'analyse comparative, que nous venons d'effectuer de l'utilisation du concept patron dans le processus de conception des LEA, montre que certains problèmes motivant l'utilisation des patrons dans les modèles d'ingénierie pédagogique sont aussi récurrents du côté de l'ingénierie logicielle. En effet en matière de conception de LEA, les problèmes généralement rencontrés concernent le manque d'expérience des concepteurs et développeurs de logiciels, ce qui rappelle sans équivoque le manque d'expérience des enseignants et des concepteurs de ressources pédagogiques évoqué précédemment et pour lequel le scénario pédagogique en tant que *patron* est utilisé. Le manque d'expérience des concepteurs et développeurs de LEA influe directement sur la performance et l'efficacité des logiciels conçus. Les patrons devraient donc être utilisés pour capitaliser l'expérience des concepteurs et programmeurs plus expérimentés afin de permettre à leur pairs de gagner en temps et en performance.

Pour construire donc notre approche de conception, nous nous baserons d'une part sur les travaux réalisés par Hadjerouit en proposant un processus permettant aux experts en pédagogie de recueillir les exigences pédagogiques et de les transmettre aux experts en génie logiciel dans un format que ces derniers peuvent exploiter. D'autre part nous complèterons l'approche qu'il a proposée afin d'obtenir une approche complète, séquentielle et itérative. Nous nous baserons aussi sur le répertoire de patrons pédagogiques proposé par Derntl afin de proposer aux concepteurs de logiciels un répertoire de patrons, qui fera partie intégrante de l'approche de conception proposée. En effet les patrons proposés par Derntl et présentés à la section 2.2.2 sont très diversifiés et couvrent une multitude de cas. Ces patrons constituent un riche répertoire et sont très bien documentés. Les patrons proposés par Hadjerrouit et présentés à la section 2.2.11 sont également dignes d'intérêt car l'auteur propose une démarche intégrale de conception d'un LEA. Ces patrons sont aussi bien documentés. La solution proposée par Hadjerrouit se distingue de toutes les autres solutions présentées puisqu'elle propose une démarche couvrant toute l'étendue du processus de développement des LEA, soit allant des exigences pédagogiques à l'implémentation d'un LEA. Mais il lui manque le cadre d'une interface fonctionnelle qui permettra à tous les domaines d'ingénierie impliqués dans le processus de développement des LEA de communiquer de façon efficiente.

2.5 Conclusion

Dans le cadre de ce chapitre, nous avons présenté les principaux modèles de conception des LEA issus des sciences de l'éducation et les principaux modèles de conception de LEA issus de l'informatique. Les résultats de l'analyse de ces modèles nous ont permis de mettre en évidence le concept *patron* utilisé dans les travaux présentés dans le but de concevoir ou d'améliorer

les LEA. L'utilisation du concept *patron* dans ces différents travaux exprime le souci des chercheurs de capitaliser l'expérience de la chaîne d'experts impliqués dans le processus de conception de ces systèmes. L'analyse effectuée nous a aussi permis d'identifier avec précision d'abord un problème de continuité entre toutes les disciplines concernées par la conception des LEA.

Ces observations nous ont conduit à nous poser la question de recherche principale suivante *Comment faire pour prendre en compte les exigences pédagogiques dans le processus de conception des LEA ?*. De cette question centrale dérivent les deux sous-questions suivantes que nous nous sommes posés : *Quelle approche de conception des LEA permettra de construire des LEA robustes satisfaisant aux exigences pédagogiques ?* et *Le concept* patron *peut-il aider à construire des LEA robustes satisfaisant aux exigences pédagogiques ?*

À la première sous-question répond l'objectif de cette thèse qui est de proposer une approche de conception des LEA basée sur les patrons et à la deuxième répond l'objectif qui est d'intégrer le concept *patron* dans cette approche. En effet, le concept de *patron* offre des perspectives intéressantes à la solution que nous formulerons et que nous souhaitons pourvue des attributs de réutilisabilité, de généricité et de portabilité. La finalité de notre solution est la possibilité de capitalisation de la somme de connaissances acquise par les experts en pédagogie, les experts en technologie éducative, et les experts en ingénierie logicielle à travers une interface robuste. Un cadre de communication entre ces experts issus de différents domaines de connaissance nécessite en effet d'être modélisé.

Le chapitre suivant est dédié à la présentation de la nouvelle approche de conception des LEA basée sur les patrons.

Chapitre 3

Approche de conception des LEA basée sur les patrons

Ce chapitre présente les différentes étapes de la conceptualisation de l'approche de conception des LEA basée sur les patrons, que nous avons proposée. Dans le chapitre précédent, nous avons présenté la comparaison de l'utilisation des patrons dans le processus d'ingénierie des LEA. Cet exercice de comparaison nous a permis d'identifier de façon particulière les travaux de Derntl (Derntl, 2005) et de Hadjerouit (Hadjerrouit, 2007) dont une analyse attentionnée a révélé les leviers utilisés pour mettre en évidence la problématique de cette thèse. Nous avons notamment mis en évidence le manque de détails et le manque d'expressivité des approches présentées dans ces travaux pour guider d'un bout à l'autre les ingénieurs logiciels dans leur tâche, et aussi le fait qu'elles sont incomplètes parce qu'il manque à leur chronologie d'importantes étapes. Ces conclusions effectuées au chapitre précédent nous ont amené à nous poser la question fondamentale suivante *«Comment faire pour prendre en compte les exigences pédagogiques dans le processus de conception des LEA ?»*. Pour répondre à cette question, nous avons proposé une nouvelle approche de conception des LEA, basée sur les patrons, dont l'objectif de ce chapitre est d'en présenter les différentes étapes et les aspects conceptuels.

La première partie de ce chapitre présente la méthodologie utilisée pour définir l'approche de conception proposée. Dans cette partie, nous introduisons le concept *patron* et décrivons le formalisme de représentation utilisé pour représenter les patrons identifiés dans le cadre de cette thèse.

La deuxième partie permet de décrire l'approche de conception proposée, une approche itérative constituée de huit étapes séquentielles. Nous présentons aussi dans cette partie un guide de recherche de patrons pour une bonne utilisabilité du répertoire de patrons proposé. Enfin, nous terminons ce chapitre par une conclusion.

3.1 Méthodologie pour la définition de notre approche de conception

Cette section présente les concepts fondamentaux de la contribution de cette thèse, la démarche scientifique et les outils utilisés pour les recherches qui y ont été effectués. D'abord, nous introduirons le concept *patron*, puis nous présenterons la classification des patrons, selon les domaines d'ingénierie ainsi que des exemples d'illustration.

3.1.1 Le concept *patron*

Il s'agit dans cette partie de définir d'abord le concept de *patron* puis d'introduire les différents types de patrons existants selon leur mode d'association. Cette section nous permet de répondre aux quetions *Qu'est-ce qu'un patron ?* et *Comment est représenté un patron ?*.

3.1.1.1 Définitions

Le concept *patron* trouve son origine dans le domaine de l'architecture (Avgeriou & Zdun, 2005; Devedzic & Harrer, 2005). Il y a été introduit par l'architecte Christopher Alexander dans son livre intitulé *The timeless way of building*. Selon Christopher Alexander, chaque patron décrit un problème récurrent dans notre environnement, et décrit la solution de ce problème de telle sorte que cette solution soit réutilisable autant de fois qu'on le souhaite (Gamma, 1995).

Cette définition, ou plutôt cette description, est la plus répandue aujourd'hui dans la littérature scientifique et relayée par un grand nombre de chercheurs tels (Beck & Johnson, 1994; Schmidt, 1995; Buschmann *et al.* , 1996; Tešanovic, 2004). Il s'agit bien d'une description du concept de *patron*, car ainsi que l'admettront Winn et Calder (Winn & Calder, 2002), définir de façon absolue et exhaustive ce concept est difficile voire impossible.

Les patrons doivent s'intégrer parfaitement à l'environnement dans lequel ils sont importés, et peuvent dans certains cas former des chaînes de patrons, c'est-à-dire des langages de patrons, avec des patrons pré-existants dans le milieu d'accueil. Les patrons doivent correspondre à la fois à des caractéristiques fonctionnelles et non fonctionnelles :

1. caractéristiques fonctionnelles : un patron possédant une caractéristique fonctionnelle constitue un outil d'aide à la décision (Winn & Calder, 2002). Dans le cas d'un langage de patrons par exemple, les patrons doivent être structurés suivant les idées et principes qu'ils véhiculent. Le caractère fonctionnel des patrons s'exprime ici par le fait qu'ils documentent une solution à un problème.
2. caractéristiques non fonctionnelles : un patron possédant une caractéristique non fonctionnelle constitue un outil d'évaluation de la faisabilité de la solution documentée (Winn & Calder, 2002). Dans ce cas, les patrons doivent répondre à certaines exigences :

a) cibler les problèmes importants en s'intéressant à «comment faire» et non à «pourquoi faire» ;

b) être génériques pour s'adapter, par définition, à des problèmes récurrents ;

c) avoir la capacité d'évoluer en étant ouverts et extensibles ;

d) avoir un pouvoir d'expression.

Comme dit précédemment, un patron n'est jamais une entitée isolée dans un environnement, mais est toujours lié à un autre patron (Winn & Calder, 2002). Cette relation d'interdépendance entre des patrons crée un maillage hierarchisé qui prend le nom de *langage de patrons*, illustré à la figure 3.1. Ce maillage de patrons a pour but de résoudre un même problème, dans un contexte donné. Si le patron constitue une solution générique et réutilisable, à un problème donné, un langage de patrons constitue donc une collection de telles solutions, ou de patrons, interagissant entre eux à différents niveaux pour résoudre un problème complexe donné.

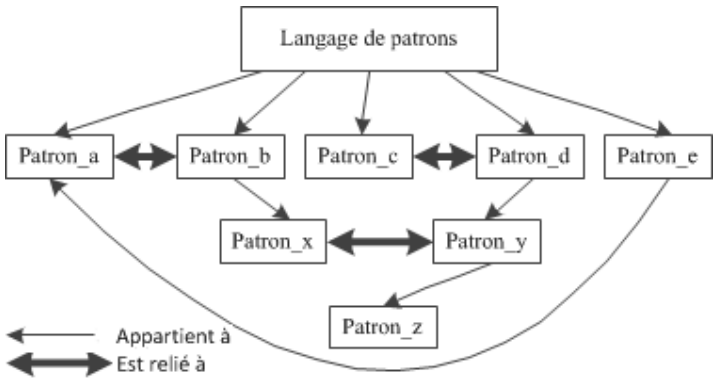


FIGURE 3.1 – Un langage de patrons.

La relation d'interdépendance entre les patrons peut être exprimée par les relations suivantes :

1. Les patrons peuvent être utilisés pour définir d'autres patrons.
2. Les patrons se complètent.
3. Les patrons présentent des similitudes et des différences.

Un *réseau de patrons* est simplement un ensemble de langage de patrons.

Le concept de patron apparaît donc comme étant un paradigme d'un intérêt majeur en ingénierie logicielle puisqu'il permet de bénéficier et de partir de la connaissance et de l'expérience d'autres personnes qui se seront déjà employées à comprendre le contexte et la solution à un problème méritant une nouvelle réflexion. De plus l'intérêt des patrons est d'autant accru qu'ils sont aussi réutilisables et adaptables.

3.1.1.2 Représentation des patrons

Les patrons peuvent être représentés à l'aide d'une représentation ou description textuelle, ou à l'aide d'une représentation graphique. Les formats de description textuelle des patrons sont à distinguer des patrons eux-mêmes. Un format correspond à une norme de description textuelle des patrons. Plusieurs formats de patrons existent. Les plus populaires sont d'une part le format *Alexandrien* qui dérive directement de la théorie formulée par Christopher Alexander sur le concept de patron et d'autre part le format qui a été proposé par le *Gang of Four* (Gamma, 1995). Nous présentons ces deux formats de description textuelle. Une brève comparaison de ces deux formats nous aidera à retenir les éléments de base nous permettant de définir un format de description textuelle de patrons adapté à notre besoin.

Format Alexandrien

Le format Alexandrien, selon l'appelation proposée par Berczuk (Berczuk, 1994), est le plus populaire de tous les formats de patrons existants. Sa caractéristique majeure est le fait qu'il propose une description assez générique du patron (Riehle & Züllighoven, 1996). Les patrons y sont décrits en forme de prose, en commençant par une description du contexte, suivie de la liste des problèmes pour lesquels le patron est applicable, puis de la présentation de la solution sous la forme d'une description générale. Ce format est décrit par huit éléments qui sont le nom, le contexte, les exemples, le raisonnement de conception, les forces et compromis, le contexte de résolution, le problème et la solution. L'objectif du format Alexandrien est de conduire l'utilisateur à générer des solutions au problème posé.

Format du Gang of Four (GoF)

Le format du GoF (Gamma, 1995) représente les patrons sous forme structurée. Contrairement au format précédent, il se focalise moins sur le moment d'application du patron que sur la structure et la dynamique du patron. Les éléments structurels du patron sont visuellement séparés grâce à l'utilisation d'entêtes et grâce à l'organisation du texte en paragraphes. Ce format est décrit par 17 éléments, dont quatre, les éléments nom, exemple, forces, problèmes, sont partagés avec le format Alexandrien. Le grand nombre d'éléments descriptif du GoF en fait un format plus descriptif que génératif et proche du langage naturel.

Le choix d'un format de représentation textuelle dépend de la quantité d'informations que contient le patron décrit. Pour les patrons complexes, la forme structurée sera préférée. Si on s'adresse à un public général, le format Alexandrien sera préféré. Ce dernier format constitue aussi le meilleur choix lorsqu'il s'agit d'identifier un problème et d'expliquer la manière dont le patron peut résoudre le problème, à cause de son caractère succint. Lorsque la description du patron contient beaucoup trop d'informations l'essence de l'information se trouve diluée dans la masse d'information. La forme structurée permet une navigation plus facile. Toutefois elle

renferme plus d'information à gérer que nécessaire. La forme narrative du format Alexandrien rend celui-ci plus facile à comprendre.

En plus de la description textuelle, une représentation graphique peut-être utilisée pour décrire des patrons. Contrairement au format de description textuelle, il n'existe pas de formalisme conventionnel pour la représentation graphique des patrons. Dans la litérature, les représentations graphiques des patrons sont généralement représentés à l'aide du formalisme d'un langage de modélisation connu comme UML ou à l'aide d'un formalisme spécifique défini par son auteur. Les formats de représentation de patrons choisis dans le cadre de notre thèse seront décrits en détails dans la section 3.2.

3.1.1.3 Classification

Le concept *patron* peut être appliqué à des processus dans divers domaines. L'application de ce concept au domaine de la pédagogie permet de décrire une solution visant à résoudre un problème récurrent bien cerné (Fincher & Utting, 2002; Carle *et al.* , 2007; Derntl, 2005; Finlay *et al.* , 2009). C'est le cas par exemple du scénario pédagogique qui permet de décrire d'une part une situation d'enseignement, c'est-à-dire qu'il décrit en détail le déroulement d'une séquence d'activité d'enseignement et d'apprentissage mais aussi apporte des solutions aux problèmes récurrents que peuvent rencontrer les enseignants dans l'exercice de leurs tâches.

En ingénierie logicielle, les patrons constituent des techniques réutilisables extraites de certaines applications et pouvant permettre de résoudre des problèmes redondants dans d'autres applications (Devedzic & Harrer, 2005). Toutefois ici, à l'inverse des autres domaines d'ingénierie, le concept de patron constitue une notion plurielle et a trouvé un large champ d'application. La notion plurielle et le large champ d'application du concept patron ont conduit la communauté des experts en génie logiciel à hiérarchiser les patrons en matière de génie logiciel. Les patrons sont ainsi ordonnés suivants différents critères de classification (Rudzki & Hammouda, 2004), dont les principaux sont :

1. classification par domaine d'application : dans ce cas, les patrons sont simplement classés suivant le domaine auquel ils sont appliqués, par exemple les applications temps réel, les systèmes embarqués, les interfaces utilisateurs, etc.
2. classification par paradigme : ici les patrons sont classés suivant les différents paradigmes de programmation logicielle, c'est-à-dire suivant qu'il constituent des patrons adaptés à la programmation orientée-objet ou adaptés à la programmation impérative. La différence entre les deux types de classification de patrons réside dans le fait que, dans le premier cas, les différents éléments constitutifs des patrons sont exprimés selon les paradigmes de la programmation orientée-objet (classe, objet, héritage, etc.) tandis que dans le deuxième cas, les éléments constitutifs des patrons sont exprimés selon la nature du problème résolu ou selon le domaine d'application des fonctions.

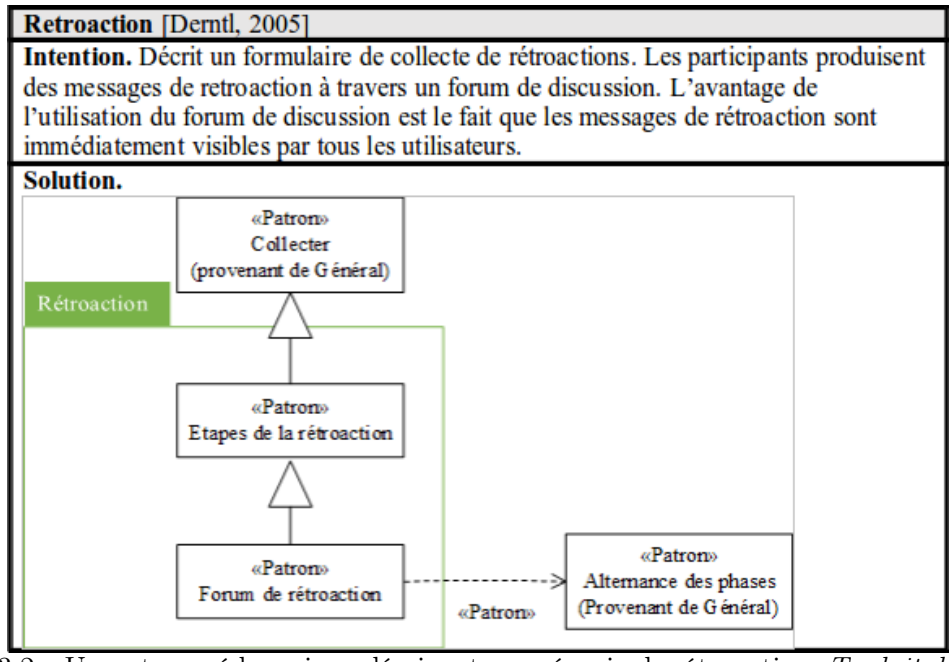


FIGURE 3.2 – Un patron pédagogique décrivant un scénario de rétroaction. *Traduit de (Derntl, 2005)*

3. classification par granularité : dans ce cas les patrons sont classés suivant le niveau d'abstraction des détails qu'ils expriment (Devedzic & Harrer, 2005; Rudzki & Hammouda, 2004), le niveau d'abstraction pouvant correspondre à tout point du processus de développement d'un logiciel. On distingue donc :

   a) les patrons d'architecture : ils expriment l'organisation structurelle du logiciel, ses différentes entités et les interactions entre elles.

   b) les patrons de conception : ils s'intéressent aux structures de conception réutilisables dans d'autres applications.

   c) les patrons d'analyse : ils s'intéressent eux au processus d'analyse des logiciels.

   d) les patrons de langage : ils décrivent des problèmes propres à un environnement de développement particulier et les solutions spécifiques adaptées.

3.1.1.4 Exemples

Dans cette section nous présentons des exemples de patrons décrits précédemment. Les commentaires sont effectués sur les figures présentées.

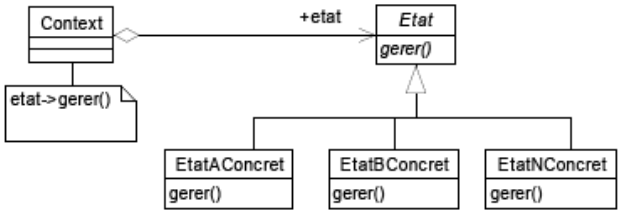
La figure 3.2 représente un patron pédagogique, *retroaction*, permettant par exemple, de décrire un processus de rétroaction d'un scénario pédagogique.

La figure 3.3 illustre, par exemple, le comportement d'un objet en programmation orientée-objet.

Patron État [Gamma *et al.*, 95]

Intention. Permet à un objet de modifier son comportement quand son état interne change. Tout se passe comme si l'objet changeait de classe.

Structure.



Constituants. La classe Contexte définit l'interface intéressant les clients (les états sont transparents pour l'utilisateur). Chaque contexte est lié à une instance qui définit son état courant. La classe Etat spécifie l'interface commune de toutes ses sous-classes ; il s'agit d'une classe abstraite. Les sous-classes concrètes de la classe État implantent le comportement spécifique associé à un état du contexte.

FIGURE 3.3 – Un patron décrivant le comportement d'un objet en programmation orientée-objet. (Hachani, 2006)

La figure 3.4 illustre un exemple de patron de déploiement d'une architecture logicielle.

Modèle-Vue-Contrôleur 2 [Turner *et al.*, 03]

Intention. Assure l'indépendance du modèle par rapport à l'interface. Les objets sont séparés en trois composants : le modèle, la vue et le contrôleur. Le modèle contient la logique et l'état de l'application. La vue représente l'interface utilisateur. le contrôleur réagit aux requêtes de l'utilisateur en effectuant les actions nécessaires sur le modèle.

Solution.

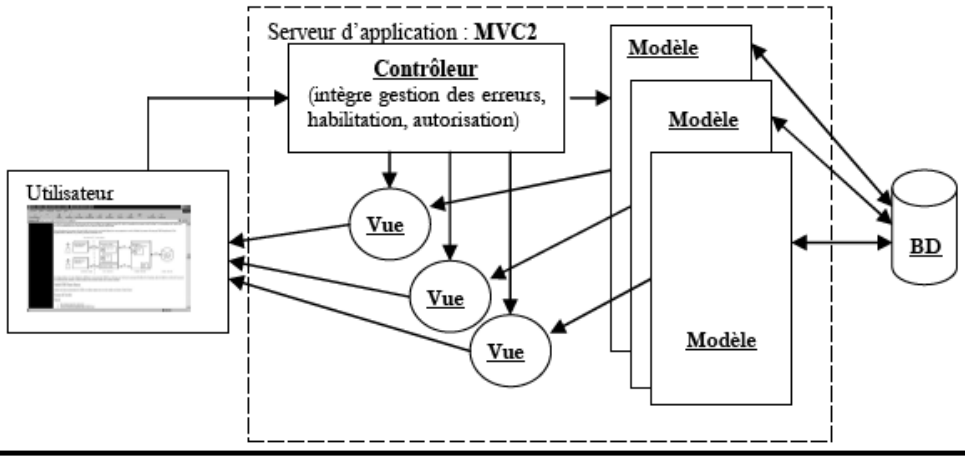


FIGURE 3.4 – Un patron décrivant l'archictecture d'un logiciel selon le paradigme MVC2. (Hachani, 2006)

En conclusion, peu importe le domaine, le concept *patron* est utilisé dans le but de capitaliser l'expérience des experts les plus expérimentés au profit des moins expérimentés. Il permet dans les deux cas de proposer des solutions réutilisables à des problèmes récurrents.

Dans la suite de cette thèse, bien que les patrons que nous avons proposés s'adressent aux experts en génie logiciel, nous y faisons référence sans utiliser cette classification. Pour faire référence à nos patrons, nous utiliserons le concept *patron* tout court, sans faire référence à un type de patron en particulier, car notre répertoire de patrons ne se focalise pas sur un type de patron, mais couvre l'ensemble des sous-domaines d'application du génie logiciel.

3.1.2 Formalisme choisi pour la représentation des patrons

Dans cette section, nous présenterons le processus de représentation des patrons. Nous présenterons d'abord le processus d'identification des patrons, suivi du format textuel et du format graphique que nous avons choisis pour représenter les patrons.

3.1.2.1 Processus d'identification des patrons

Le répertoire des patrons qui constituent notre approche de conception des LEA résulte d'un processus itératif comprenant la série d'activités présentée à la figure 3.5. Le point de départ de ce processus est la collecte des observables de la situation d'enseignement et d'apprentissage. Les observables sont les éléments qui permettent de décrire la situation d'enseignement et d'apprentissage.

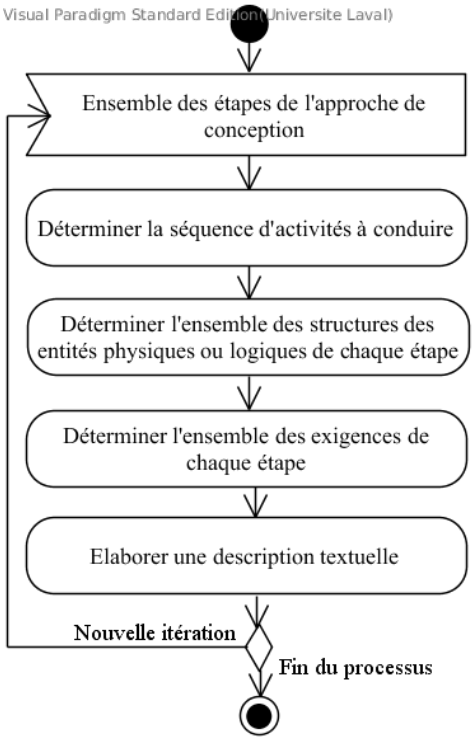


FIGURE 3.5 – Diagramme d'activités représentant le processus d'identification des patrons.

Après ce prélude, pour chacune des étapes de l'approche, le processus consiste à déterminer la séquénce d'activités à mener pour en définir les objectifs, ainsi que les actions à effectuer, les acteurs devant intervenir dans l'étape, et les livrables attendus à la fin.

Ensuite, le processus doit déterminer l'ensemble des structures des entités physiques ou logiques de l'étape pour mettre en évidence les informations de nature structurelle de l'étape, tel que les acteurs impliqués, les types de données, les fonctions, leurs inter-relations, leurs dépendances, etc. A l'issue de cette étape de détermination des structures des entités physiques ou logiques, des patrons peuvent être recensés. De façon formelle, dans le cadre d'une relation d'associativité, les structures définies au niveau de chaque étape peuvent être considérées comme étant les images des fonctions de l'étape. Ainsi à chaque fonctionnalité correspond au moins une structure et inversement. L'ensemble de ces structures constitue le format graphique des patrons.

Vient ensuite l'étape de détermination de l'ensemble des exigences de chaque étape. Celle-ci permet de déterminer pour chaque étape les contraintes dont il faut tenir compte, tel que les normes et standards, les compromis à faire, et autres besoins. Des patrons ont également été recensés à la fin de cette étape.

Enfin la dernière action à réaliser pour chaque étape est la description textuelle des patrons, à l'aide du format de représentation des patrons, aux fins de documentation. Cette description textuelle permet de présenter les patrons tout en faisant abstraction des différents formalismes utilisés.

Cette suite d'activités a permis de recenser un grand nombre de patrons qui ont permis de constituer notre catalogue de patrons. Dans la section suivante, nous présenterons le format textuel choisi pour représenter nos patrons.

3.1.2.2 Format textuel choisi

Les formats textuels présentés à la section 3.1.1.1 ne sont pas suffisamment expressifs pour convenir à notre contexte car ils ne permettent pas par exemple d'effectuer une recherche à l'aide de mots-clés. Dans le but de collecter le maximum d'informations sur les patrons proposés dans le cadre de cette thèse, nous définissons ci-dessous un format textuel pour représenter les patrons correspondants à nos besoins.

Le format que nous proposons conserve une bonne partie des informations des formats Alexandrien et GoF présentés à la section 3.1.1.1.

La définition de ce format a été motivée d'une part par la nécessité de collecter le maximum d'information sur les nouveaux patrons qui seront proposés, à l'intention des utilisateurs potentiels. D'autre part, le choix de ce format a aussi été motivé par l'expressivité insuffisante des formats de patron déjà existants pour décrire de façon satisfaisante les patrons que nous suggérons dans le cadre de notre approche de conception. En effet bien que la proposition majeure de cette thèse soit adressée prioritairement aux experts en architecture logicielle et aux développeurs, son caractère multidisciplinaire nous oblige à trouver le format idéal, c'est-

à-dire celui offrant un bon compromis entre d'une part la description sémantique du patron et d'autre part la description de ses spécifications. Nous avons donc repris dans ce nouveau format des éléments des formats précédemment présentés, et y avons ajouté un champs *Mots-clés* qui permettra aux utilisateurs de notre répertoire de patrons d'y effectuer une recherche en fonction des mots clés décrivant le patron et un champs *Evaluation* qui permettra de mettre en place une stratégie en vue de l'appréciation de l'efficacité des patrons.

Le format textuel choisi contient les champs d'information suivants :

- `Identifiant:` il s'agit d'une chaîne de caractères alpha-numérique qui permet d'identifier de façon unique un patron.
- `Nom:` il s'agit d'un identificateur du patron. Il est court et est composé de quatre mots au maximum.
- `Définition:` il s'agit de décrire le patron et ses composantes à partir de généralisations.
- `Mots-clés:` il s'agit des mots permettant d'effectuer une recherche dans le répertoire des patrons.
- `Motivation:` il s'agit de décrire un scénario modèle d'utilisation du patron. Ce scénario décrit un problème donné et la manière dont les structures du patron le résolvent. Cela permet de faire comprendre aux personnes devant utiliser le patron son bien-fondé.
- `Exemple:` un exemple concret d'utilisation du patron, dans un contexte spécifique.
- `Applicabilité:` il s'agit des contextes dans lesquels le patron peut être utilisé.
- `Conséquences:` il s'agit de l'implémentation dans des contextes différents pour mieux faire ressortir les conséquences d'utilisation du patron.

Notre format textuel permet de mieux définir les nouveaux patrons en mettant l'emphase à la fois sur les aspects techniques, technologiques et pédagogiques. Il permet aussi de mieux définir les nouveaux patrons en tenant compte de l'aspect multidisciplinaire sur lequel repose l'environnement technologique et technique des LEA.

Ce format est pertinent car il permet de décrire les objectifs et les contraintes d'utilisation des nouveaux patrons. De plus le format proposé permet la collecte de propositions, auprès des utilisateurs, pour résoudre les limites qu'ils constatent dans l'application des patrons. Il permet aussi à l'aide des mots-clés d'effectuer une recherche dans le répertoire des patrons.

3.1.2.3 Formalisme graphique choisi

Le processus de développement des LEA modernes se trouve à la croisée des disciplines ingénierie logicielle, pédagogie et ingénierie du multimédia. En conséquence, la taille de la documentation des systèmes et des LEA explose et avec elle sa complexité.

Pour pallier ces problèmes concernant la complexité grandissante des LEA, nous avons recherché un langage de modélisation correspondant à nos besoins. Le tableau 3.1 présente les caractéristiques du langage de modélisation recherché et les met en correspondance avec les caractéristiques du langage de modélisation SysML [1] (Roques, 2013). SysML est un langage de modélisation dédié à la modélisation de systèmes complexes et défini par l'Object Management Group (OMG) [2], le même organisme en charge de la spécification du langage UML. SysML est fortement inspiré de la version 2 d'UML dont il reprend d'ailleurs un certain nombre de formalismes.

Tout d'abord, nous avions besoin d'un langage capable de représenter avec beaucoup d'expressivité les concepts de la programmation orienté-objet tel que le polymorphisme, l'héritage et l'encapsulation et qui soit aussi en mesure de gérer les contraintes de conception liées au caractère pluridisciplinaire des LEA. Nous avions aussi besoin d'un outil capable de pouvoir représenter le concept d'exigence, et les inter-relations pouvant exister entre les exigences. Enfin nous cherchions aussi un langage offrant une riche abstraction, et non centré sur la documentation mais plutôt sur les modèles, tout en minimisant la taille et la complexité des modèles produits. Le langage de modélisation UML [3] qui s'est imposé comme langage dominant en la matière correspond certainement au premier critère, c'est-à-dire qu'il permet de représenter effectivement les concepts de la programmation orientée-objet, mais ne permet pas de maîtriser les projets de conception complexes à la croisée de plusieurs disciplines comme celui de la conception d'un LEA. De plus, le langage UML n'est pas muni des formalismes nécessaires pour la représentation du concept d'exigences.

En matière d'ingénierie des systèmes requérant une approche multidisciplinaire, le choix, s'il en est un, d'une telle approche n'est guère varié. C'est tout naturellement que le langage SysML s'impose de lui-même ainsi que l'illustre le tableau 3.1.

1. SysML (Systems Modeling Language) : http://www.omg.org/spec/SysML/1.3/
2. OMG (Object Management Group) : http://www.omg.org/
3. UML : (Unified Modeling Language) : http://www.omg.org/spec/UML/2.5

Caractéristiques du langage de modélisation recherché	Caractéristiques de SysML
■ doit permettre d'utiliser une approche orientée modèles permettant de produire une documentation succinte ; ■ doit permettre de maîtriser la complexité de notre projet, celui de proposer une approche de conception des LEA basée sur les patrons ; ■ à travers ses différents modèles doit permettre de donner accès à une représentation abstraite des différents aspects du système ; ■ doit permettre de représenter les exigences d'un système aussi complexe qu'un LEA sans s'embarrasser des concepts de classes, d'objets ou d'héritage ; ■ doit être non centré sur la documentation mais plutôt sur les modèles ; ■ doit permettre d'offrir une riche abstraction tout en minimisant la taille et la complexité des modèles produits ; ■ doit être un langage de modélisation graphique.	✓ permet d'utiliser une approche orientée modèles ; ✓ permet de représenter les exigences du système ; ✓ permet à des acteurs de métiers différents de collaborer pour concevoir un logiciel pluritechnique ; ✓ permet la réutilisation de librairies ; ✓ permet d'obtenir une modélisation de très haut niveau indépendante des langages et des environnements ; ✓ permet de regrouper les spécifications, contraintes, et paramètres d'un système provenant de plusieurs domaines.

TABLE 3.1 – Correspondance entre les caractéristiques recherchées et celles de SysML

Notre objectif étant de proposer une approche de conception d'un LEA, il nous a paru indispensable de pouvoir décrire à la fois les aspects dynamiques, les aspects statiques ou structurels et les aspects fonctionnels de notre approche. A partir du formalisme du langage SysML, nous avons décidé de représenter les aspects fonctionnels en utilisant le formalisme du `diagramme des exigences` ; pour représenter les aspects statiques ou structurels nous avons utilisé le formalisme du `diagramme des blocs de définition de données` et pour représenter les aspects dynamiques de notre approche nous avons utilisé le formalisme du `diagramme d'activités`.

Dans cette thèse, nous avons utilisé l'outil de modélisation `Visual paradigm`[4] pour représenter les différents diagrammes du langage SysML. La version de l'outil utilisée est la version 11.2 et la licence d'utilisation est la licence académique standard.

Nous présentons ci-dessous uniquement les formalismes du langage SysML utilisés dans notre processus de modélisation, soit ceux du diagramme des exigences, du diagramme des blocs de définition de données et du diagramme d'activités.

Le diagramme des exigences

Le diagramme des exigences permet de représenter les hiérarchies d'exigences, ainsi que leurs relations de dérivation, de raffinement, de satisfaction et de vérification. Il permet de représenter les aspects fonctionnels du système en développement et est à ce titre un diagramme

4. http://www.visual-paradigm.com/

fonctionnel. Il permet aussi de décrire les exigences d'ordre technique exprimées par le client que doit satisfaire le système. Le diagramme des exigences peut exprimer une fonction que devra réaliser le système ou une condition de performance technique, physique, de sécurité, de fiabilité, d'ergonomie, etc. Il représente un ensemble dans lequel chaque élément, c'est-à-dire chaque exigence, est un patron.

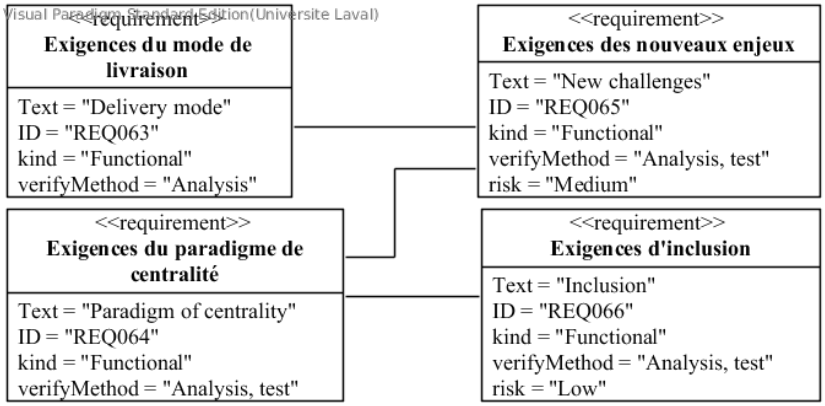


FIGURE 3.6 – Exemple d'un diagramme des exigences.

La figure 3.6 est un exemple de diagramme des exigences. Les symboles utilisés sont :

- Le rectangle marqué du stéréotype `requirement` comprenant un titre et des compartiments étagés regroupant des propriétés particulières permet de décrire les exigences correspondant à une capacité ou une contrainte à satisfaire par un système, ou une condition de performance technique, physique, de sécurité, de fiabilité, d'ergonomie, d'esthétisme.
- Les liaisons indiquent des relations pouvant lier les différentes exigences. Par exemple, une exigence peut dépendre d'une autre.
- Chaque exigence définie l'est à l'aide des attributs suivants, générés automatiquement par l'outil de modélisation `Visual paradigm` :
  - *Text :* il permet de décrire l'exigence sous une forme textuelle.
  - *ID :* il permet d'attribuer un identifiant unique à l'exigence.
  - *Kind :* il permet de définir le type *Performance* ou *Functional* ou *Interface* de l'exigence.
  - *VerifyMethod :* il permet de définir la méthode de validation de l'exigence *Analysis* ou *Demonstration* ou *Test* ou *Verification* de l'exigence.
  - *Risk :* il permet de définir le niveau de priorité de l'exigence *High* ou *Medium* ou *Low* de l'exigence.

Dans le cadre de notre thèse, le diagramme des exigences est un patron car il permet d'exprimer les conditions logiques dont il faut tenir compte pour la conception du LEA.

Le diagramme de définition des blocs

Le diagramme de définition des blocs est une hiérarchie de blocs reliés entre eux par des relations précisant la nature de leurs dépendances, leur hiérarchie ou les entités desquelles elles dérivent, etc. Le diagramme des blocs permet de représenter les blocs, leurs propriétés et leurs relations et définit un ensemble d'instances partageant les mêmes propriétés, mais possédant chacune une identité unique. Le bloc est une entité délimitée et décomposable qui encapsule des attributs (variables d'état), des propriétés et des opérations (procédures comportementales). Il contient aussi des ports permettant de représenter les flux d'échanges avec l'extérieur. Un diagramme des blocs permet de modéliser tout le système ou un élément matériel ou logiciel. Dans le cadre de ce chapitre, les aspects structuraux du LEA sont représentés à l'aide du diagramme de définition des blocs.

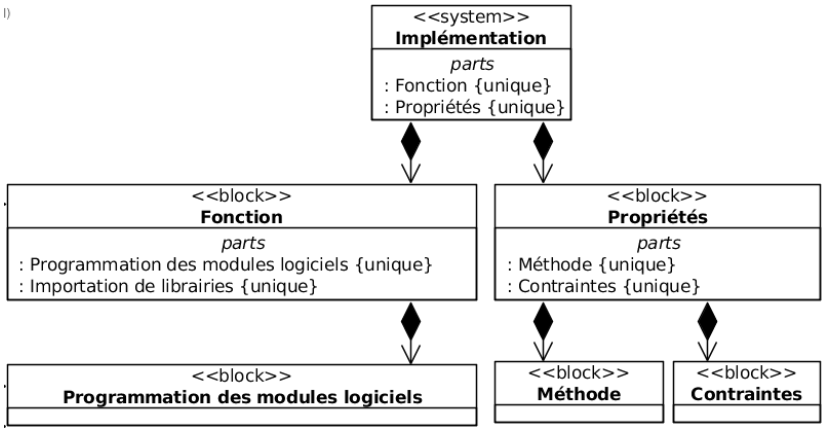


FIGURE 3.7 – Exemple d'un diagramme de définition des blocs.

Les symboles utilisés dans le diagramme de définition des blocs illustré à la figure 3.7 sont :

- Le rectangle marqué du stéréotype *block* comprenant un titre et des compartiments étagés regroupant des propriétés particulières : il permet de représenter la hiérachie du système et de définir et classifier les composants du système en blocs.
- Les flêches orientées avec losange plein : elles indiquent le fait que le *block* supérieur possède le *block* inférieur et que ce dernier est un raffinement du *block* supérieur.

Dans le cadre de notre thèse, le diagramme de définition des blocs est un patron car il permet de représenter les structures des patrons.

Le diagramme d'activités

Le diagramme d'activités, comme le montre la figure 3.8 permet de décrire de façon graphique un processus d'affaire ou un algorithme et ses activités parallèles et conditionnelles de façon détaillée.

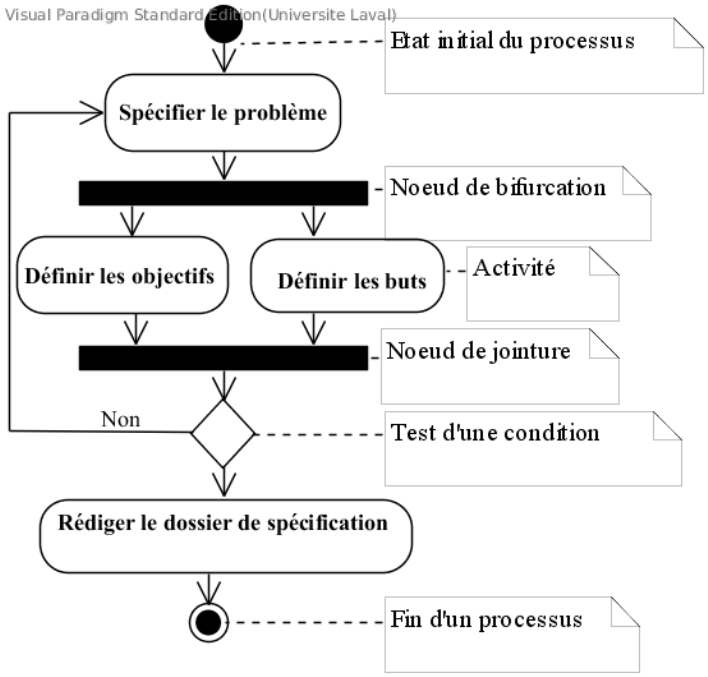


FIGURE 3.8 – Exemple d'un diagramme d'activités.

Dans le cadre de ce chapitre, le formalisme du diagramme d'activités est un patron car il permet de représenter les aspects comportementaux, donc les flux d'activités, ayant cours au niveau de chaque phase de l'approche de conception.

Dans la section suivante, nous présenterons la structure générale de l'architecture de la plate-forme logicielle.

3.2 Description de l'approche de conception proposée

Les résultats de l'analyse comparative des modèles de conception des LEA effectuée dans le chapitre 2, nous ont permis d'identifier plusieurs travaux utilisant le concept patron, en particulier celui de Hadjerrouit (Hadjerrouit, 2007). Dans la solution qu'il propose, Hadjerouit élabore un cadre de conception des LEA composé de neuf étapes, tel qu'illustré à la figure 3.9. Ces étapes sont chronologiques et itératives.

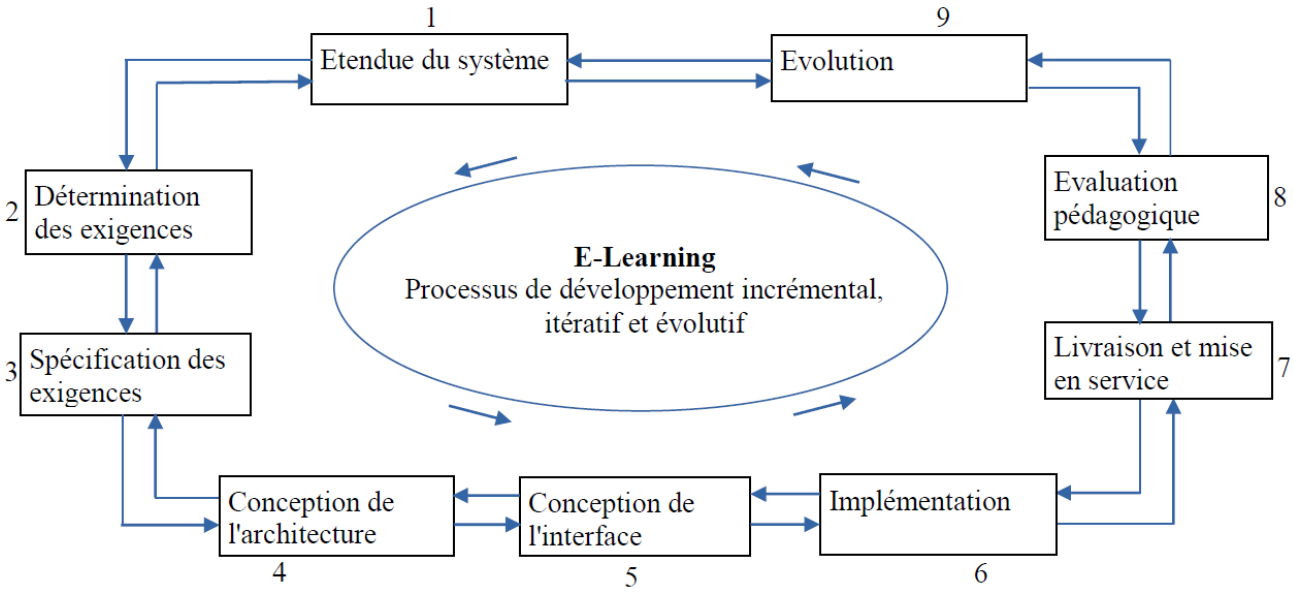


FIGURE 3.9 – Processus de développement d'un LEA (traduit de (Hadjerrouit, 2007))

Cependant, ce cadre proposé par Hadjerouit, bien que proposant des étapes pertinentes, n'est pas suffisamment expressif pour répondre aux besoins d'une approche de conception d'un LEA. Nous rappelons ici succintement les insuffisances que nous avons mises en évidence de la solution proposée par Hadjerrouit dans le chapitre 2 :

- manque de détails et de continuité : elle débute bien par une énumération des théories de l'apprentissage et s'achève par l'implémentation d'une plateforme, mais entre ces deux extrémités, cette solution ne propose pas de cadre de synergie entre les acteurs de l'équipe de développement du LEA, c'est-à-dire les experts en formation à distance, les experts en technologie éducative et les experts en ingénierie logicielle.
- manque d'un cadre rigoureux : il n'y a pas de cadre définissant la séquence chronologique des étapes à suivre par les ingénieurs logiciels pour développer des LEA efficients.

Outre ces insuffisances, la solution proposée par Hadjerrouit fait abstraction des phases fondamentales dans le cycle de vie d'un logiciel, suivantes : choix technologiques et validation. Il est en effet important de mentionner ces phases, et de les définir dans un cadre intégrant la définition d'une démarche séquentielle, se voulant être structurée et méthodique. Notons le fait que Hadjerrouit n'a plus effectué de publication suite à ce travail.

Bien que la solution de Hadjerrouit soit incomplète, nous l'avons utilisée et enrichie pour proposer notre approche de conception des LEA.

Dans cette section, nous décrivons cette approche. Ainsi, nous présenterons d'abord les différentes phases, puis les différents paquetages recensés. Nous terminerons ensuite la section en présentant le guide de recherche que nous proposons, permettant de retrouver plus facilement les patrons dans le répertoire.

3.2.1 Les différentes phases de l'approche proposée

Dans le cadre de la définition de notre approche de conception des LEA, nous nous évertuerons à détailler chacune de ses huit phases illustrées par la figure 3.10. Cependant, la mise en évidence des patrons et leur réutilisation ne concernera que les phases en rapport directement avec les étapes techniques de la conception logicielle, c'est-à-dire les phases 2 à 8. Ce choix a été fait pour tenir compte du fait que le résultat de cette thèse est destiné notamment aux architectes, concepteurs, et développeurs logiciels, tel que spécifié au chapitre 1. C'est ce qui explique l'absence de flêches partant ou venant des autres phases présentées à la figure 3.10.

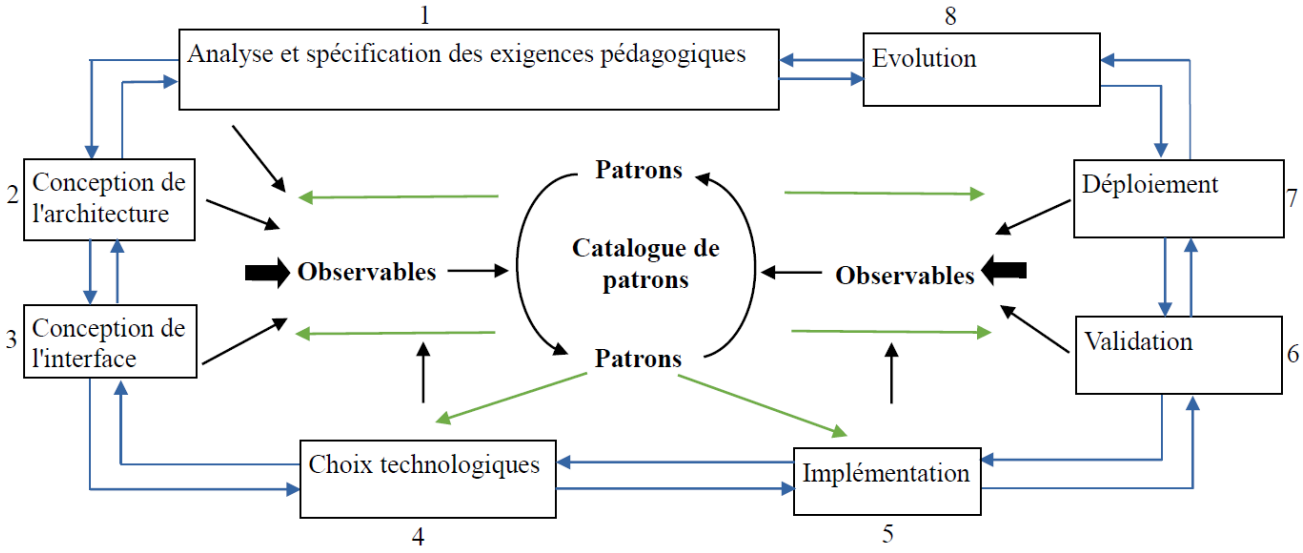


FIGURE 3.10 – Approche de conception d'un LEA proposée.

Pour rendre notre

Nous avons estimé opportun de regrouper les phases 1 à 3 de la solution de Hadjerouit, présentée à la figure 3.9 et même de lever certaines ambiguïtés apparaissant entre ces trois phases, en les spécifiant davantage. Cela a été fait d'une part pour des besoins d'optimalité, et d'autre part pour mieux souligner les phases que cible notre approche. Nous obtenons après ce travail de regroupement, de dissociation et d'élaboration notre approche de conception des LEA, plus expressive, moins ambigüe et plus complète.

Dans les paragraphes suivants, nous expliquerons la signification des différentes flêches apparaissant sur la figure 3.10, d'abord des flêches de transition entre phases de couleur bleue, reliant deux phases successives, puis celles de couleur noire (orientées vers le centre de l'image), et enfin celles de couleur verte (orientées vers l'extérieur).

3.2.1.1 Des phases chronologiques et itératives

L'approche de conception proposée est constituée d'un ensemble de huit phases formant une chaîne de progression chronologique et itérative. Ce processus est chronologique car la phase

précédente aboutit à la production d'un résultat utilisé dans la phase suivante et ainsi de suite. Par exemple à la fin de la phase 1 qui est celle de l'*Analyse et spécification des exigences pédagogiques*, le résultat produit est l'ensemble des caractéristiques permettant de définir le problème posé et le répertoire des exigences pédagogiques. Ces informations sont réutilisées au niveau des phases 2 et 3 qui sont respectivement celles de la *Conception de l'interface utilisateur* et de la *Conception de l'architecture*. Le processus est itératif car en cas de besoin il est possible de revenir à l'étape précédente à des fins de corrections ou de mises à jour. Par exemple, la figure 3.10 indique qu'à la phase 2, il est possible de retourner à l'étape précédente qui est celle de l'*analyse et spécification des exigences pédagogiques* soit par exemple afin d'y affiner la définition du problème ou de ses objectifs, ou alors à des fins de vérification simplement. Cette possibilité de cyclicité est exprimée sur la figure 3.10 par la matérialisation de deux flêches de sens opposé entre deux phases successives.

Ces phases illustrées sur la figure 3.10 sont présentées brièvement ci-dessous.

- **Phase 1 : Analyse et spécification des exigences pédagogiques**
  - **Buts :** définir le point de départ, les intentions et les objectifs réalisables et vérifiables du problème à résoudre ; étudier la faisabilité du projet ; rechercher toutes les connaissances et solutions proches du problème spécifié existantes ; exprimer et rassembler les besoins pédagogiques auxquels doivent répondre le logiciel en cours de conception ; rassembler le maximum d'idées libres et spontanées pour compléter le catalogue des exigences ;
  - **Acteurs :** analyste en informatique, ingénieur en formation à distance, expert en technologie éducative.
- **Phase 2 : Conception de l'architecture logicielle**
  - **But :** fixer les modalités d'assemblage des composants théoriques logiciels, définir de façon exhaustive les spécifications (architecturales, logicielles) du système ; identifier les performances du système, différents composants d'un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions ;
  - **Acteurs :** architectes et concepteurs logiciels.
- **Phase 3 : Conception de l'interface utilisateur**
  - **But :** fixer les modalités d'assemblage des composants théoriques de l'interface utilisateur et définir de façon exhaustive leurs spécifications ;
  - **Acteurs :** architectes et concepteurs logiciels, experts en interface personne-machine.
- **Phase 4 : Choix technologiques**
  - **But :** fixer les spécifications des matériels et logiciels nécessaires d'une part à la conception du LEA envisagé, et d'autre part à son futur déploiement ; identifier les

performances du système, des différents composants informatiques, leurs interrelations et leurs interactions ;

- **Acteurs :** architectes et concepteurs logiciels.

- **Phase 5 : Implémentation**
  - **But :** mise en oeuvre dans un langage de programmation des spécifications des dossiers d'architecture et d'interface utilisateur ;
  - **Acteurs :** développeurs logiciels.
- **Phase 6 : Validation**
  - **But :** établir la preuve documentée que les spécifications et exigences ont été respectées ;
  - **Acteurs :** ingénieur en formation à distance, expert en technologie éducative, architectes et concepteurs logiciels, experts en interface personne-machine, développeurs logiciels, client.
- **Phase 7 : Déploiement**
  - **But :** remise du logiciel à l'entité demandeuse ;
  - **Acteurs :** développeurs logiciels, client.
- **Phase 8 : Evolution**
  - **But :** faire évoluer le logiciel pour y ajouter de nouvelles fonctionnalités ou le corriger pour rectifier des erreurs ;
  - **Acteurs :** ingénieur en formation à distance, expert en technologie éducative, architectes et concepteurs logiciels, experts en interface personne-machine, client.

3.2.1.2 Des phases observées et analysées

Une flêche de couleur noire orientée vers le centre de la figure part de chaque phase de l'approche. Cette flèche indique que des informations collectées suite à l'observation du monde réel, c'est-à-dire suite à l'observation de cette phase, dans un environnement réel, permettent la mise en évidence de patrons puis la constitution d'un catalogue de patrons. L'environnement réel, ou le monde réel, est constitué des documents livresques ou disponibles sur le Web, de la base de connaissances personnelles de l'équipe en charge du développement du LEA, et des situations réelles d'enseignement et d'apprentissage.

Une situation d'enseignement décrit en détail le déroulement des activités d'enseignement et intègre donc la prise en compte de la connaissance à enseigner, l'organisation de la connaissance en séquences d'enseignement, la conception des matériaux pédagogiques, et la mise en scène de la manière dont cette connaissance sera transmise. L'observation puis l'analyse des situations d'enseignement permettra de savoir comment elles peuvent être formalisées sous la forme de patrons et réutilisées dans le contexte des LEA. La finalité de la prise en compte des situations

d'enseignement dans les LEA est de permettre à l'enseignant de tirer parti au maximum des atouts technologiques les plus modernes du Web afin de dispenser ses cours à des étudiants dont les habitudes de vie et comportementales sont fortement influencées par ces technologies.

Une situation d'apprentissage quant à elle décrit le déroulement d'une séquence d'activités d'apprentissage. Dans ce cadre, l'observation puis l'analyse des situations d'apprentissage doivent aider les concepteurs et développeurs de LEA à évaluer l'effet de l'environnement logiciel sur l'activité d'apprentissage des étudiants, à les formaliser sous la forme de patrons en vue de les réutiliser dans le contexte des LEA, et enfin à mettre en évidence les contraintes pédagogiques.

3.2.1.3 Des phases restructurées par une approche écologique

Les flêches de couleur verte sont orientées vers l'extérieur. Elles partent du catalogue de patrons vers les phases de l'approche indiquant le fait que des patrons sont utilisés, voire réutilisés, provenant de notre catalogue de patrons ou de catalogues de patrons existants dans la littérature scientifique pour structurer ces phases en vue de la conception du LEA. La couleur verte exprime le caractère écologique de notre approche soit le fait de réutiliser des patrons existants afin de proposer une approche structurée et méthodique de conception des LEA.

3.2.1.4 Niveaux d'abstraction proposés

Pour une bonne organisation du répertoire de patrons identifiés, nous l'avons organisé en trois niveaux d'abstraction logiques représentés à la figure 3.11. Ces trois niveaux expriment le niveau de granularité des patrons recensés et sont : le niveau N_0 issu de l'obervation du *Monde réel* et qui est le premier niveau de l'abstraction, suivi du niveau supérieur N_1 qui est le niveau des *Patrons*, et du niveau N_2 qui est celui des *Paquetages*, le plus haut niveau d'abstraction. Nous distinguons ainsi respectivement le niveau du *Monde réel* à partir duquel l'ensemble des observations a été effectuées, le niveau des *Patrons* qui contient les résultats des observations effectuées et le niveau des *Paquetages* où chaque paquetage constitue un regroupement de patrons identifiés dans le cadre d'une même phase de l'approche de conception. Les paquetages forment donc des langages ou des réseaux de patrons.

N_0 : le monde réel

Au niveau du monde réel, les observateurs que sont les concepteurs des LEA observent les situations d'enseignement et d'apprentissage. Ils observent les activités de cours, d'évaluation des étudiants. Ils observent aussi les acteurs des situations, etc. Ces observations ont permis de mettre en évidence un premier niveau de généralisation constitué d'observables N_0 afin de structurer cet ensemble d'instances en patrons, formalisés sous la forme de diagrammes d'exigences, de blocs et d'activités. Les patrons ainsi identifiés se situent au niveau N_1 de l'échelle d'abstraction proposée.

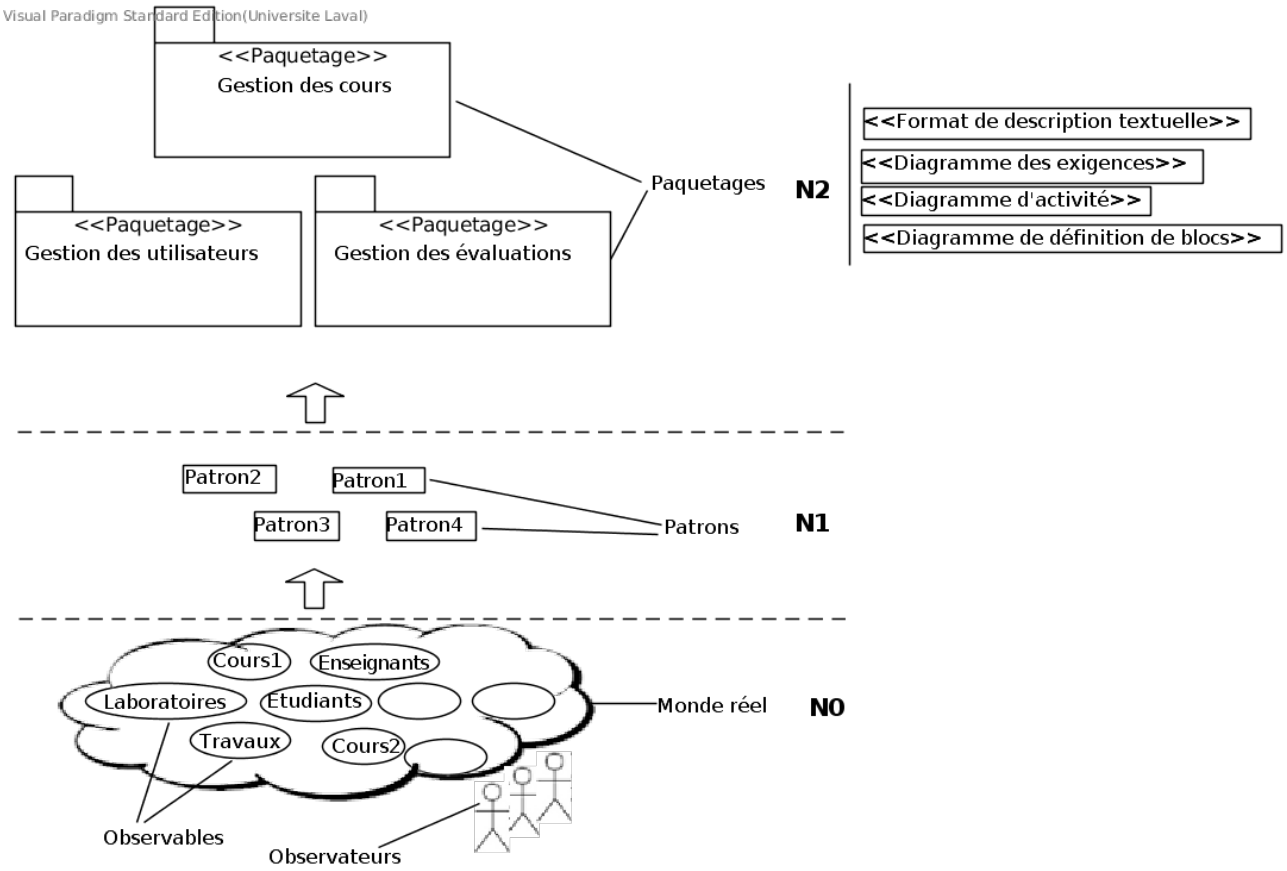


FIGURE 3.11 – Les niveaux d'abstraction utilisés.

N_1 : les diagrammes des exigences, de définition des blocs, et d'activités

Les diagrammes des exigences, de définition des blocs, et d'activités sont des patrons issus des propriétés des instances recensées au niveau N_0. Le processus de généralisation suivi a abouti à l'identification de ces patrons génériques. Les patrons correspondants aux activités de chacune des phases de notre approche de conception ont été identifiés et sont présentés un peu plus loin. Pour chaque phase, nous avons défini tour à tour, dans le paquetage correspondant, un format de description textuelle, un diagramme des exigences, un diagramme de définition des blocs, et un diagramme d'activités.

N_2 : les paquetages

L'ensemble des patrons de niveau paquetage recensés est présenté dans le tableau 3.2.

Les paquetages de patrons sont présentés de façon plus détaillée dans le chapitre suivant.

Numéro	Nom du patron
P1	Exigences pédagogiques
P2	Acteurs
P4	Architecture logicielle
P3	Interface utilisateurs
P5	Choix technologiques
P6	Implémentation
P7	Validation
P8	Déploiement
P9	Evolution

TABLE 3.2 – Ensemble des patrons de niveau paquetage recensés.

Le paquetage de patrons est la structure de plus haut niveau de l'organisation hiérarchique des patrons que nous avons recensés. Le paquetage est un mécanisme permettant généralement de regrouper les patrons appartenant à une même famille de patrons. Il représente un module ou une fonctionnalité du LEA. Les paquetages peuvent être imbriqués les uns dans les autres.

Le niveau paquetage permet aussi de comprendre l'organisation logique des patrons : ceux-ci sont regroupés en paquetages correspondants logiquement aux phases de l'approche de conception. Notre décomposition en paquetages ne suit pas des critères fonctionnels. Les patrons sont regroupés en paquetages selon des critères purement logiques. Cela permet de générer une cohérence forte entre les éléments d'un même paquetage et un couplage faible entre paquetages. Chaque paquetage contient des patrons représentés à l'aide des différents formalismes du langage SysML que sont le diagramme des activités, le diagramme de définition des blocs, et le diagramme des exigences. A ces trois éléments, nous avons intégré au paquetage un quatrième formalisme ne faisant pas partie du formalisme SysML, afin d'améliorer l'expressivité des patrons : il s'agit du format de description textuelle de patron.

La forme générale de la structure du LEA est exprimée par la hiérarchie des paquetages et par le réseau de relations de dépendance entre paquetages.

La liste ci-dessous présente l'ensemble des paquetages définis, ainsi que la phase de la figure 3.10 à laquelle ils appartiennent. le lien paquetage-phase est exprimé dans la liste suivante par la relation (*paquetage*, *phase*) :

P1 (*Exigences pédagogiques*, *Analyse et spécification du logiciel d'enseignement et d'apprentissage*) : ce paquetage contient la liste de l'ensemble des besoins pédagogiques que doit vérifier le LEA.

P2 (*Acteurs*, *Toutes les phases*) : il contient la liste des acteurs du processus de développement du LEA.

P3 (*Architecture logicielle*, *Architecture logicielle*) : ce paquetage définit d'une part la liste des besoins que doivent satisfaire l'architecture logicielle, et d'autre part la liste des fonctionnalités du logiciel.

P4 (*Interface utilisateurs*, *Interface utilisateurs*) : ce paquetage définit d'une part la liste des besoins que doivent satisfaire les différentes interfaces du logiciel, et d'autre part la liste des fonctionnalités d'une interface.

P5 (*Choix technologiques*, *Choix technologiques*) : il contient la liste des fonctions de la phase des choix technologiques.

P6 (*Implémentation*, *Implémentation*) : il contient la liste des fonctions de la phase d'implémentation.

P7 (*Validation*, *Validation*) : ce paquetage contient la liste des fonctions de la phase de validation.

P8 (*Déploiement*, *Déploiement*) : il contient la liste des fonctions de la phase de déploiement.

P9 (*Evolution*, *Evolution*) : ce paquetage contient la liste des fonctions de la phase d'évolution.

L'ensemble de ces paquetages, leurs interrelations et interactions sont représentés sur la figure 3.12.

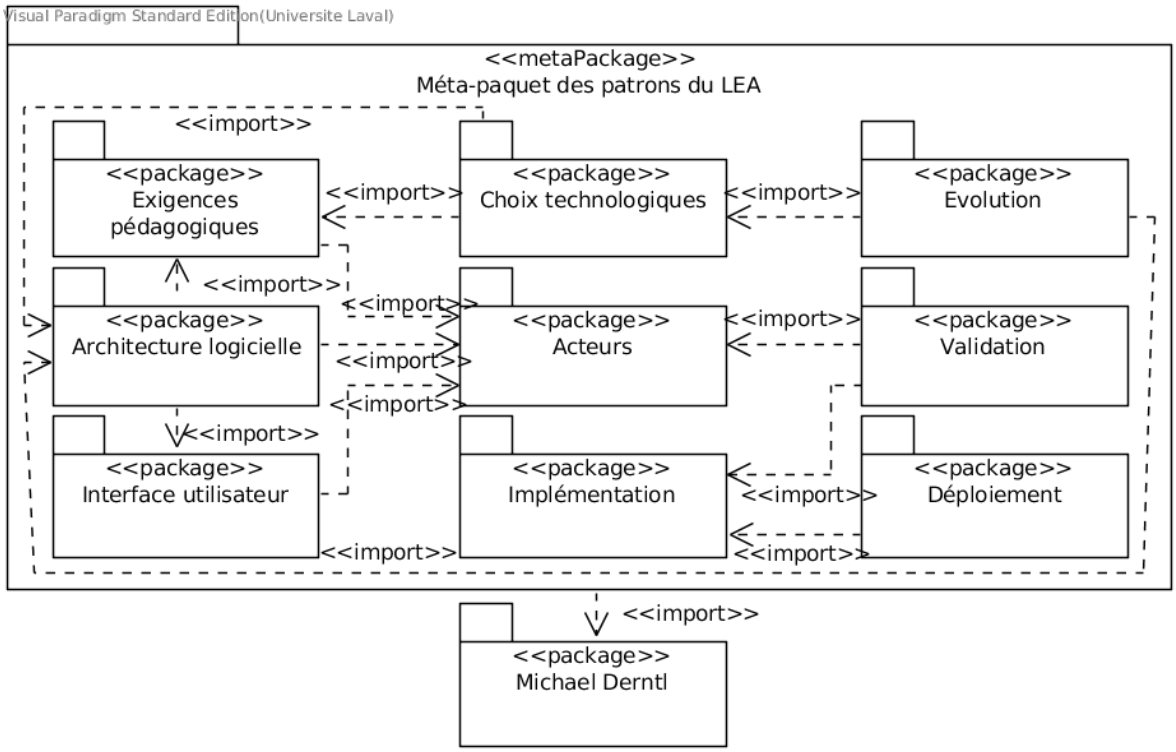


FIGURE 3.12 – Diagramme des paquetages du LEA.

Les différents paquetages de l'architecture logicielle sont liés par une relation d'importation, *import*, qui exprime les relations de dépendance entre eux. Ainsi cette figure indique par

exemple que le paquetage *Architecture logicielle* dépend du paquetage *Exigences pédagogiques* et *importe* par conséquent des patrons de ce dernier, et que le processus de développement du LEA, et par conséquent tous les paquetages, dépend du paquetage Michael Derntl dont il importe des patrons.

Les patrons recensés font partie d'une liste énumérative au sein de laquelle la séquence `P«Numéro»`, où `P` fait référence au terme `patron` et `Numéro` étant un nombre attribué au patron selon la chronologie d'apparition du patron dans la liste énumérative, désigne le numéro d'identification attribué au patron. Ainsi par exemple, `P15` est un numéro d'identification attribué à un patron.

Pour rendre le répertoire de patrons proposé dans cette thèse utilisable, et permettre de facilement trouver les patrons, nous avons proposé un *Guide d'utilisation des patrons*, que nous présentons dans la section suivante.

3.2.2 Guide de recherche de patrons

Pour une bonne utilisabilité du répertoire de patrons, nous avons proposé un guide de recherche de patrons. Ce guide consiste en un modèle d'exploration du répertoire de patrons basé sur un arbre de dépendance des patrons, représenté à la figure 3.13. Cet arbre a une profondeur de 4, correspondant aux différents niveaux d'abstraction des patrons présentés dans ce chapitre. Au niveau des feuilles de l'arbre se retrouvent les patrons unitaires c'est-à-dire ceux qui sont irréductibles tel que le patron *Controleur d'application* que l'on retrouve au niveau du diagramme des blocs de l'interface utilisateur ou le patron *Modèle d'intégration sémantique* que l'on retrouve au niveau du diagramme de définition des blocs de l'architecture logicielle. Au niveau hiérarchique supérieur immédiat, on retrouve des langages de patrons regroupés au sein de diagrammes de définition des blocs ou de diagrammes d'exigence. Au niveau hiérarchique supérieur, nous retrouvons les réseaux de patrons formés d'associations de diagrammes de définition des blocs ou de diagrammes d'exigence. La racine de l'arbre représente la racine du répertoire global de patrons proposés.

Il est possible d'accéder à un patron à un niveau quelconque de cet arbre en formulant une requête d'accès à un patron à l'aide d'un langage de manipulation de données tel que le langage SQL et de mots-clés obtenus après une réduction littéraire ou verbale d'une liste d'exigences. Par exemple, pour accéder au patron *Controleur d'application* que l'on retrouve au niveau du diagramme des blocs de l'interface utilisateur, et qui se trouve au niveau *feuille* de l'arbre de décision, nous pourrons formuler la requête suivante :

- en utilisant les mots-clés associés au patron : SELECT "patron" FROM "Conception" WHERE mots-clés="Exigences d'interface, affichage" ;
- en utilisant le nom de son package d'appartenance : SELECT "patron" FROM "Conception" WHERE package="Interface" and diagramme="Bloc" ;

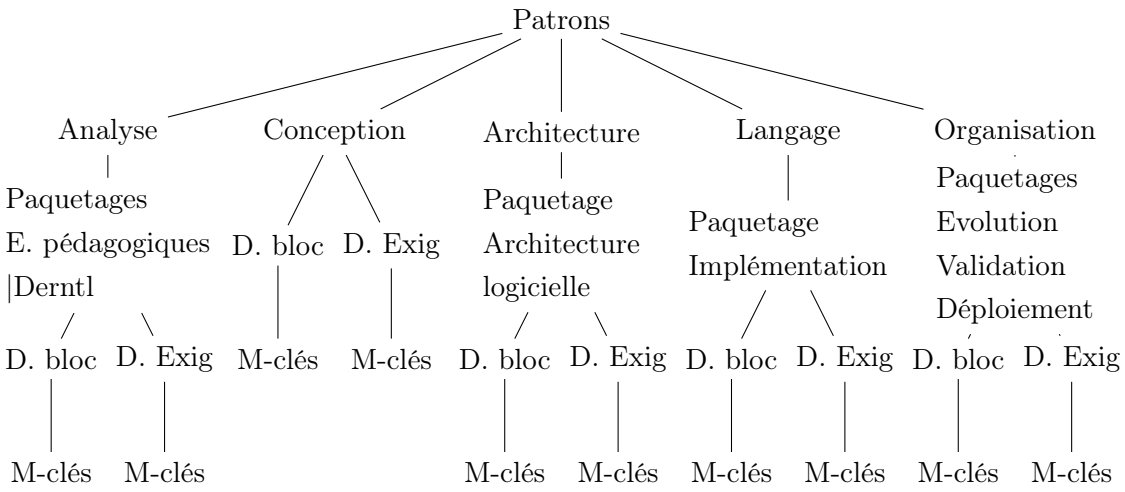


FIGURE 3.13 – Arbre de dépendance des patrons.

3.3 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle démarche de conception d'un LEA, constituée de huit étapes. Ce processus de conception des LEA proposé permet aux experts en pédagogie de reccueillir les exigences pédagogiques et de les transmettre aux experts en génie logiciel dans un format que ces derniers peuvent exploiter. L'approche que nous avons proposée est complète, séquentielle et itérative. Nous avons aussi présenté deux formalismes de représentation de nos patrons, un formalisme textuel et un formalisme graphique. A partir de ces différents formalismes nous avons proposé et présenté trois niveaux d'abstraction logiques des patrons identifiés. Enfin, nous avons proposé un guide de recherche de patrons qui permettra de rechercher et de sélectionner des patrons adéquats du répertoire de patrons de façon méthodique. Le prochain chapitre, sera consacré à la présentation du répertoire des patrons qui ont été identifiés dans le cadre cette approche.

Chapitre 4

Répertoire des patrons regroupés en paquetages

Ce chapitre est dédié à la présentation détaillée du contenu des différents paquetages avec leurs patrons, issus de notre approche de conception des LEA. Comme mentionné dans le précédent chapitre, nous avons utilisé différents éléments pour mettre en évidence les structures des paquetages et des patrons. Nous avons d'abord utilisé un format textuel, présenté au chapitre précédent, qui permet de décrire en langage naturel les éléments qui constituent le paquetage. Nous avons repris les diagrammes d'exigences et de définition de blocs du langage SysML. Pour rappel, le diagramme des exigences permet de décrire l'ensemble des conditions inhérentes au paquetage et le diagramme de définition de blocs permet de représenter les structures des paquetages et des patrons.

La mise en évidence des aspects structuraux de chaque paquetage, à l'aide de ces différents formalismes, nous a permis de constituer le catalogue de patrons correspondant. Ces patrons sont organisés en paquetages correspondants aux différentes phases de l'approche de conception que nous avons proposée, à l'exception toutefois du paquetage *Acteurs*. En effet, le paquetage *Acteurs* ne correspond à aucune phase de l'approche proposée. Il permet simplement de décrire pour chaque acteur ses caractéristiques, ses compétences, les autorisations qu'il possède pour l'exécution de sa tâche, les contraintes qu'il doit respecter, ses préférences mais aussi ses responsabilités. Les caractéristiques comprennent les nom, rôle, type, position et fonction de chaque acteur. Le paquetage *Acteurs* permet aussi de décrire le flux d'interaction entre les acteurs que sont l'analyste fonctionnel, l'expert en pédagogie, l'expert en formation à distance, l'expert en technologie éducative, le concepteur logiciel, l'architecte logiciel et le développeur du logiciel. Pour nous focaliser sur la présentation des paquetages correspondants aux différentes phases de l'approche de conception, le paquetage *Acteurs* ne sera donc pas présenté de façon régulière au même titre que les autres paquetages.

Pour ce chapitre, nous avons adopté une présentation particulière, différente de celle des autres

chapitres et plus appropriée à son contenu. Chaque section principale, qui débute à une nouvelle page, correspond aux éléments descripteurs d'un seul paquetage. Ces éléments sont présentés en suivant soit la représentation textuelle, le diagramme des exigences et le diagramme de définition de blocs. Pour chacun de ces deux diagrammes, nous avons ajouté des exemples de patrons décrits à l'aide d'un diagramme de définition de blocs et d'un diagramme d'activités. Tous ces diagrammes ne se retrouvent pas dans la liste des figures du début de la thèse, car ils n'ont pas la même vocation que celles-ci. En effet, ils ne sont pas là seulement pour illustrer nos propos ; ils représentent les éléments structuraux du paquetage et se suffisent à eux-mêmes pour le décrire.

4.1 Le paquetage *Pédagogique*

Le paquetage *Pédagogique* regroupe les patrons permettant d'exprimer les conditions pédagogiques à prendre en compte dans le processus d'ingénierie des LEA. Ces conditions sont spécifiées sous la forme d'exigences pédagogiques. Le catalogue des exigences est produit suite à l'observation des situations d'enseignement et d'apprentissage. Cette observation a permis de mettre en évidence les patrons pédagogiques. Le paquetage *Pédagogique* importe des patrons des paquetages *Acteurs* et *Michael Derntl*. Il comprend un diagramme des exigences et un diagramme de définition de blocs.

4.1.1 Représentation textuelle

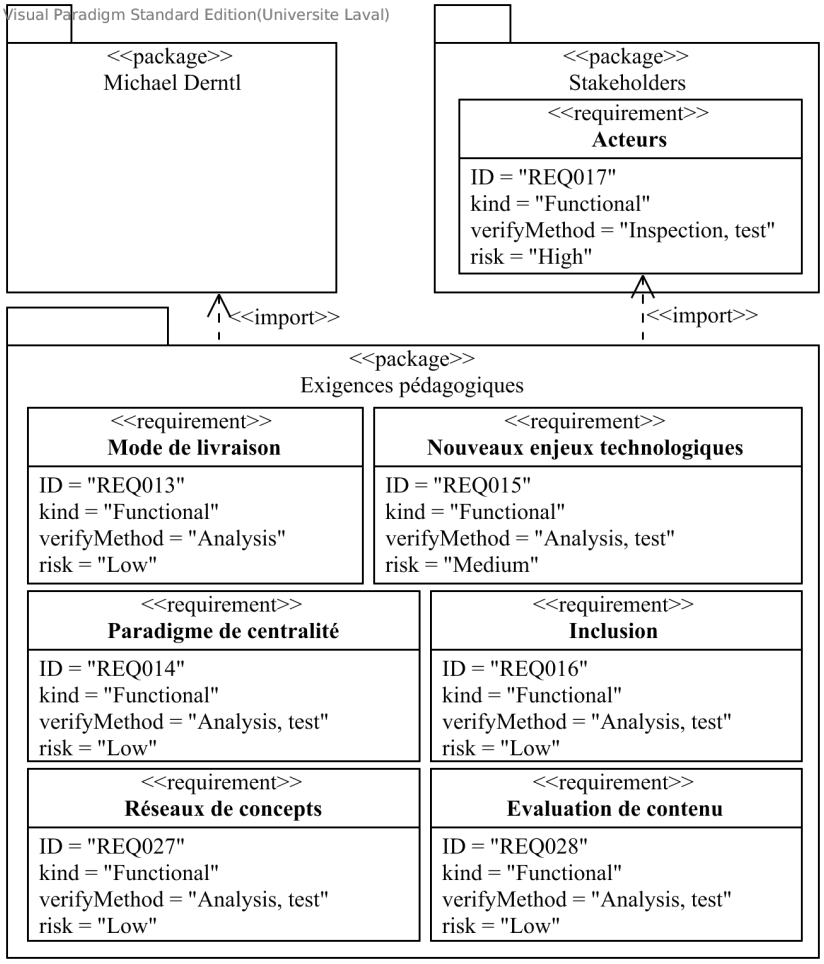
- `Identifiant:` PKG002.
- `Nom:` Exigences-pédagogiques-logiciel-d'enseignement.
- `Définition:` cet ensemble de patrons constituent l'ensemble des exigences pédagogiques qui expriment les besoins pédagogiques auxquels doivent répondre le logiciel en cours de conception et ses fonctionnalités et propriétés.
- `Mots-clés:` exigences pédagogiques, mode de livraison.
- `Motivation:` l'équipe chargée du développement du LEA peut avoir des difficultés à dresser la liste des fondements pédagogiques que doit respecter le logiciel. Ce paquetage décrit cette liste et présente les relations de dépendance entre les différents éléments pédagogiques. L'équipe chargée du développement du logiciel gagne ainsi du temps et bénéficie de l'expérience de ses pairs.
- `Exemple:` cette liste de patrons pédagogiques peut être utilisée pour le développement d'un logiciel de discussion instantanée, un forum de discussion, un wiki, conçu pour des activités péagogiques.
- `Applicabilité:` ce paquetage de patrons est à utiliser pour déterminer les exigences pédagogiques auxquelles doit répondre le LEA.

■ `Conséquences:`

— L'équipe de développement gagne un temps précieux en évitant de longues recherches sur les patrons pédagogiques à considérer.

— L'équipe de développement évite des erreurs de conception qui peuvent survenir suite à l'oubli d'une exigence pédagogique fondamentale.

4.1.2 Diagramme des exigences



Ce diagramme des exigences pédagogiques représente les conditions pédagogiques devant toujours être satisfaites par le LEA envisagé. Il peut nécessiter le recours à d'autres paquetages. Chaque exigence représentée sur le diagramme des exigences est un patron.

4.1.2.1 Patrons recensés

P10 *Mode de livraison :* il s'agit du mode d'enseignement qui peut être individualisé, ou collaboratif ou hybride.

P11 *Paradigme de centralité :* il s'agit de définir si l'environnement pédagogique sera centré sur l'étudiant ou sur l'enseignant ou symétrique.

P12 *Réseaux de concepts :* un réseau de concepts permet de partir d'anciennes connaissances pour en produire de nouvelles.

P13 *Evaluation de contenu :* il s'agit d'évaluer le contenu produit par les étudiants à travers le principe de l'*analyse sémantique latente*.

P14 *Nouveaux enjeux technologiques :* la prise en compte des technologies les plus récentes dans les patrons que nous proposons, leurs potentiels, ouvre de nouvelles perspectives pour l'enseignement et l'apprentissage auxquelles les exigences pédagogiques doivent s'adapter.

P15 *Inclusion :* l'interopéralité des technologies les plus évoluées prises en compte dans les patrons que nous proposons ouvre aussi de nouvelles perspectives pour l'enseignement et l'apprentissage auxquelles les exigences pédagogiques doivent s'adapter.

4.1.2.2 Exemple : le patron *Nouveaux enjeux technologiques*

Le patron *Nouveaux enjeux technologiques* permet de définir les règles, d'un point de vue pédagogique, de prise en compte des technologies les plus récentes dans le processus de conception du LEA. Le potentiel de ces nouvelles technologies offre de nouvelles perspectives pour l'enseignement et l'apprentissage auxquelles les exigences pédagogiques doivent s'adapter.

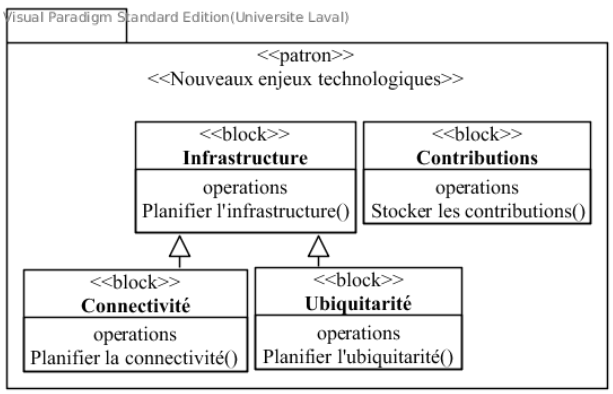
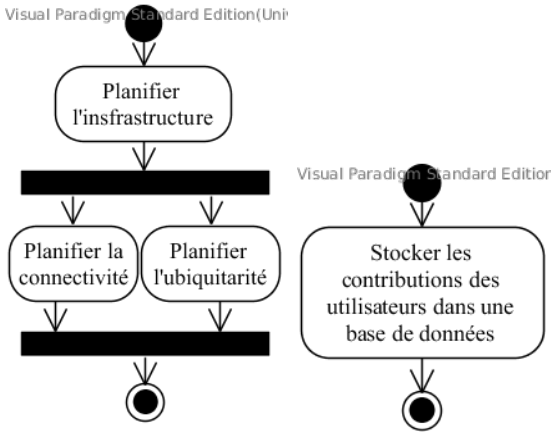


Diagramme de définition de blocs



Diagrammes des activités

Il s'agit d'abord de prévoir la mise en place de l'infrastructure c'est-à-dire de la connectique, de la bande passante, de la sécurité, etc., nécessaire au fonctionnement optimal de la nouvelle technologie à intégrer au LEA et de prévoir aussi un cadre d'évolutivité. Il faut ensuite prévoir la mise en place de l'infrastructure logique et physique nécessaire pour permettre l'accessibilité à la technologie considérée en tout lieu ou en plusieurs lieux simultanément et de prévoir aussi la mise en place d'une stratégie efficace d'assistance aux utilisateurs. Enfin, il s'agit aussi de

prévoir l'archivage et le stockage, dans des journaux ou des bases de donneés, des messages postés par les utilisateurs, de prévoir l'élaboration de règles d'encadrement de l'utilisation de la technologie et de prévoir aussi l'évaluation de la pertinence et de l'efficacité de la technologie, c'est-à-dire le degré avec lequel le logiciel correspond aux exigences du logiciel.

4.1.2.3 Exemple : le patron *Réseaux de concepts*

Un réseau de concept, ou carte conceptuelle, permet de créer de nouvelles connaissances à partir des messages qu'ont posté les étudiants sur le LEA. Il permet aussi d'une part d'organiser et structurer les connaissances enfouies dans ces messages et d'autre part de les représenter graphiquement. Le patron *Réseaux de concepts* permet donc de décrire le processus d'élaboration de nouvelles connaissances à partir d'anciennes connaissances.

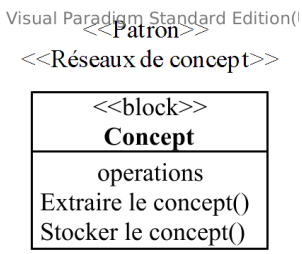


Diagramme de définition de blocs

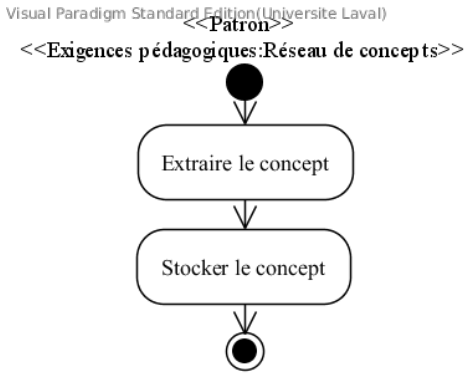


Diagramme des activités

On extrait un concept des messages postés par les étudiants sur le LEA, par exemple dans le cadre d'un forum de discussion pédagogique. Le réseau logique de concepts se construit ainsi autour d'un concept principal. Les concepts extraits sont ensuite stockés dans une base de donnnés. Le patron *Réseaux de concept* dépend du patron *Exigences pédagogiques*.

4.1.3 Diagramme de définition de blocs

Pour constituer le diagramme de définition de blocs du paquetage *Pédagogique*, nous avons choisi de réutiliser les structures définies à cet effet par Michael Derntl (Derntl, 2005).

Michael Derntl a défini 50 patrons pédagogiques auxquels nous ferons référence à travers la création et l'utilisation d'un paquetage nommé *Michael Derntl*. Le lecteur pourra se référer au contenu de notre paquetage *Michael Derntl* en prenant connaissance de la thèse de doctorat de Derntl (Derntl, 2005) [1].

1. voir pages 169-401

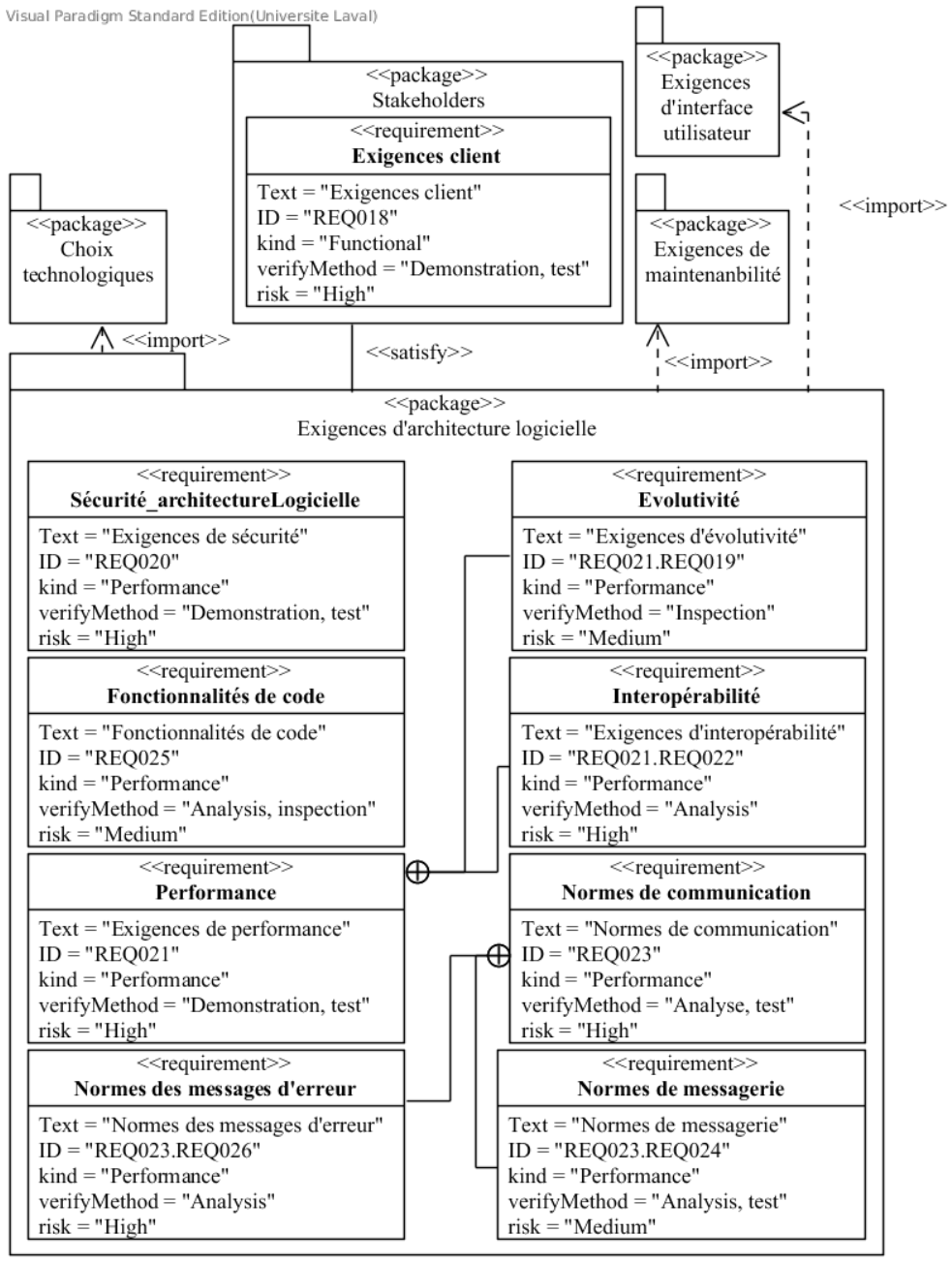
4.2 Le paquetage *Architecture logicielle*

Il comprend l'ensemble des patrons du paquetage *Architecture logicielle*. Dans le cadre de cette thèse, nous définissons l'architecture logicielle comme étant la description schématique de haut niveau de la structure d'un logiciel, de ses différents composants et de la manière dont ces composants interagissent entre eux.

4.2.1 Représentation textuelle

- `Identifiant:` PKG004.
- `Nom:` Exigences-architecture-logicielle.
- `Définition:` cet ensemble de patrons est l'ensemble des patrons à considérer pour la conception de l'architecture d'un LEA moderne et compatible avec le Web 4.0. Ces patrons s'appliquent aux logiciels d'enseignements asynchrones et synchrones. Ils décrivent les éléments clés d'une architecture logicielle, leurs interrelations et interactions et présentent sous une forme structurée et intuitive l'ensemble des contradictions et compromis auxquels doit faire face l'équipe des développeurs en vue de satisfaire les exigences du client. Enfin ces patrons permettent de décrire les détails d'implémentation du code.
- `Mots-clés:` exigences d'architecture, cadre de conception.
- `Motivation:` le passage de la phase d'analyse logicielle extrêmement abstraite où l'on collecte les exigences du client, à la phase de conception de l'architecture du logiciel peut s'avérer être laborieux voire très peu intuitif pour l'équipe de développement. Ces patrons d'architecture logicielle permettent donc de décrire comment ce passage doit être effectué et comment l'architecture doit être réalisée afin de répondre non seulement aux exigences du client mais aussi aux normes et standards à respecter. En somme ces patrons permettent de décrire les bonnes manières en matière de conception de l'architecture.
- `Exemple:` ces patrons peuvent permettre de définir le style architectural du logiciel (en couches, centrée sur les données, orientée objets, etc.).
- `Applicabilité:` utiliser ce paquetage de patrons pour la conception de l'architecture d'un système.
- `Conséquences:` aucune conséquence identifiée.

4.2.2 Diagramme des exigences



Ce diagramme des exigences de l'architecture logicielle importe des exigences des paquetages *Choix technologiques*, *Exigences de maintenabilité* ou d'évolution et *Acteurs* dont il importe partiellement ou non des exigences.

4.2.2.1 Patrons recensés

P16 *Sécurité_ architectureLogicielle :* contraintes de sécurité à remplir par l'architecture logicielle.

P17 *Fonctionnalités de code :* en vue d'en faciliter l'implémentation future, il s'agit de décomposer l'architecture du LEA en fonctionnalités ou modules de code selon des contraintes bien définies.

P18 *Performance :* il s'agit de définir le niveau optimal de certains paramètres de performance tels que la vitesse, le degré de disponibilité, le délai de réponse et le délai de récupération des diverses fonctions logicielles.

P19 *Norme des messages d'erreur :* il s'agit de définir la codification des messages d'erreur.

P20 *Norme de messagerie :* définir les normes régissant les échanges de messagerie entre utilisateurs du LEA.

P21 *Evolutivité :* définir les normes et les contraintes de maintenance de l'architecture logicielle.

P22 *Normes de communication :* définir le choix des protocoles de communication, et les contraintes d'utilisation de ces protocoles.

P23 *Interopérabilité :* définir les contraintes et les normes d'interopérabilité entre les différentes applications et technologies constituant le LEA.

4.2.2.2 Exemple : le patron *Interopérabilité*

L'interoperabilité est la capacité que possède un logiciel ou une technologie à fonctionner avec d'autres logiciels existants ou futurs, sans restriction d'accès ou de mise en œuvre grâce à un ensemble structuré de normes, standards, spécifications et politiques.

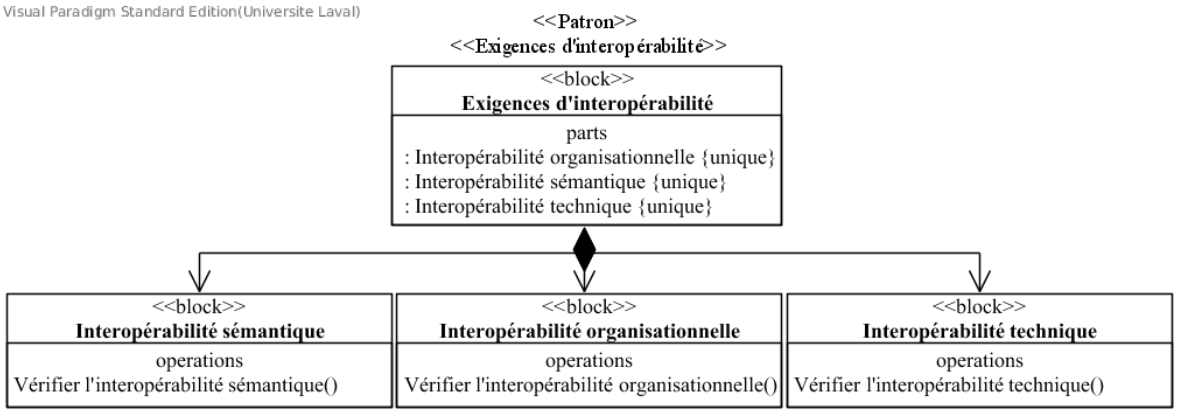


Diagramme de définition de blocs du patron *Interopérabilité*.

Il s'agit d'abord de vérifier l'interopérabilité organisationnelle, c'est-à-dire de décrire les étapes de modélisation des processus métiers dans le but de prendre en compte la collaboration entre

logiciels et technologies qui n'ont pas les mêmes structures organisationnelles et qui ne gèrent pas des processus similaires. Il s'agit ensuite de vérifier l'interopérabilité sémantique, c'est-à-dire de garantir que le sens exact des informations échangées entre logiciels ou technologies peut être compris par toute application qui n'a pas été conçue initialement dans ce but. Cet aspect facilite l'agrégation et la réutilisation d'informations hétérogènes et se base notamment sur la réutilisabilité, la standardisation des données et l'utilisation de métadonnées qui renseignent sur le contexte lié à la création des données. Il s'agit enfin de vérifier l'interopérabilité technique, c'est-à-dire de prendre en compte les aspects liés à la connexion technique des systèmes et des services informatiques. Cet aspect concerne des domaines aussi variés que la définition d'interfaces et de standards ouverts, l'intégration des données, la présentation et l'échange d'informations, l'accessibilité ou les services de sécurité, etc.

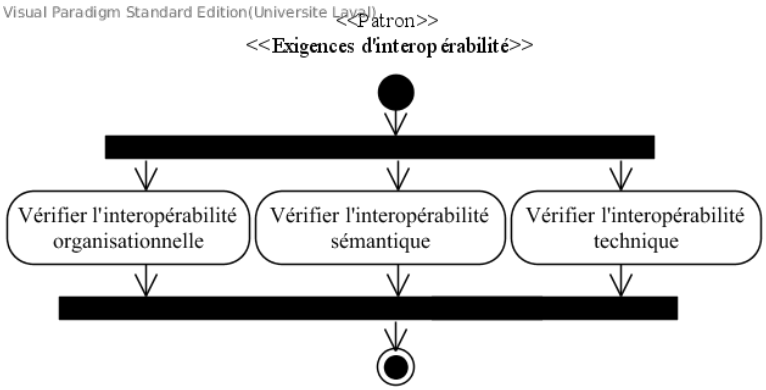


Diagramme d'activités du patron *Interopérabilité*.

4.2.2.3 Exemple : le patron *Performance*

Le patron *Performance* permet de décrire les règles permettant de définir les paramètres de performance tels que la vitesse, le degré de disponibilité, le délai de réponse et le délai de récupération des diverses fonctions logicielles.

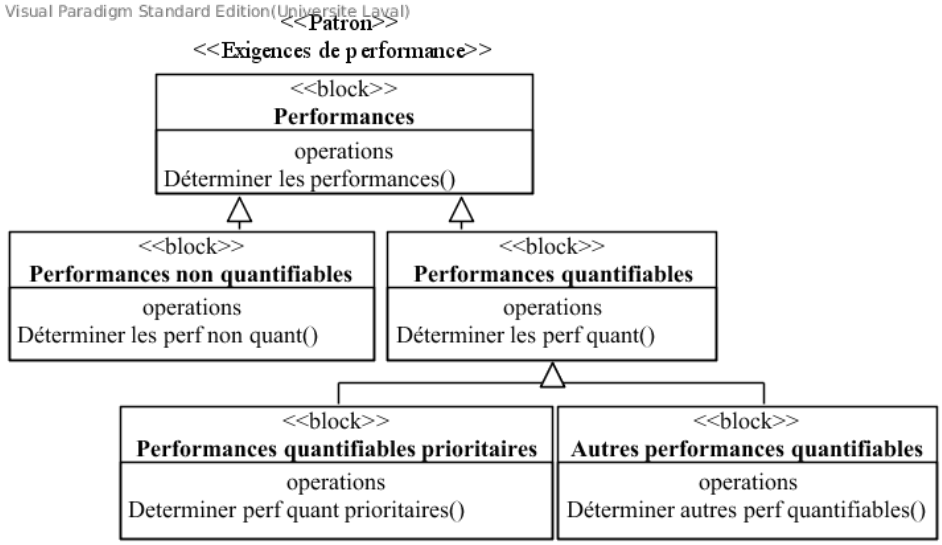


Diagramme de définition de blocs du patron *Performance*.

Le diagramme d'activités du patron *Performance* indique les activités associées au diagramme de définition de blocs précédemment présenté. Comme activités, il s'agit d'abord de la catégorisation des exigences selon les exigences quantifiables (temps de recherche pour les requêtes, temps nécessaire pour l'affichage d'une page, nombre maximal de fonctionnalités gérées par le logiciel, nombre maximal de connexions simultanées, taille maximale des fichiers sources, etc,) et celles qui ne le sont pas. Il faut ensuite déterminer les exigences les plus importantes.

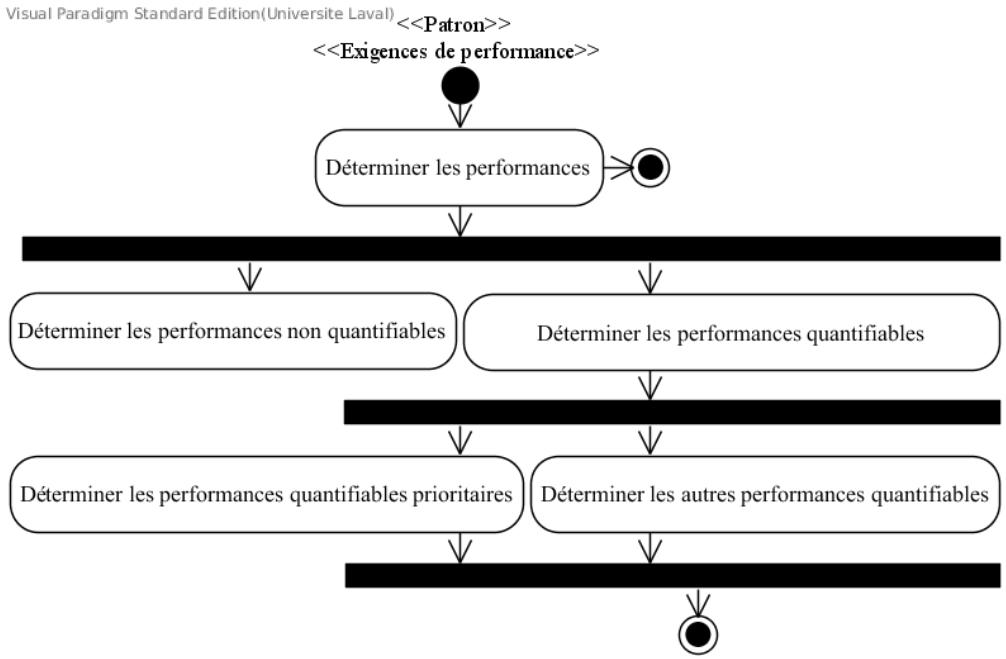
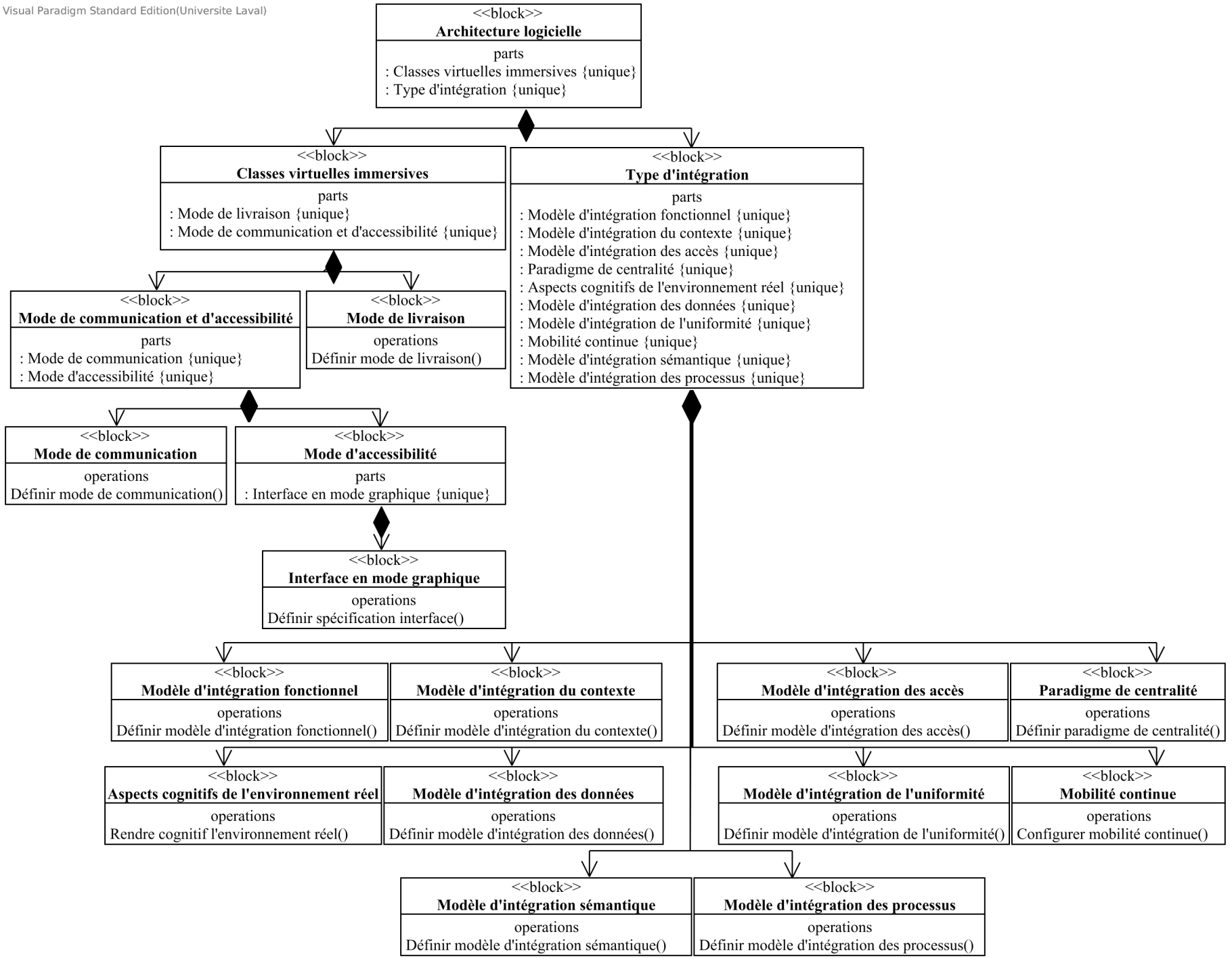


Diagramme d'activités du patron *Performance*.

4.2.3 Diagramme de définition de blocs (page suivante)



4.2.3.1 Patrons recensés

P24 *Classes virtuelles immersives :* pour renforcer le sentiment de présence en classe en permettant aux étudiants et enseignants d'interagir et de manipuler en temps réel des objets en 3 D.

P25 *Mode de livraison :* modélisation du mode d'enseignement qui peut être individualisé, ou collaboratif ou hybride.

P26 *Mode de communication :* modélisation du mode de communication qui peut être synchrone ou asynchrone ou hybride.

P27 *Mode d'accessibilité :* il s'agit de définir et de modéliser le mode d'accessibilité, c'est-à-dire si le contenu de la plateforme d'enseignement et d'apprentissage sera accessible en mode *hors connexion* ou non. Notons ici la nuance entre les modes *asynchrone avec une connectivité au réseau Internet* et *hors connexion*, c'est-à-dire *asynchrone sans connectivité au réseau Internet.*

P28 *Interface :* il s'agit de modéliser le type d'interface de livraison du contenu qui peut être celui d'un système embarqué, ou celui d'un système mobile, ou celui d'une surface tactile, ou celui d'un système natifs ou celui d'une solution hybride.

P29 *Mobilité continue :* possibilité pour l'étudiant et l'enseignant de passer sans discontinuité d'un type d'interface à un autre.

P30 *Paradigme de centralité :* implémentation du type d'environnement pédagogique, soit centré sur l'étudiant, soit sur l'enseignant ou symétrique.

P31 *Aspects cognitifs de l'environnement réel :* doter d'intelligence les environnements du monde réel.

P32 *Type d'intégration :* les différentes composantes de l'application doivent être intégrées en vue d'un traitement optimal de l'information. L'intégration consiste en l'union de parties, ou de données entières, afin d'améliorer la qualité de l'information.

P33 *Modèle d'intégration des données :* la plateforme d'enseignement à distance fait partie intégrante d'un environnement numérique plus ou moins étendu. Les données reccueillies dans cet environnement numérique devraient être accessibles sur la plateforme et vice-versa. La prise en compte de l'intégration des données diminue le coût de collecte de celles-ci et améliore leur qualité, leur cohérence et leur consistance, et évite de ce fait leur redondance, ce qui engendre aussi un coût non négligeable. Par exemple, les paramètres d'authentification des étudiants aux autres services électroniques qui requièrent une authentification peuvent être réutilisés pour permettre aux étudiants de s'authentifier avec les mêmes paramètres sur la plateforme logicielle d'enseignement.

P34 *Modèle d'intégration sémantique :* ce modèle s'avère pertinent dans un environnement où les composants de plusieurs systèmes distincts utilisent les mêmes concepts et de ce fait sont supposées interprèter les données de façon identiques. Par exemple si une

application émettrice d'une information à l'intention d'une application tierce, utilise le code "Qwerty" et que l'application réceptrice utilise le code «Azerty» alors le principe d'intégration sémantique n'est pas respecté. Le principe d'intégration sémantique s'exprime à travers l'implémentation d'un algorithme de traduction entre les deux systèmes communiquants.

P35 *Modèle d'intégration de l'uniformité :* plusieurs composantes d'application munies d'interfaces peuvent afficher une même information, par exemple la liste des salles d'informatique libres actuellement, ou la liste et la date des examens prévus au cours de la session.

P36 *Modèle d'intégration des accès :* les composants nécessaires à l'exécution d'une certaine tâche peuvent être utilisés lorsque nécessaire.

P37 *Modèle d'intégration des processus :* dans un environnement numérique diversifié, comme celui d'une université, plusieurs fonctionnalités similaires peuvent être implémentées au niveau de plusieurs applications distinctes. Le modèle d'intégration fonctionnel permet alors de s'assurer que l'on évite une redondance des mêmes fonctionnalités. Par exemple, les modules d'authentification, de gestion de la liste des cours, peuvent être implémentées au niveau de plusieurs applications disctinctes. Le modèle d'intégration fonctionnel permet de faire en sorte que le même module d'authentification soit implémenté une seule fois et appelé au niveau des applications où cela est nécessaire, de même que pour le module de gestion de la liste des cours.

P38 *Modèle d'intégration fonctionnel :* les attributs devant être utilisés dans plusieurs composants sont implémentés une seule fois.

P39 *Modèle d'intégration du contexte :* les différents niveaux d'intégration de données, sémantique, d'accès peuvent bien se faire. Cependant il est indispensable de conserver le contexte des données et des applications agrégées sous peine de devoir manipuler des informations dénuées de sens.

4.2.3.2 Exemple : le patron *Architecture logicielle : :Interface*

Le patron *Architecture logicielle : :Interface* permet de décrire comment les interfaces utilisateur du LEA s'intègrent à l'architecture logicielle globale du LEA.

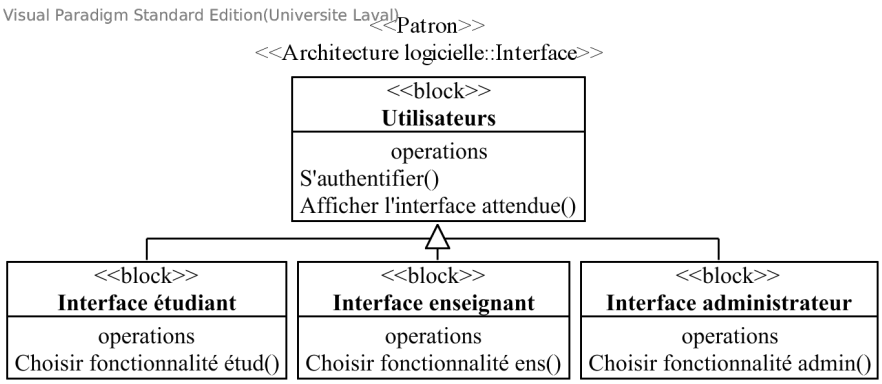


Diagramme de définition de blocs du patron *Architecture logicielle : :Interface*.

L'utilisateur s'authentifie. En fonction des paramètres d'authentification entrés, et selon les privilèges d'authentification qui lui sont octroyés, il accède à des interfaces, pour étudiant, enseignant ou administrateur, d'où il peut choisir les fonctionnalités du LEA qui l'intéressent.

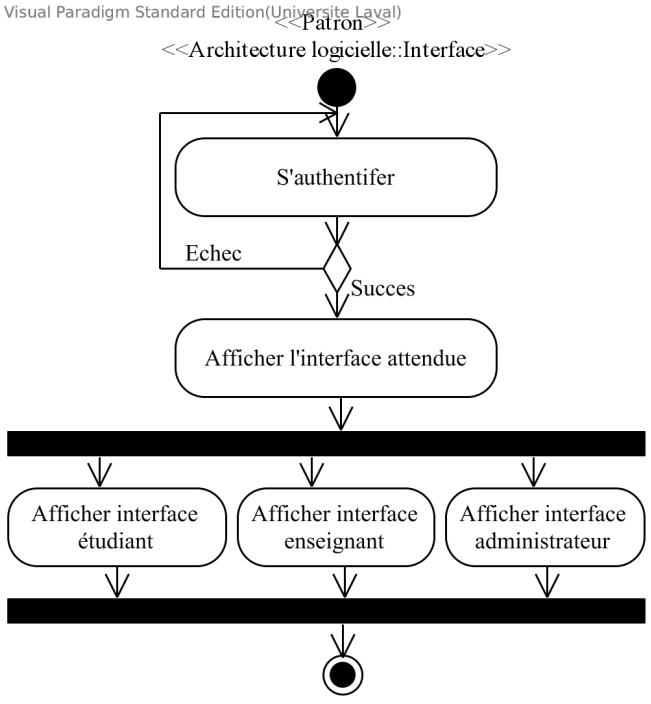


Diagramme d'activités du patron *Architecture logicielle : :Interface*.

4.2.3.3 Exemple : le patron *Modèle d'intégration des données*

Le patron *Modèle d'intégration des données* permet de définir les règles permettant de recueillir des données provenant de plusieurs applications distinctes et de les intégrer dans une même application de destination, tout en veillant à la cohérence et à la consistance de ces données.

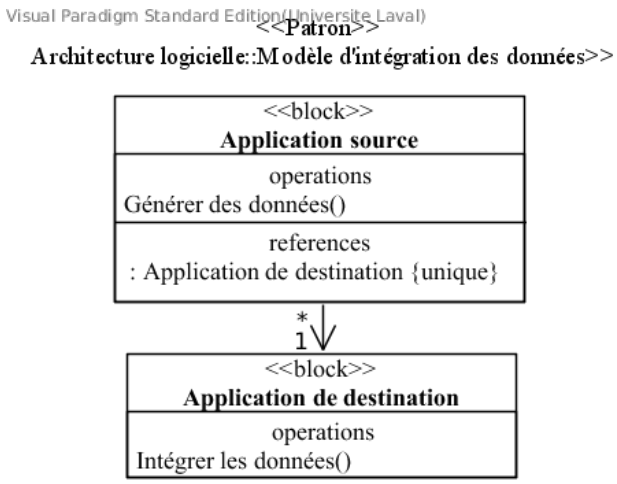


Diagramme de définition de blocs

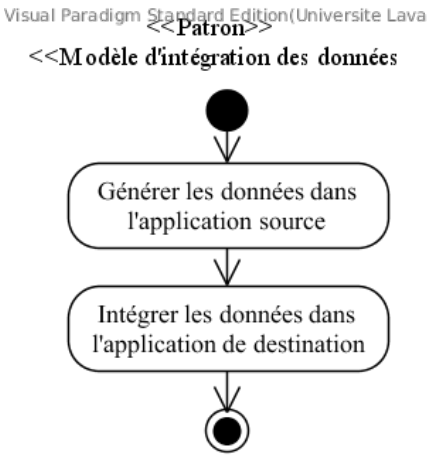


Diagramme des activités

Les données qui doivent être intégrées sont d'abord rassemblées. Celles-ci proviennent de différents logiciels et doivent être prises en compte par le LEA. En effet, le LEA fait partie intégrante d'un système d'environnement numérique plus ou moins étendu. Les données reccueillies dans cet environnement numérique devraient être accessibles sur le LEA et vice-versa. La prise en compte de l'intégration des données diminue le coût de collecte de celles-ci et améliore leur qualité, leur cohérence et leur consistance, et évite de ce fait leur redondance, ce qui engendre aussi un coût non négligeable. Par exemple, les paramètres d'authentification des étudiants aux autres services électroniques qui requièrent une authentification peuvent être réutilisés pour permettre aux étudiants de s'authentifier avec les mêmes paramètres sur la plateforme logicielle d'enseignement. Après avoir collecté ces données, il faut les unifier au sein de l'application qui doit les recevoir.

4.3 Le paquetage *Interface utilisateur*

Il permet de fixer les modalités d'assemblage des composants théoriques de l'interface utilisateur et de définir de façon exhaustive leurs spécifications. Il comprend l'ensemble des patrons du paquetage *Interface utilisateur*.

4.3.1 Représentation textuelle

- `Identifiant:` PKG003.
- `Nom:` Exigences-interface-utilisateur.
- `Définition:` cet ensemble de patrons est l'ensemble des patrons à considérer pour la conception de l'interface utilisateur, graphique ou non, d'un LEA moderne et compatible avec le Web 4.0. Ces patrons s'appliquent aux logiciels d'enseignements asynchrones et synchrones.
- `Mots-clés:` exigences d'interface, affichage.
- `Motivation:` la conception de l'interface utlisateur peut s'avérer être cauchemardesque pour l'équipe de développement, surtout dans le cadre du développement d'un LEA. En effet, entre les couleurs à considérer ou non, les options d'accessibilité pour personnes malvoyantes ou malentendantes, ou le balisage et le guidage des utilisateurs, ou encore la compatibilité ascendante à prévoir avec les terminaux informatiques, il y a des contractions auxquelles l'équipe de développement doit faire face et des compromis à faire afin que les fonctionnalités de l'application soient compréhensibles des utilisateurs, que ceux-ci en aient une très bonne perception et que le LEA soit simple à utiliser. L'interface utilisateur est donc essentielle puisque c'est par elle que l'utilisateur entre en contact avec le logiciel, l'accepte ou le rejette. En conséquence, ces patrons permettent de décrire comment doit être réalisée l'interface et quelles doivent en être les fonctionnalités, facilitant ainsi la tâche de l'équipe de développement.
- `Exemple:` ces patrons peuvent servir à définir les caractéristiques techniques d'une interface utilisateur et les composants qu'elle doit contenir.
- `Applicabilité:` utiliser ce paquetage de patrons pour la conception de l'interface utilisateur d'un système.
- `Conséquences:` aucune conséquencee identifiée.

4.3.2 Diagramme des exigences

L'ensemble des exigences de l'interface utilisateur dépend des paquetages *Architecture logicielle*, *Exigences pédagogiques* et *Acteurs* dont il satisfait partiellement ou non des exigences.

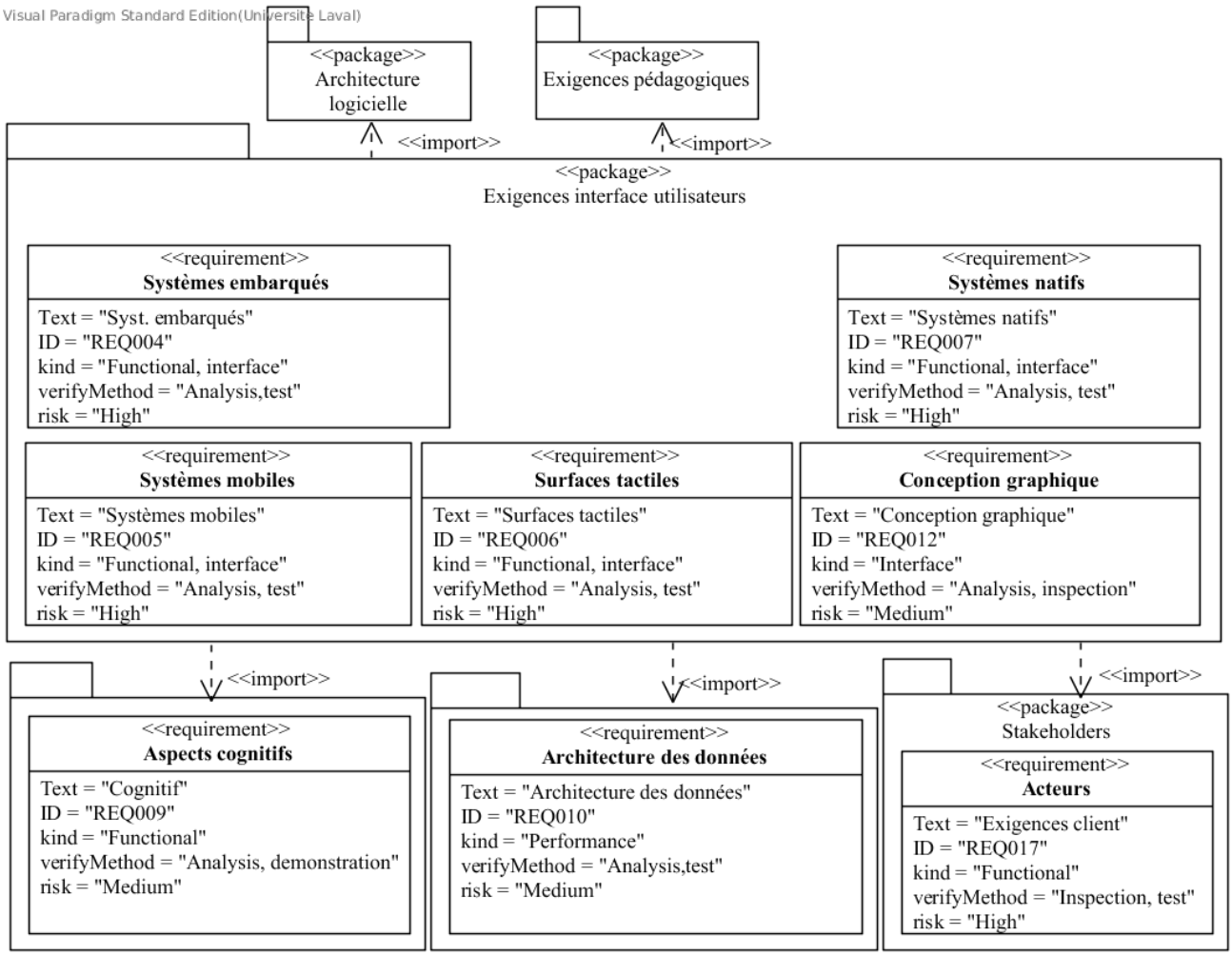


Diagramme des exigences du paquetage *Interface utilisateur*.

4.3.2.1 Patrons recensés

P40 *Systèmes embarqués :* système électronique et informatique autonome, n'ayant pas une vocation purement informatique.

P41 *Systèmes mobiles :* appareil mobile, par exemple un téléphone intelligent.

P42 *Surfaces tactiles :* surface interactive et dynamique dotée d'une interface tactile et multitouches, par exemple, table interactive, mur interactif.

P43 *Conception graphique :* création de la charte graphique de l'interface.

P44 *Systèmes natifs :* ordinateur traditionnel tel un poste de travail ou un ordinateur portable.

P45 *Aspects cognitifs :* contraintes d'interface pour un système à doter d'intelligence.

P46 *Architecture des données :* découpage et organisation des données sur l'interface utilisateur.

4.3.2.2 Exemple : le patron *Systèmes natifs*

Le système natif désigne un ordinateur traditionnel tel un poste de travail ou un ordinateur portable. Le patron *Systèmes natifs* permet de définir les règles de conception de l'interface utilisateur d'un LEA pour ce type de matériel.

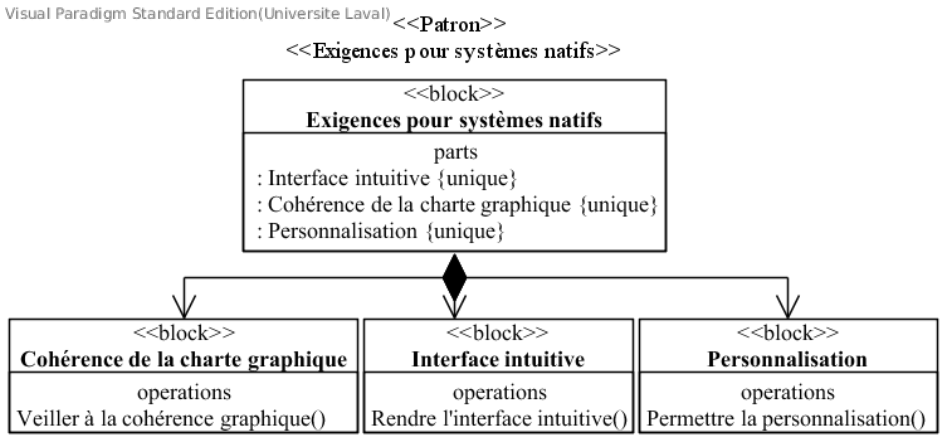


Diagramme de définition de blocs du patron *Systèmes natifs*.

La première étape consiste à rendre l'interface intuitive, c'est-à-dire la rendre très utilisable et conviviale pour une prise en main immédiate. La composition et l'enchaînement des écrans doivent être évidents et la terminologie utilisée dans les interfaces doit être simple et dénuée de toute ambiguïté. Il s'agit ensuite de veiller à la cohérence de la charte graphique, c'est-à-dire que, dans un même groupe de pages ou dans un même module, de s'assurer de la cohérence des couleurs, des polices de caractères, de l'emplacement des éléments et leurs fonctionnalités, etc. Enfin, il s'agit de permettre la personnalisation, c'est-à-dire de permettre aux utilisateurs de personnaliser leurs interfaces selon leurs préférences.

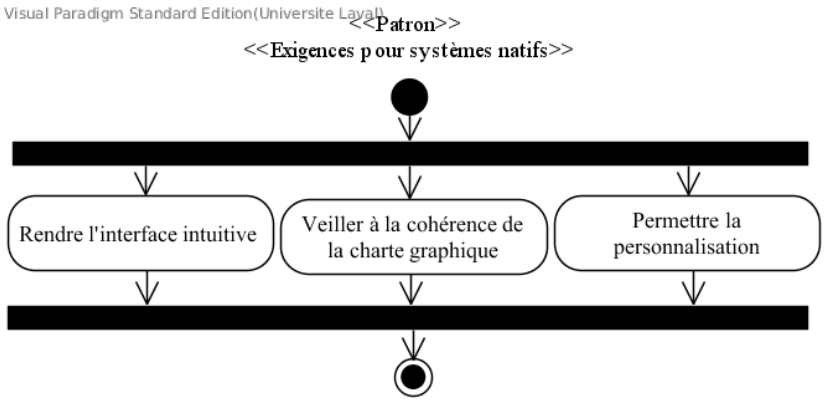
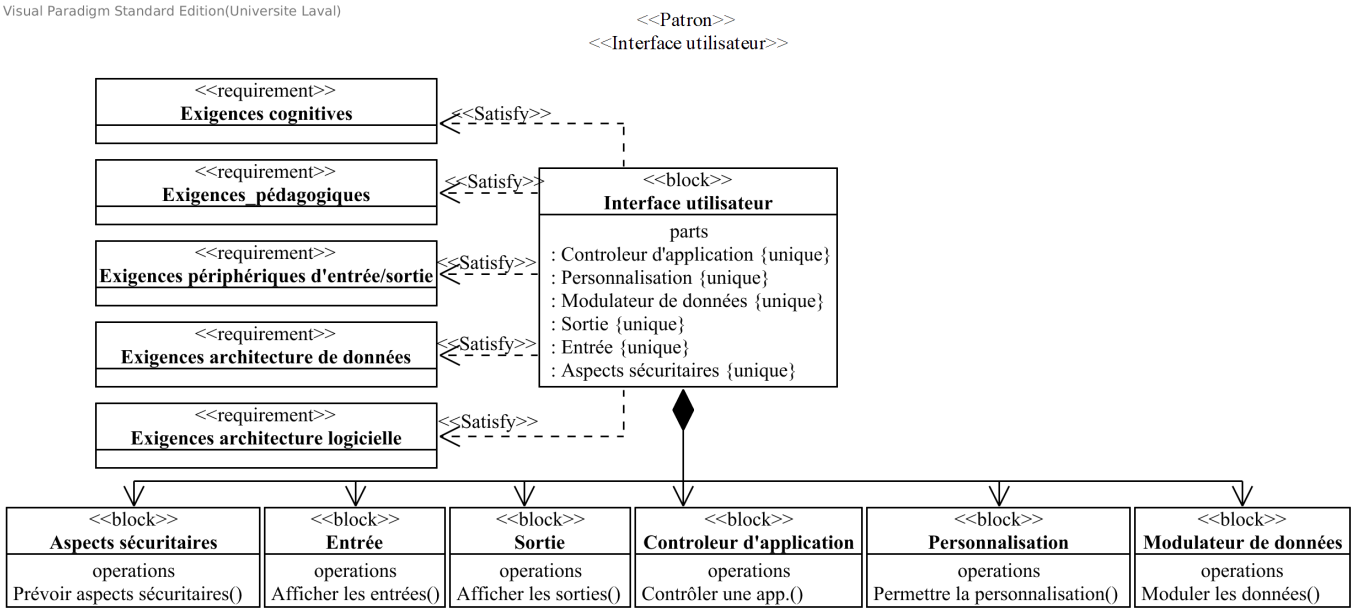


Diagramme d'activités du patron *Systèmes natifs*.

4.3.3 Diagramme de définition de blocs



4.3.3.1 Patrons recensés

P47 *Entrée* pour la capture de données.

P48 *Sortie* pour l'affichage des données traitées.

P49 *Sécurité :* aspects sécuritaires de l'interface utilisateur.

P50 *Contrôleur d'application :* module de l'interface chargé de démarrer une application automatiquement à partir de l'interface, par exemple application de géolocalisation.

P51 *Personnalisation :* possibilité pour l'utilisateur de personnaliser son interface selon ses préférences.

P52 *Modulateur de données :* module de l'interface chargé de réaliser l'aggrégation, c'est-à-dire la fusion de données, en cas de besoin.

4.3.3.2 Exemple : le patron *Personnalisation*

La personnalisation est la possibilité pour l'utilisateur de personnaliser son interface selon ses préférences. Le diagramme de définition de blocs du patron *Personnalisation* représente les principales structures qui par leurs interactions permettent à l'utilisateur de pouvoir personaliser ses interfaces graphiques. Le terme utilisateur désigne l'étudiant ou l'enseignant.

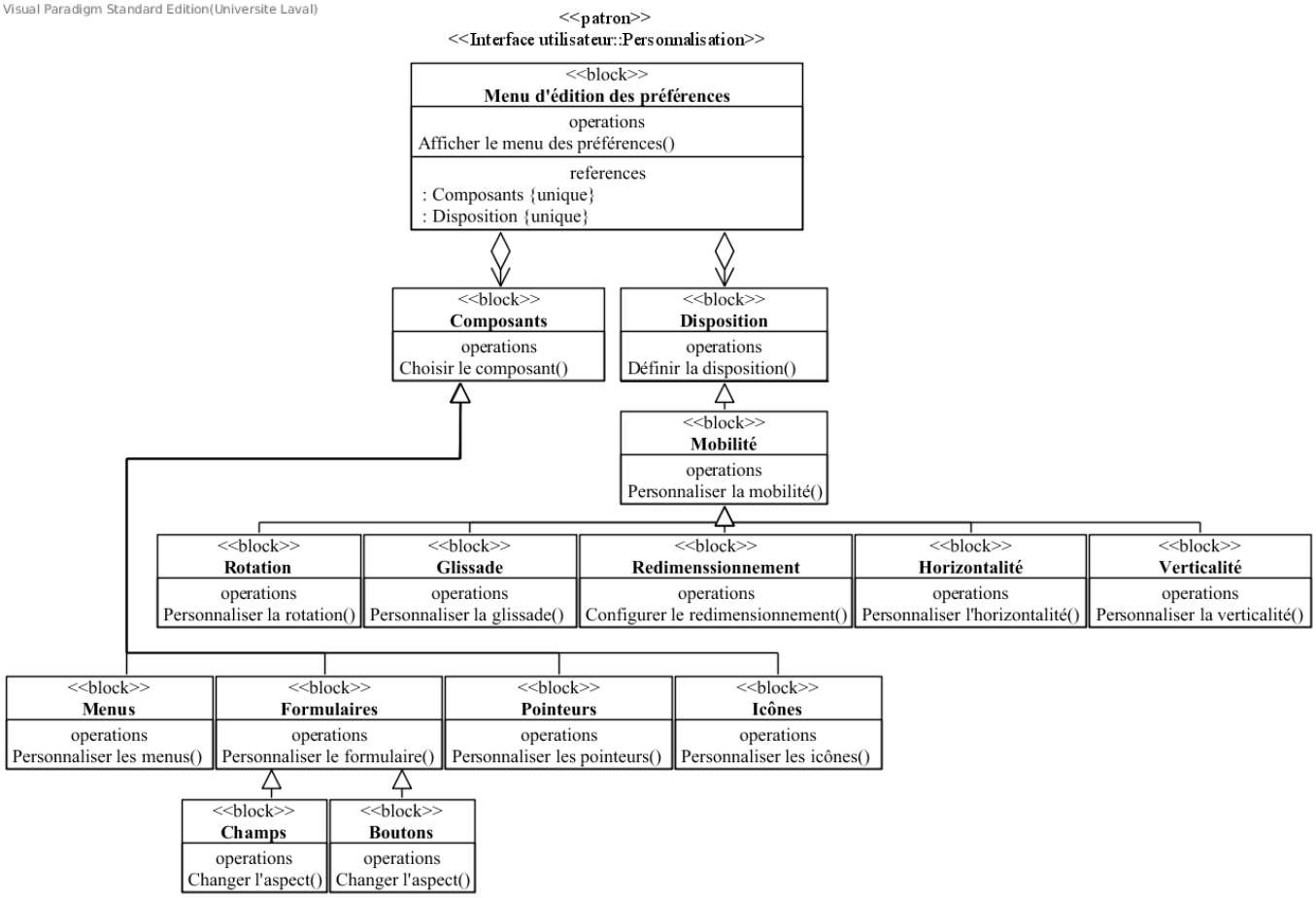


Diagramme de définition de blocs du patron *Interface utilisateur : :Personnalisation.*

Le diagramme d'activités ci-dessous a été allégé d'un certain nombre d'activités correspondants à des blocs du diagramme de définition des blocs précédent, pour le rendre plus lisible.

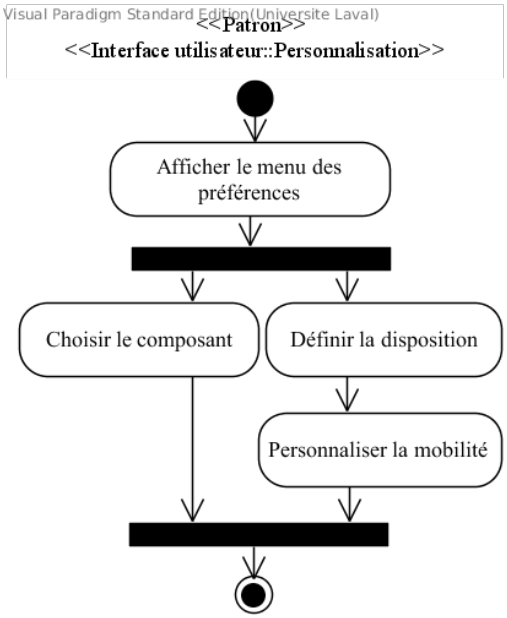


Diagramme d'activités du patron *Interface utilisateur : :Personnalisation.*

Comme activités à réaliser dans le cadre de la personnalisation, il s'agit d'abord d'afficher le menu des préférences du LEA et de choisir de modifier les préférences d'un composant, tel un bouton d'un formulaire, des éléments d'un menu, ou une icône, ou de choisir de modifier la disposition d'un objet, c'est-à-dire sa verticalité ou son horizontalité, ou alors de le faire roter selon un angle défini.

4.3.3.3 Exemple : le patron *Contrôleur d'application*

Le *Contrôleur d'application* est un module de l'interface du LEA chargé de démarrer automatiquement une tierce application à partir de l'interface. L'exécution automatique de cette tierce application peut se déclencher par exemple suite à l'entrée d'une information précise dans un champ pré-déterminé de l'interface.

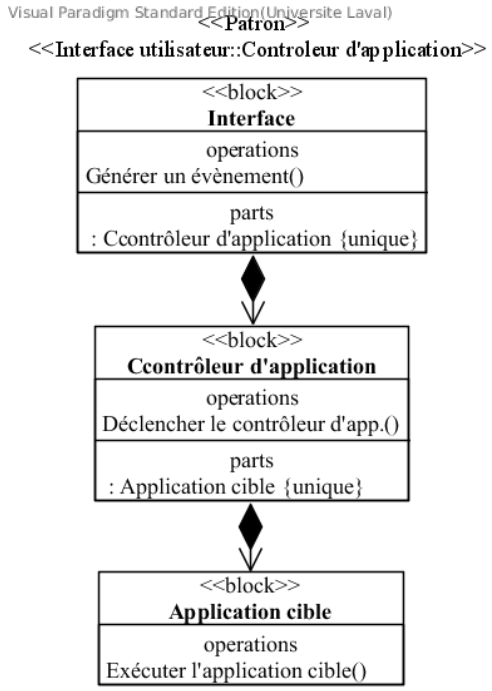


Diagramme de définition de blocs

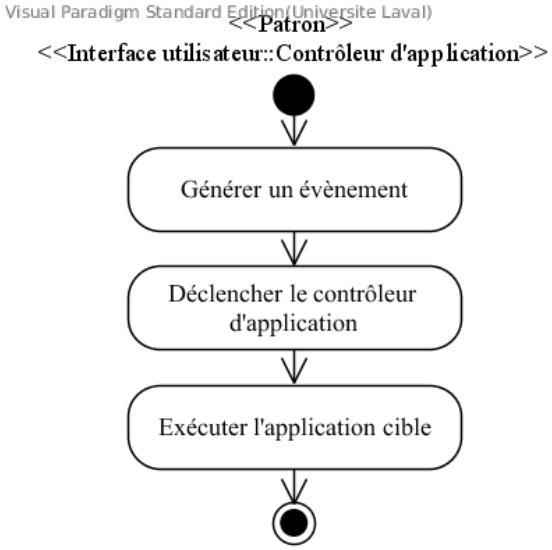


Diagramme des activités

L'interface génère un évènement suite à une action spécifique de l'utilisateur, comme la saisie d'une donnée. Par exemple, l'utilisateur étudiant peut avoir sélectionné le nom d'un cours qu'il doit suivre. Le contrôleur d'application pourrait se charger ensuite de rechercher l'application cible associée à l'évènement généré. Si cette application cible est trouvée, le contrôleur d'application l'exécute. Cette application tierce peut être par exemple une fonction de géolocalisation qui, à l'aide d'une carte géographique, indiquera à l'utilisateur étudiant le chemin à suivre, et la distance à parcourir, depuis l'endroit où il se trouve sur le campus jusqu'au local où doit avoir lieu son cours.

4.4 Le paquetage *Choix technologiques*

Il permet de fixer les spécifications du matériel et des logiciels nécessaires d'une part à la conception du LEA envisagé, et d'autre part à son futur déploiement ainsi que d'identifier les performances du système, des différents composants informatiques, leurs interrelations et leurs interactions.

4.4.1 Représentation textuelle

- `Identifiant:` PKG005.
- `Nom:` Choix-technologiques.
- `Définition:` ces patrons décrivent comment doivent être effectués les choix technologiques indispensables au développement du LEA mais aussi de spécifier les caractéristiques des choix technologiques à effectuer et devant constituer l'environnement d'exécution du logiciel après son développement. Les choix technologiques concernent les aspects logiciel aussi bien que matériel.
- `Mots-clés:` choix du matériel, choix des logiciels.
- `Motivation:` D'une part l'équipe de développement du LEA peut être inexpérimentée ou faire de mauvais choix concernant les logiciels de développement à utiliser, eu égard au contexte multidisciplinaire des logiciels d'enseignement. D'autre part, une fois le développement du logiciel terminé, le client peut ne pas savoir quelles sont les exigences matérielles et logicielles minimales, ou optimales à mettre en place pour le fonctionnement optimal du LEA. Ces patrons permettent de décrire ce qu'il faut faire dans chacune de ces situations.
- `Exemple:` ces patrons peuvent servir à définir les spécifications du matériel et des logiciels qui doivent intervenir dans la conception du LEA, et aussi à définir les spécifications du matériel et logiciels qui doivent héberger le LEA.
- `Applicabilité:` utiliser ce paquetage de patrons pour effectuer des choix technologiques stratégiques en vue de la conception puis la mise en oeuvre d'un système.
- `Conséquences:` aucune conséquence identifiée.

4.4.2 Diagramme des exigences

Dans cette phase de définition des choix technologiques, les exigences formulées par les architectes et les concepteurs logiciels sont spécifiées et structurées à l'aide du diagramme des exigences. Ces exigences tiennent compte également des exigences définies au sein des paquetages *Interface utilisateur* et *Architecture logicielle*.

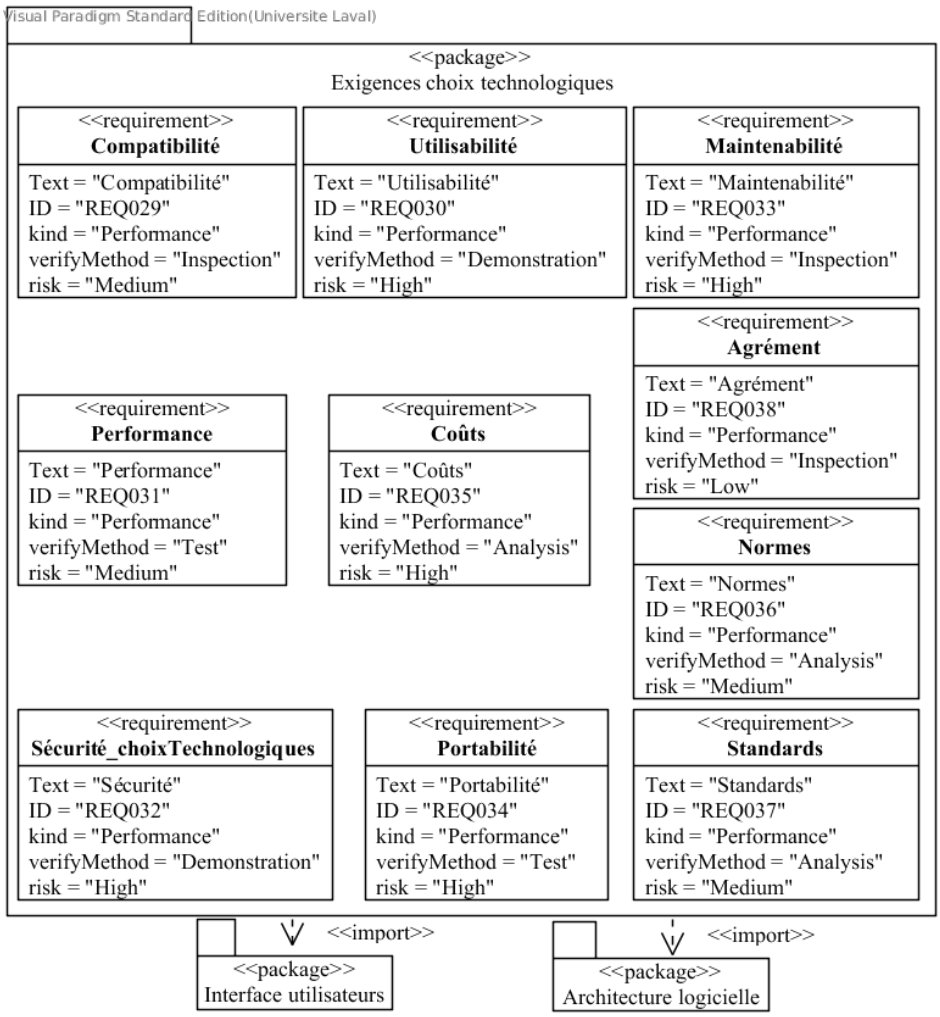


Diagramme des exigences du paquetage *Choix technologiques*.

4.4.2.1 Patrons recensés

P53 *Compatibilité :* il s'agit de l'ensemble des règles de compatibilité à respecter. Il s'agit aussi de définir le sens de la compatibilité : doit-elle être ascendante ou descendante ?

P54 *Utilisabilité :* c'est l'ensemble des règles d'utilisabilité, c'est-à-dire définissant le degré de facilité, de compréhension et d'apprentissage des choix technologiques effectués.

P55 *Maintenabilité :* il s'agit de l'ensemble des règles définissant le degré de facilité de maintenance, de modification, de correction, d'amélioration ou d'adaptation que doivent offrir les choix technologiques opérés.

P56 *Performance :* c'est l'ensemble des règles définissant la rapidité d'exécution des traitements que doivent offrir les choix technologiques faits.

P57 *Contraintes de coûts :* il s'agit de l'ensemble des contraintes budgétaires pouvant nuancer les autres exigences et nécessiter avec elles des compromis.

P58 *Agrément :* c'est l'ensemble des règles d'agrément auxquelles doivent répondre les fournisseurs des choix technologiques effectués.

P59 *Normes :* il s'agit de l'ensemble des spécifications internationales auxquelles doivent correspondre les choix technologiques opérés.

P60 *Standards :* c'est l'ensemble des règles locales auxquelles doivent correspondre les choix technologiques faits.

P61 *Sécurité_ choixTechnologiques :* il s'agit de l'ensemble des règles de sécurité auxquelles doivent répondre les choix technologiques.

P62 *Portabilité :* c'est l'ensemble des règles définissant le dégré de facilité de transfert des choix technologiques effectués d'un environnement à l'autre.

Cette liste d'exigences est partiellement extraite de la norme ISO/IEC 25010 :2011 [2].

4.4.2.2 Exemple : le patron *Portabilité*

La portabilité est l'ensemble des règles définissant le degré de facilité de transfert du LEA ou d'un de ses composants d'un environnement technologique à un autre.

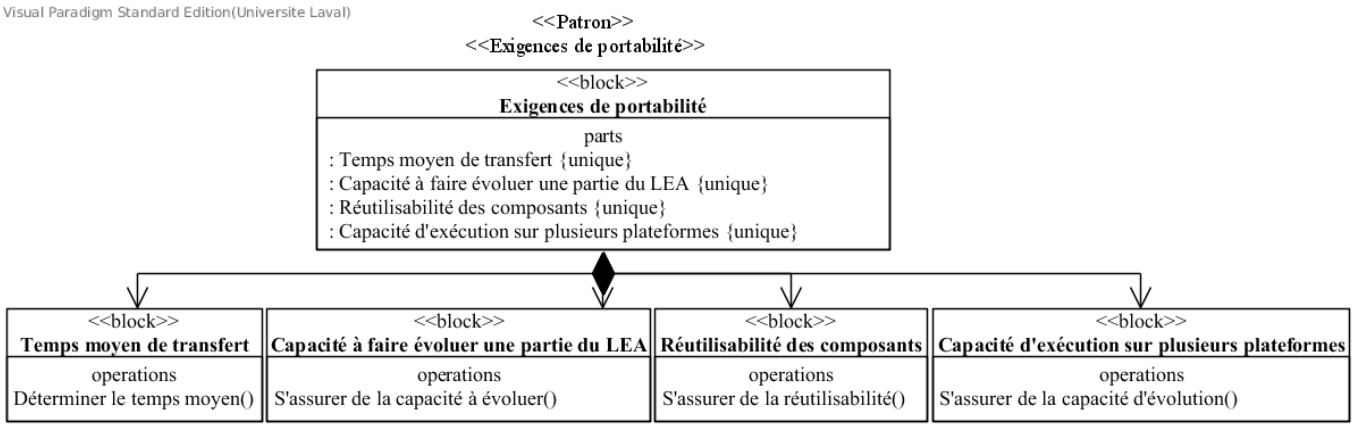


Diagramme de définition de blocs du patron *Portabilité*.

Comme activités à réaliser, il s'agit d'abord de s'assurer de la réutilisabilité des composants, c'est-à-dire de s'assurer de la capacité à réutiliser les composants du logiciel ou sa structure. Il s'agit ensuite de s'assurer de la capacité à pouvoir faire évoluer une partie du logiciel, c'est-à-dire à pouvoir changer une partie du logiciel sans que cela n'affecte le fonctionnement du logiciel ou de son environnement. Ensuite il faut s'assurer de la capacité d'exécution des composants choisis sur différentes plateformes et enfin, déterminer le temps moyen nécessaire pour transférer le logiciel d'une plate-forme ou d'un environnement à un autre (facilité d'adaptation, facilité d'installation, coexistence, interchangeabilité).

2. http://www.iso.org/iso/fr/catalogue_detail.htm?csnumber=35733.

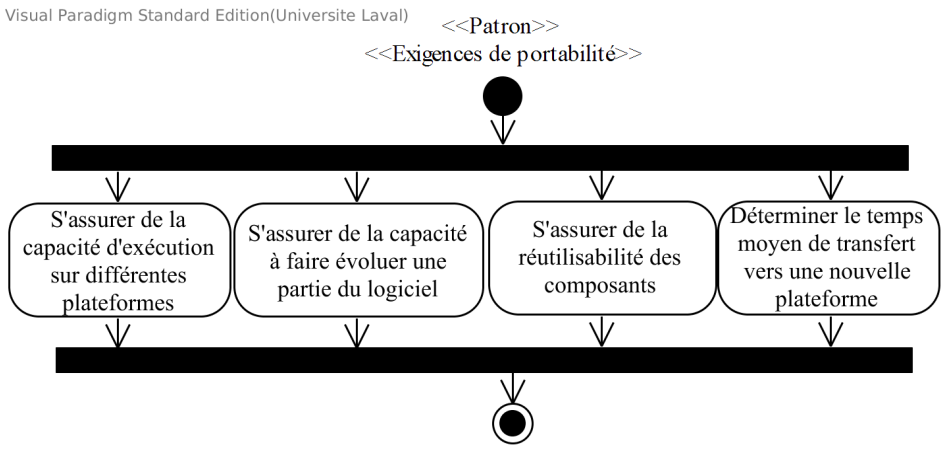
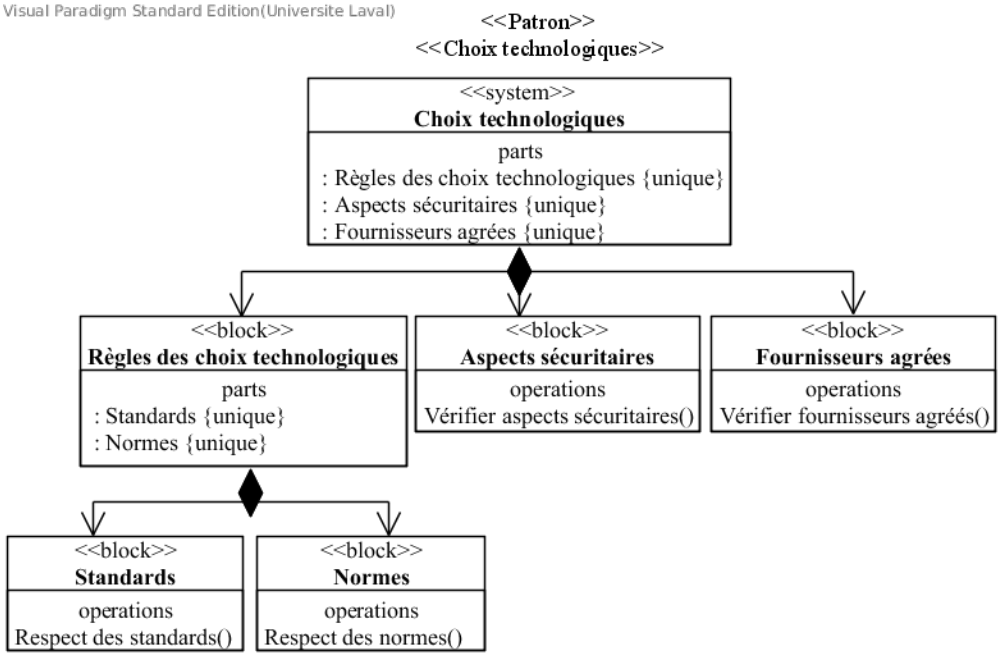


Diagramme d'activités du patron *Portabilité*.

4.4.3 Diagramme de définition de blocs



4.4.3.1 Patrons recensés

P63 *Règles des choix technologiques :* élaboration des règles des choix technologiques.

P64 *Standards :* liste des standards internationaux (ISO/IEEE ...) dont il faut tenir compte pour effectuer les choix technologiques.

P65 *Normes locales :* liste des normes locales (préférence pour une marque ou un fournisseur,...) dont il faut tenir compte pour effectuer les choix technologiques.

P66 *Aspects sécuritaires :* aspects sécuritaires des choix technologiques à effectuer.

P67 *Fournisseurs agréés :* liste de fournisseurs choisis pour la livraison des choix technologiques.

4.4.3.2 Exemple : le patron *Elaboration des règles des choix technologiques*

Ce patron permet de définir les règles permettant de choisir l'ensemble des logiciels et du matériel nécessaires au développement et au déploiement du LEA.

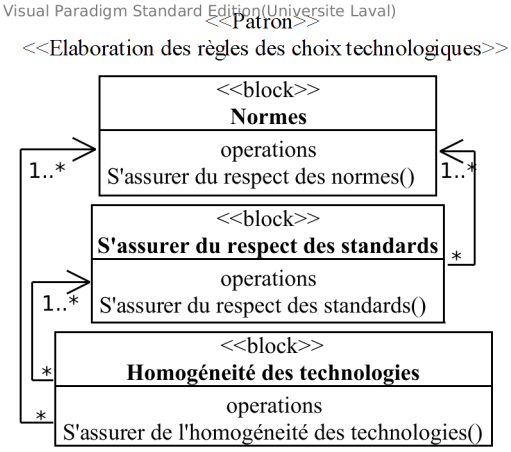


Diagramme de définition de blocs

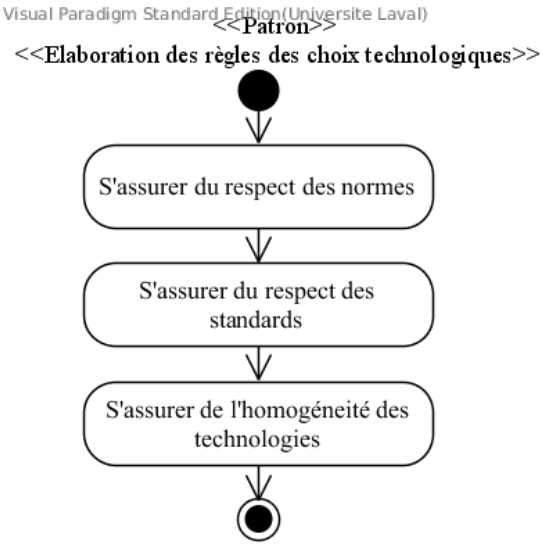


Diagramme des activités

Il s'agit d'abord de veiller à ce que les choix technologiques effectués soient conformes aux règles de l'institution qui porte la maîtrise d'ouvrage du LEA. Il s'agit aussi de veiller à ce que les choix technologiques effectués soient conformes aux règles internationales en la matière. Puis il s'agit enfin de veiller à ce que les choix technologiques effectués soient compatibles les uns avec les autres.

4.5 Le paquetage *Implémentation*

Ce paquetage a pour objectif l'implémentation dans un langage de programmation des spécifications des dossiers d'architecture et d'interface utilisateur.

4.5.1 Représentation textuelle

- `Identifiant:` PKG006.
- `Nom:` Phase-implémentation.
- `Définition:` les patrons de ce paquetage permettent de décrire comment la phase de mise en oeuvre doit se faire à partir des artéfacts de la phase de conception de l'architecture logicielle.
- `Motivation:` l'équipe de développement peut avoir des difficultés à savoir comment effectuer la transition entre la phase de conception de l'architecture logicielle et celle de l'implémentation. Les enjeux sont importants, peuvent être très complexes, et peuvent s'exprimer en termes de généricité du logiciel ou selon sa portabilité ou selon sa maintenabilité, etc. Les patrons du paquetage *Implémentation* offrent donc des balises d'implémentation à l'équipe de développement.
- `Exemple:` ces patrons peuvent servir à définir les spécifications de généricité, de modularité, de portabilité et de maintenabilité du code.
- `Applicabilité:` utiliser ce paquetage de patrons pour implémenter l'architecture d'un système dans un langage de programmation.
- `Conséquences:`
  - coût de maintenance réduits.
  - temps de mise en œuvre optimisé.
  - performances du code optimales.
  - fiabilité du code.

4.5.2 Diagramme des exigences

Dans la phase d'implémentation du logiciel les exigences formulées par les architectes et les concepteurs logiciels, sont spécifiées et structurées à l'aide du diagramme des exigences. Ces exigences tiennent compte également des exigences définies au sein des paquetages *Interface utilisateurs*, *Architecture logicielle* et *Choix technologiques*.

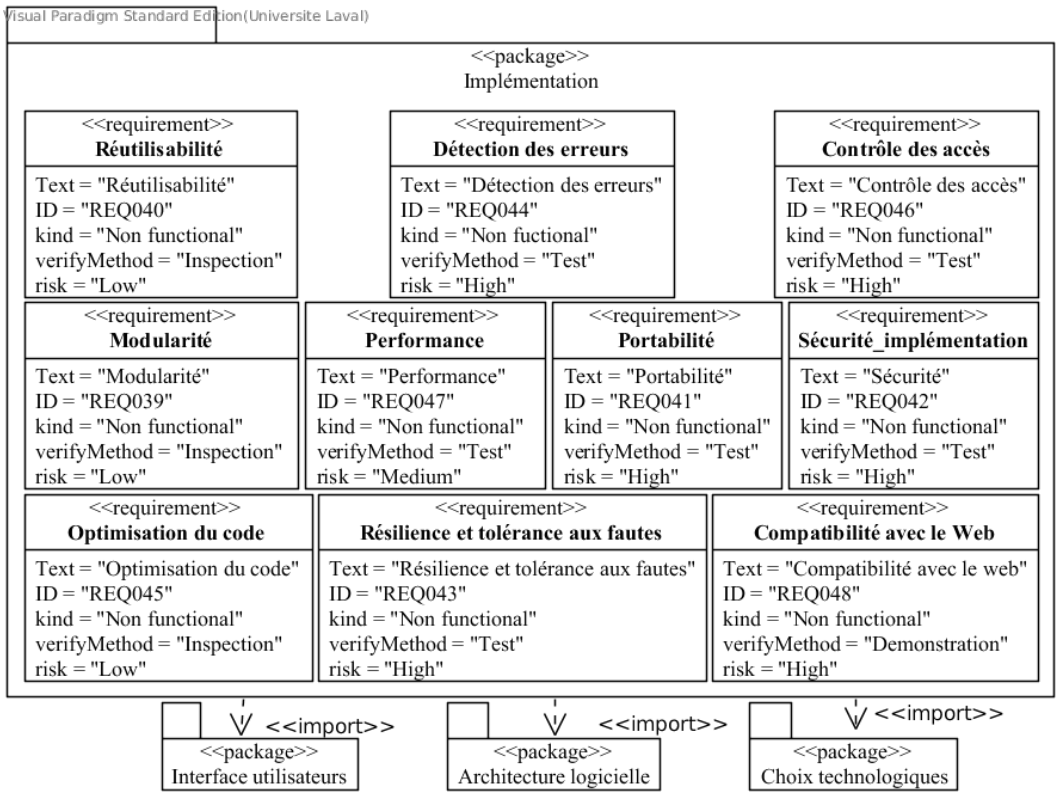


Diagramme des exigences du paquetage *Implémentation*.

4.5.2.1 Patrons recensés

P68 *Modularité :* il s'agit de définir les règles et les contraintes exigeant l'ecriture modulaire du code.

P69 *Réutilisabilité :* c'est l'ensemble des règles exigeant l'écriture de modules de code réutilisable.

P70 *Détection des erreurs :* il s'agit de définir les règles de détection des causes d'erreurs afin de les anticiper ou de les gérer efficacement.

P71 *Contrôle des accès :* c'est l'ensemble des règles exigeant le contrôle des accès aux données, ou leur modification, sans permission.

P72 *Optimisation du code :* il s'agit de définir les règles de minimisation de la complexité algorithmique, et aussi de la longueur du code. Sans ces règles, la performance, la maintenabilité et la sécurité du logiciel pourrait en pâtir.

P73 *Résilience et tolérance aux fautes :* c'est l'ensemble des règles exigeant de s'assurer qu'en cas d'erreur les fonctions importantes du logiciel s'exécutent toujours.

P74 *Sécurité_ implémentation :* il s'agit de définir les règles permettant d'éviter que les erreurs ne soient les causes d'intrusions malveillantes.

P75 *Performance :* c'est l'ensemble des règles exigeant que le temps de réponse de l'exécution d'un traitement soit toujours en deça d'une limite maximale.

P76 *Portabilité :* il s'agit de définir les règles permettant que le code soit transférable d'un environnement à un autre, et aussi soit exécutable d'un environnement à un autre.

P77 *Compatibilité avec le Web :* c'est l'ensemble des règles exigeant que le logiciel soit compatible avec les technologies du Web 3.0 et du Web 4.0.

4.5.2.2 Exemple : le patron *Optimisation du code*

Le patron *Optimisation du code* permet de définir les règles de minimisation de la complexité algorithmique, et aussi de la longueur du code. Sans ces règles, la performance, la maintenabilité et la sécurité du logiciel pourraient en pâtir.

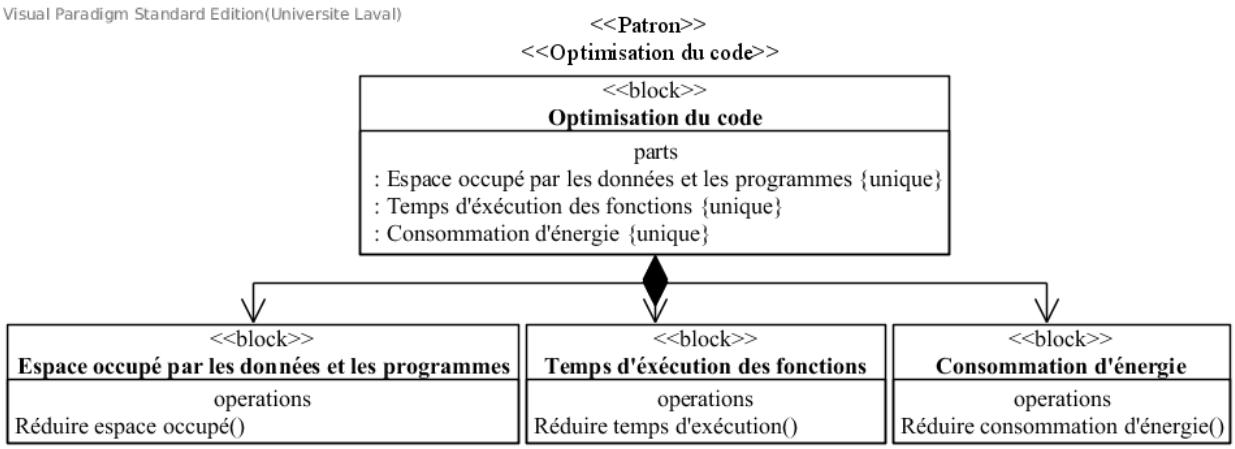


Diagramme de définition de blocs du patron *Optimisation.*

Comme activités, il s'agit d'abord de réduire le temps d'exécution des fonctions, c'est-à-dire qu'il faut veiller à ce que les fonctions soient le moins complexe possible. Ensuite il s'agit de veiller à ce que les données et les programmes occupent le moins d'espace possible sur le système qui les héberge, et enfin il s'agit de réduire la consommation d'énergie, c'est-à-dire de veiller à ce que le programme nécessite une consommation minimale d'énergie.

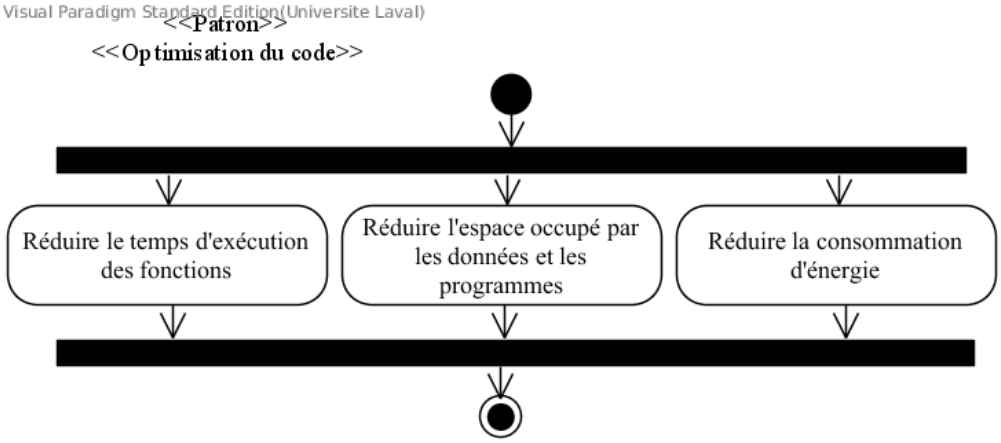


Diagramme d'activités du patron *Optimisation.*

4.5.2.3 Exemple : le patron *Résilience et tolérance aux fautes*

Le patron *Résilience et tolérance aux fautes* est l'ensemble des règles exigeant de s'assurer qu'en cas d'erreur les fonctions importantes du logiciel continueront toujours à s'exécuter.

Diagramme de définition de blocs

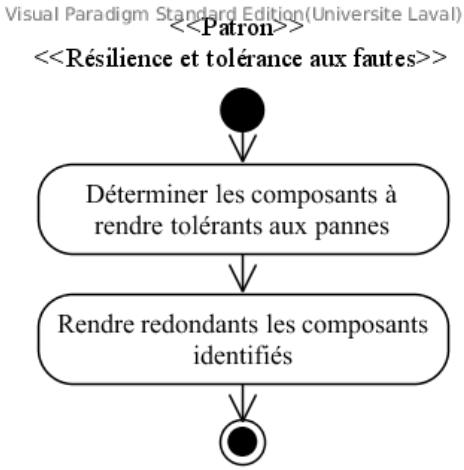
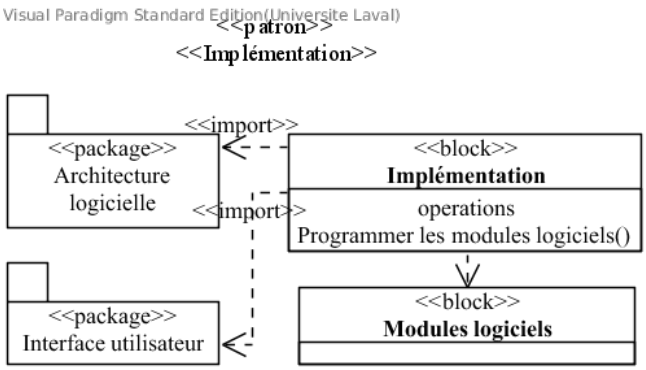


Diagramme des activités

La première activité consiste à déterminer les composants à rendre tolérants aux pannes. Il faut pour cela mettre en place une stratégie permettant au système de continuer à fonctionner même si certains de ses composants critiques tombent en panne. Ensuite, il s'agit de rendre redondants les composants identifiés en mettant en place une stratégie permettant à un composant de prendre automatiquement la relève lorsqu'un composant tiers tombe en panne.

4.5.3 Diagramme de définition de blocs

Les patrons du paquetage *Implémentation* dépendent des paquetages *Architecture logicielle*, et *Interface utilisateur* dont il doit satisfaire des exigences.



4.5.3.1 Patron recensé

P78 *Programmer les modules logiciels :* traduction en langage de programmation des modules de l'architecture logicielle.

4.5.3.2 Exemple : le patron *Programmation des modules logiciels*

Le patron *Programmation des modules logiciels* permet de définir l'ensemble des règles de traduction en langage de programmation des modules de l'architecture logicielle.

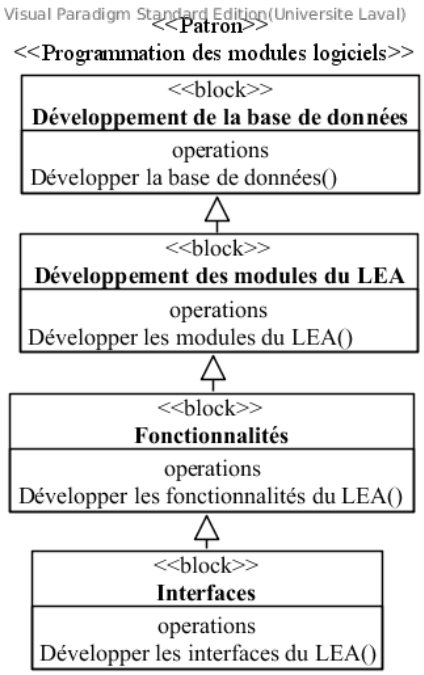


Diagramme de définition de blocs

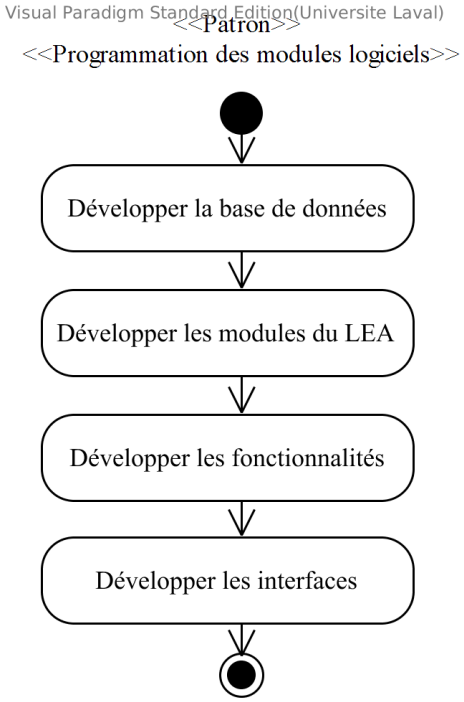


Diagramme des activités

Il s'agit du développement de la base de données du LEA et du développement des fonctionnalités du LEA.

4.6 Le paquetage *Validation*

Il permet d'établir la preuve documentée que les spécifications et exigences ont été respectées. Il permet pour cela de tester la nouvelle plateforme, afin d'y déceler d'éventuels problèmes et de ce fait procéder aux changements conséquents avant la phase de déploiement.

Le paquetage *Validation* a pour objectif la vérification du respect des spécifications et exigences implémentées durant la phase précédente d'implémentation dans un langage de programmation. Elle permet de vérifier la compatibilité du nouveau système avec chacune des spécifications et exigences déterminées pendant tout le processus de conception. Pour cela, des tests de conformité, de performance et de robustesse sont effectués après l'assemblage des différents composants développés de façon isolée, du nouveau logiciel, et doivent être tous passés avec succès.

4.6.1 Représentation textuelle

- `Identifiant:` PKG007.
- `Nom:` Validation-logiciel-enseignement.
- `Définition:` ce patron permet de baliser la phase de validation ou de tests du LEA.
- `Mots-clés:` validation, tests.
- `Motivation:` du fait de sa complexité et de son caractère multidisciplinaire, il sera certainement facile et intuitif à l'équipe de développement du logiciel de procéder aux tests classiques effectués sur un logiciel. Cependant, l'équipe de développement risque d'oublier d'effectuer les tests sur les aspects qui caractérisent même le LEA : les aspects pédagogiques. Ces patrons décrivent donc les tests de validation à effectuer et comment il faut les faire.
- `Exemple:` test des exigences d'architecture, test des exigences d'intégration, test des exigences pédagogiques, etc.
- `Applicabilité:` utiliser ce paquetage de patrons pour tester un système et s'assurer qu'il répond en tous points aux exigences et contraintes qu'il doit respecter.
- `Conséquences:` aucune conséquence identifiée.

4.6.2 Diagramme des exigences

Dans la phase de validation les exigences formulées par les experts en technologie éducative, les experts en formation en distance, les architectes et les concepteurs logiciels, sont spécifiées et structurées à l'aide du diagramme des exigences. Ces exigences tiennent compte également de celles définies au sein des paquetages *Exigences pédagogiques*, *Interface utilisateur* et *Architecture logicielle*.

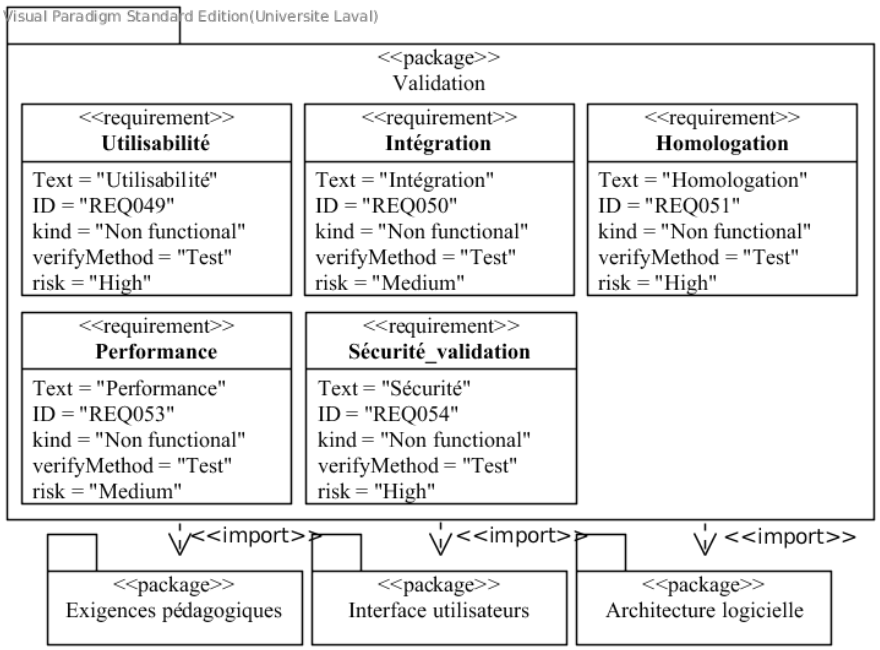


Diagramme des exigences du paquetage *Validation*.

4.6.2.1 Patrons recensés

P79 *Utilisabilité :* il s'agit de définir les règles de vérification de l'acceptabilité du logiciel auprès des utilisateurs.

P80 *Intégration :* c'est l'ensemble des règles définissant les tests d'intégration ou tests unitaires.

P81 *Homologation :* il s'agit de définir les règles de vérification du respect des exigences d'intégration fonctionnelle.

P82 *Performance :* c'est l'ensemble des règles vérifiant la conformité aux exigences de performance.

P83 *Sécurité_ validation :* il s'agit de définir les règles de vérification du respect des exigences de sécurité du logiciel.

4.6.2.2 Exemple : le patron *Utilisabilité*

Le patron *Utilisabilité* permet de définir les règles de vérification de l'acceptabilité du LEA auprès des utilisateurs.

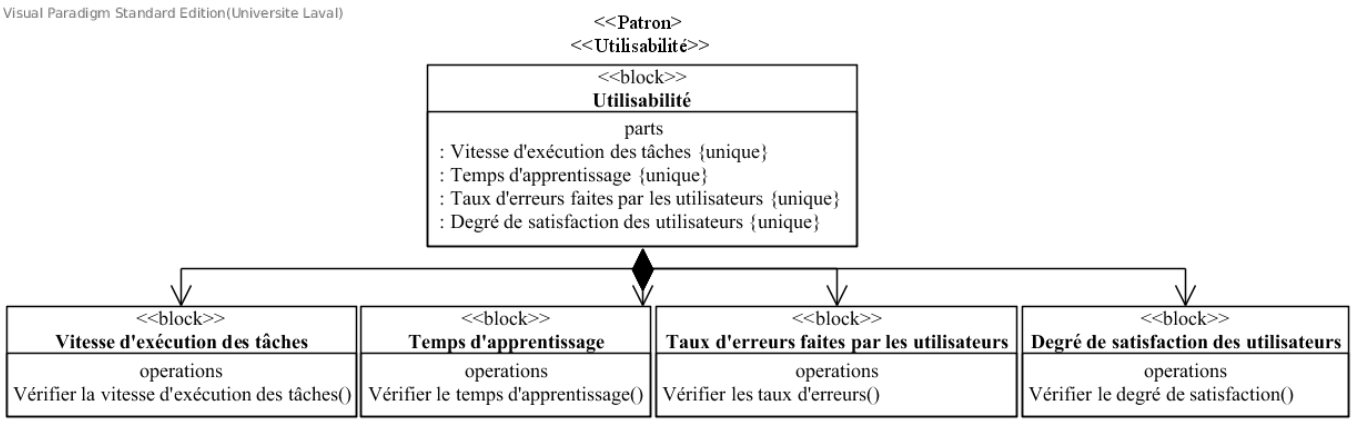


Diagramme de définition de blocs du patron *Utilisabilité*.

Le diagramme d'activités du patron *Utilisabilité* spécifie d'abord de vérifier le temps d'apprentissage du LEA, c'est-à-dire évaluer le temps de familiarisation de l'utilisateur avec les interfaces du LEA. Ensuite il faut vérifier la vitesse d'exécution des tâches du LEA, c'est-à-dire évaluer la réactivité du LEA. Enfin, il s'agit de vérifier les taux d'erreurs faites par les utilisateurs, c'est-à-dire évaluer la capacité de l'utilisateur à choisir rapidement la fonctionnalité qu'il désire exécuter, puis de vérifier le dégré de satisfaction des utilisateurs.

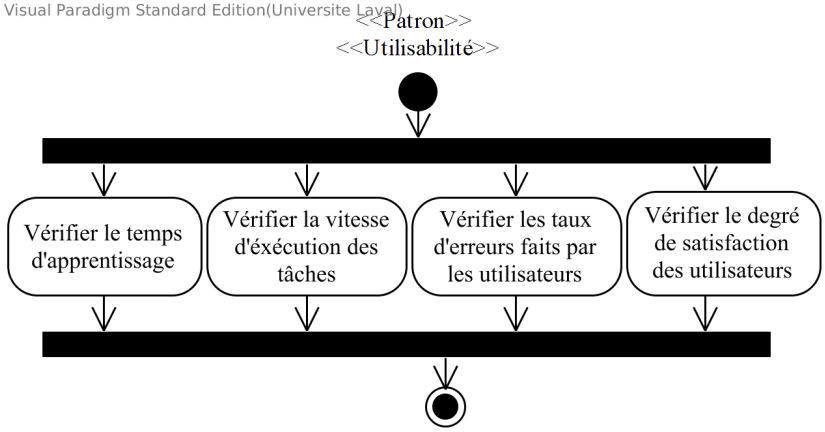
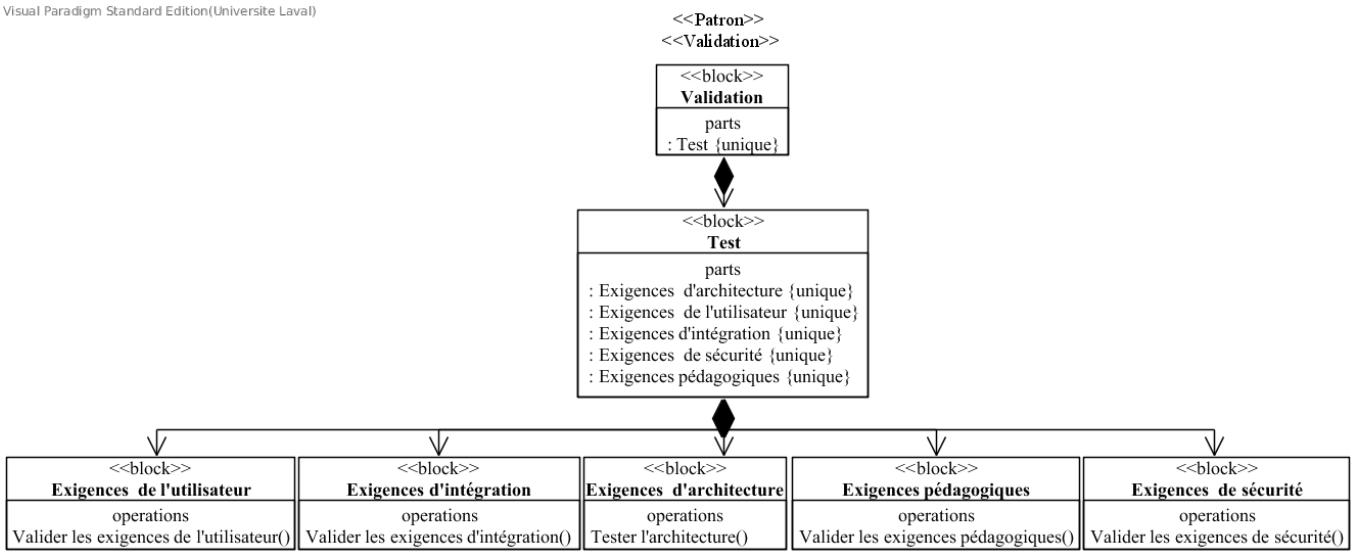


Diagramme d'activités du patron *Utilisabilité*.

4.6.3 Diagramme de définition de blocs



4.6.3.1 Patrons recensés

P84 *Tests :* type de tests à effectuer ;

P85 *Sécurité :* validation des exigences de sécurité.

P86 *Test_ exig_ architecture :* validation des exigences d'architecture.

P87 *Test_ exig_ pédagogiques :* validation des exigences pédagogiques.

P88 *Test_ exig_ utilisateur :* validation des exigences de l'utilisateur.

P89 *Intégration :* validation des exigences d'intégration.

4.6.3.2 Exemple : le patron *Test_ exig_ architecture*

Le patron *Architecture* permet de décrire l'ensemble des règles décrivant le processus de validation de l'architecture logicielle du LEA.

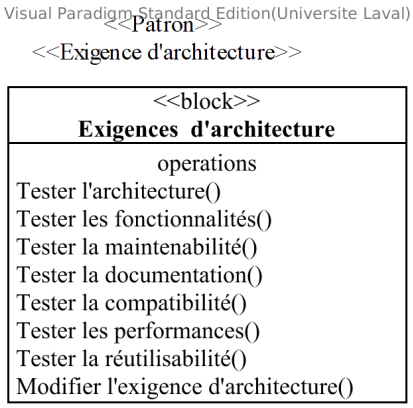


Diagramme de définition de blocs

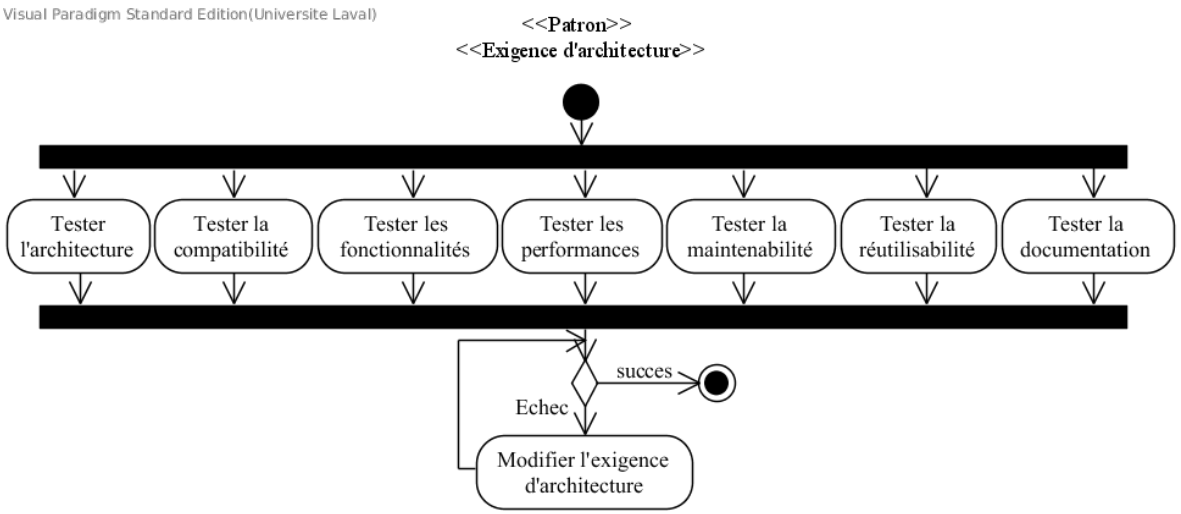


Diagramme des activités

Il s'agit d'abord de choisir l'exigence d'architecture à tester, du test de cette exigence et du fait de faire si nécessaire les modifications adéquates au niveau de l'architecture du logiciel. Le processus de test reprend jusqu'à ce que les résultats du test soient satisfaisants.

4.6.3.3 Exemple : le patron *Test_ exig_ utilisateur*

Le patron *Exigences de l'utilisateur* décrit les règles décrivant le processus de validation des exigences de l'utilisateur.

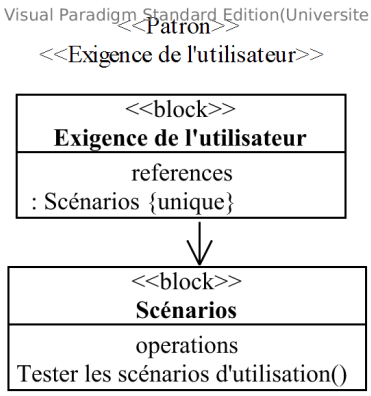


Diagramme de définition de blocs

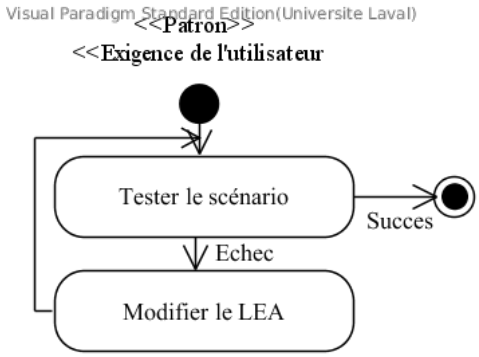


Diagramme des activités

L'activité consiste en la définition d'un scénario décrivant un ensemble de fonctionnalités à tester, et du test du scénario considéré. En cas d'échec, les fonctionnalités en jeu sont adéquatement modifiées et le processus de test relancé.

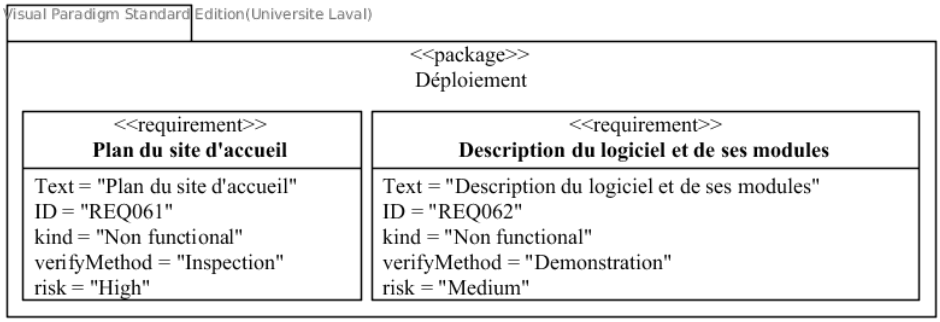
4.7 Le paquetage *Déploiement*

L'objectif de ce paquetage est d'installer le nouveau logiciel dans son environnement d'exploitation normal. Le processus d'installation du LEA comprend une seule activité majeure qui est celle de la définition des configurations : il s'agit du processus de déploiement du nouveau logiciel dans son environnement d'exploitation. Des configurations pour ajuster le nouveau logiciel à son nouvel environnement sont indispensables et celles-ci doivent être soigneusement documentées.

4.7.1 Représentation textuelle

- `Identifiant:` PKG008.
- `Nom:` Déploiement-logiciel-enseignement.
- `Définition:` ce patron définit comment la mise en service du LEA doit s'effectuer, ses paramétrages en fonction du contexte du client, les ressources de configuration dont il faut tenir compte, etc.
- `Motivation:` au moment du déploiement du logiciel, l'inexpérience de l'équipe chargée de son déploiement peut entraver le processus ou le rendre davantage complexe et en proportionnellement augmenter les coûts si par exemple il dure plus longtemps que prévu. L'équipe de déploiement doit disposer d'informations exhaustives sur l'environnement de mise en service du logiciel, et doit effectuer une planification minutieuse. Les patrons de la phase de déploiement permettent de baliser la phase de déploiement du LEA.
- `Exemple:` intégration des données, migration des systèmes et des données, etc.
- `Applicabilité:` utiliser ce paquetage de patrons pour déployer un système, c'est-à-dire le mettre en oeuvre dans l'environnement d'accueil où il est censé régulièrement fonctionner.
- `Conséquences:` aucune conséquence identifiée.

4.7.2 Diagramme des exigences



4.7.2.1 Patrons recensés

P90 *Plan du site d'accueil :* il s'agit de définir les règles exigeant de connaître à l'avance le contexte environnemental matériel et logiciel dans lequel le logiciel doit être déployé.

P91 *Description du logiciel et de ses composants :* c'est l'ensemble des règles exigeant de disposer de suffisamment d'information sémantiques sur les contraintes matérielles, logicielles, les dépendances, ..., pour davantage faciliter le déploiement.

4.7.2.2 Exemple : le patron *Plan du site d'accueil*

Le patron *Plan du site d'accueil* permet de définir les règles d'installation du LEA dans son environnement d'accueil.

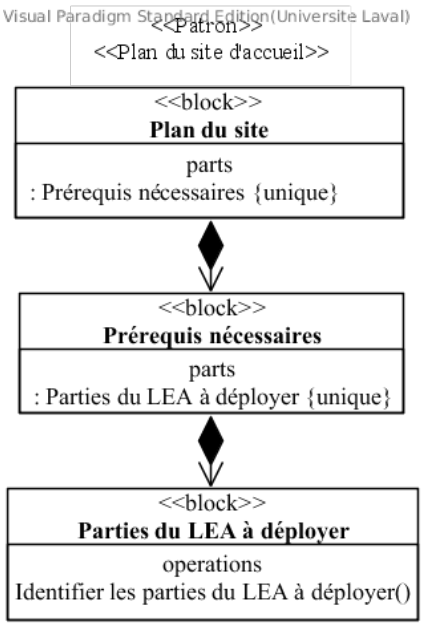


Diagramme de définition de blocs

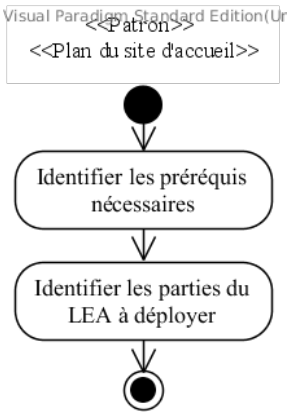
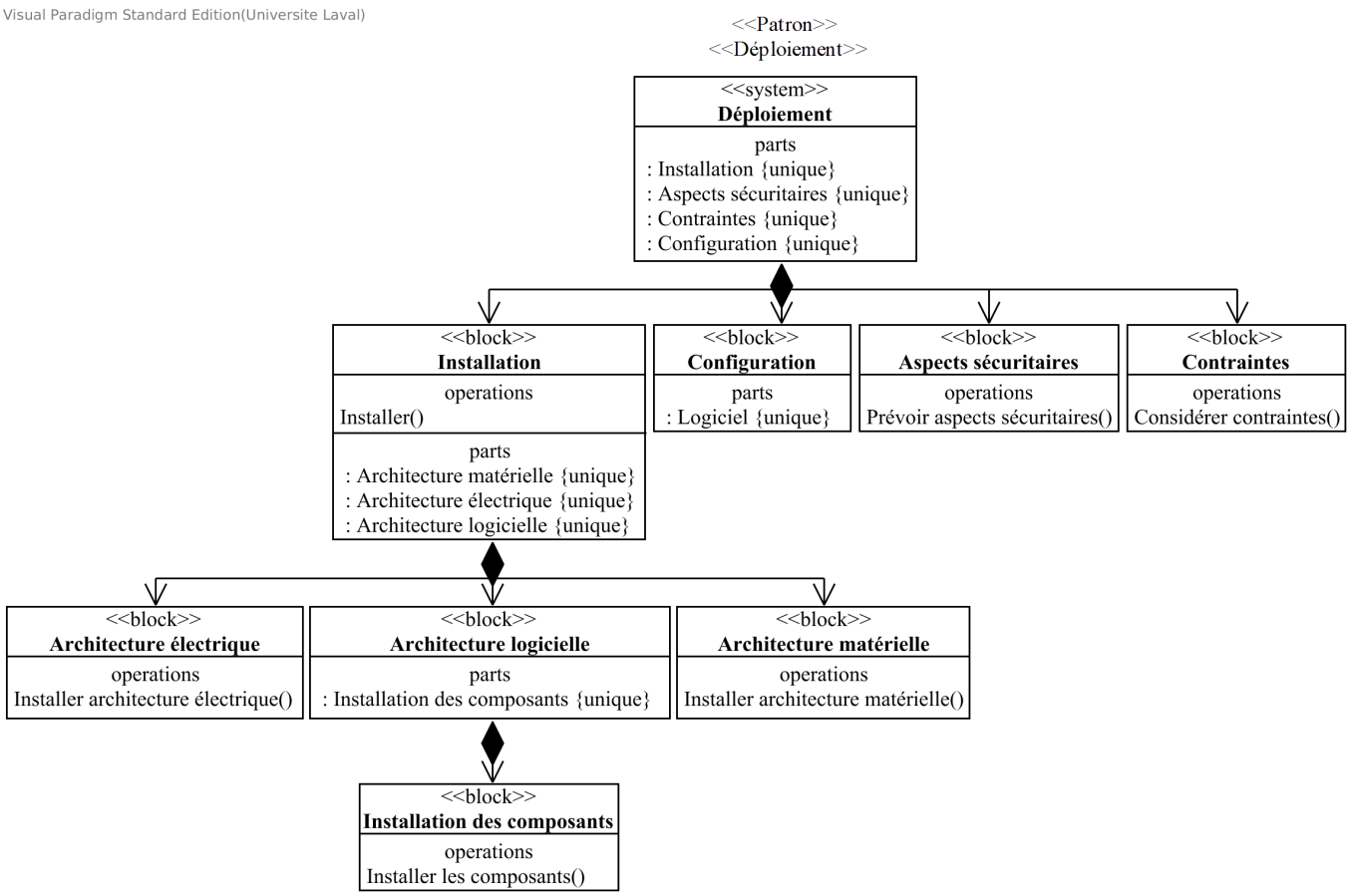


Diagramme des activités

La première activité consiste à décrire d'abord tous les prérequis nécessaires au déploiement du LEA dans son environnement d'accueil. Ces prérequis peuvent être, par exemple, l'espace disque nécessaire sur le serveur d'hébergement, la capacité optimale du processeur, les librairies nécessaires, etc. Il faut ensuite identifier les parties du LEA à déployer.

4.7.3 Diagramme de définition de blocs



4.7.3.1 Patrons recensés

P92 *Installation :* étapes d'installation du LEA.

P93 *Architecture logicielle :* étapes d'installation des modules implémentés lors de la phase d'implémentation.

P94 *Architecture matérielle :* étapes d'installation des composants matériels sur lesquels la plateforme logicielle doit être hébergée.

P95 *Architecture électrique :* ajustement de la configuration électrique à prévoir en fonction des composants matériels installés.

P96 *Configuration :* paramètres de configuration de la plateforme logicielle à définir.

P97 *Contraintes :* ensemble des contraintes particulières pouvant affecter le processus de déploiement.

P98 *Aspects sécuritaires :* paramètres de sécurité à ajuster ou compléter.

4.7.3.2 Exemple : le patron *Configuration*

Le patron *Configuration* permet de définir les paramètres du LEA en vue de sa mise en production, et de l'environnement d'accueil du LEA.

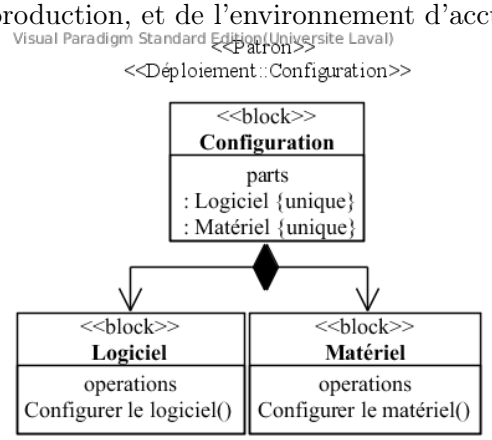


Diagramme de définition de blocs

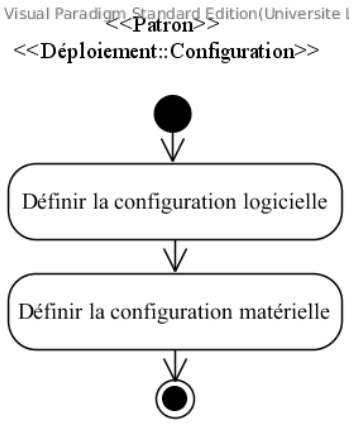


Diagramme des activités

Il s'agit d'abord de définir les paramètres logiciel du LEA. L'environnement logiciel comprend les logiciels systèmes tels que le système d'exploitation, les machines virtuelles, les pilotes, etc., et les logiciels d'application avec lesquels doit interagir le LEA. Ensuite, il faut aussi définir la configuration matérielle du LEA. L'environnement matériel comprend les serveurs physiques d'hébergement et les connectivités locale, Internet et électrique adéquates.

4.7.3.3 Exemple : le patron *Aspect sécuritaires*

Le patron *Aspect sécuritaires* permet de décrire les aspects sécuritaires à considérer dans le cadre du déploiement du LEA.

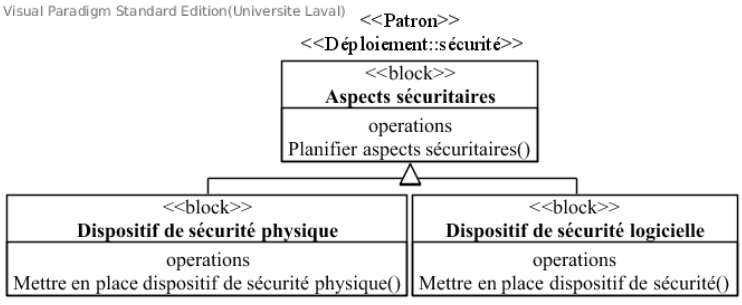


Diagramme de définition de blocs

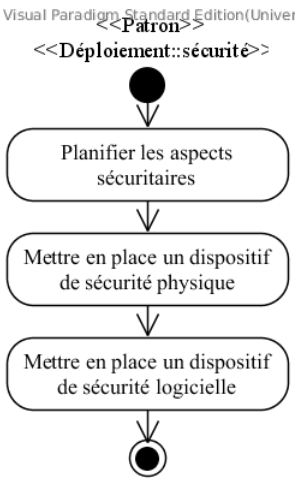


Diagramme des activités

La première activité consiste en la mise en place d'un dispositif de sécurité physique pour sécuriser l'accès au serveur physique d'hébergement du LEA. Ensuite il s'agit de mettre en place un dispositif de sécurité logicielle pour sécuriser l'accès au LEA à l'aide d'un dispositif logiciel tel qu'un pare-feu, un antivirus, etc.

4.8 Le paquetage *Evolution*

Il permet de faire évoluer le logiciel pour y ajouter de nouvelles fonctionnalités ou de le corriger pour rectifier des erreurs. Il a pour objectif la mise en oeuvre d'une politique de maintenance du logiciel.

4.8.1 Représentation textuelle

- `Identifiant:` PKG009.
- `Nom:` Evolution-logiciel-enseignement.
- `Définition:` ce patron décrit la manière dont des modifications doivent être apportées à un logiciel après sa mise en oeuvre soit pour en corriger les erreurs soit pour le faire évoluer.
- `Mots-clés:` évolution, maintenance.
- `Motivation:` l'équipe de développement du logiciel peut ne pas être en mesure de déterminer les ordres de priorité des différentes tâches de maintenance. Ces patrons permettent de baliser donc le processus de maintenance du logiciel.
- `Exemple:` maintenance corrective, maintenance adaptative, maintenance perfective, maintenance préventive.
- `Applicabilité:` utiliser ce paquetage de patrons pour effectuer la maintenance d'un système et le faire évoluer d'un état A à un état B.
- `Conséquences:` aucune conséquence identifiée.

4.8.2 Diagramme des exigences

Dans la phase d'évolution du logiciel, les exigences formulées par les experts en technologie éducative, les experts en formation à distance, les architectes et les concepteurs logiciels, sont spécifiées et structurées à l'aide du diagramme des exigences. Ces exigences tiennent compte également des exigences définies au sein des paquetages *Interface utilisateur*, *Architecture logicielle* et *Choix technologiques*. L'ensemble des exigences d'évolution doit par ailleurs vérifier les exigences d'implémentation puisque l'évolution ou la maintenance consiste en une reprise partielle de cette phase.

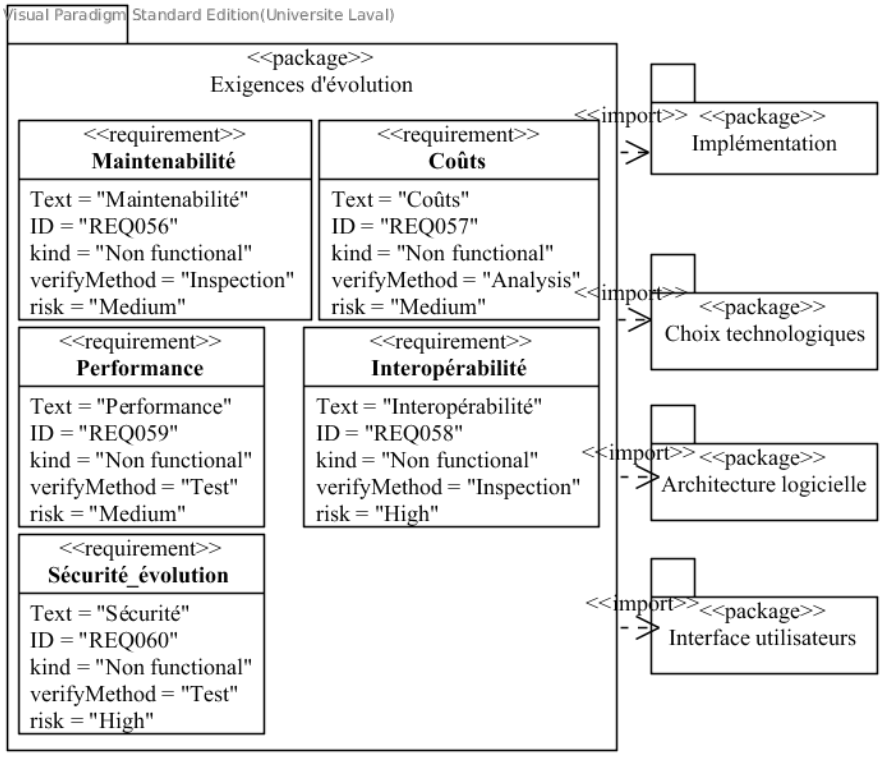


Diagramme des exigences du paquetage *Evolution*.

4.8.2.1 Patrons recensés

P99 *Maintenabilité :* il s'agit de définir le dégré de maintenabilité et le type de maintenance à appliquer.

P100 *Coûts :* c'est l'ensemble des contraintes budgétaires à prendre en compte.

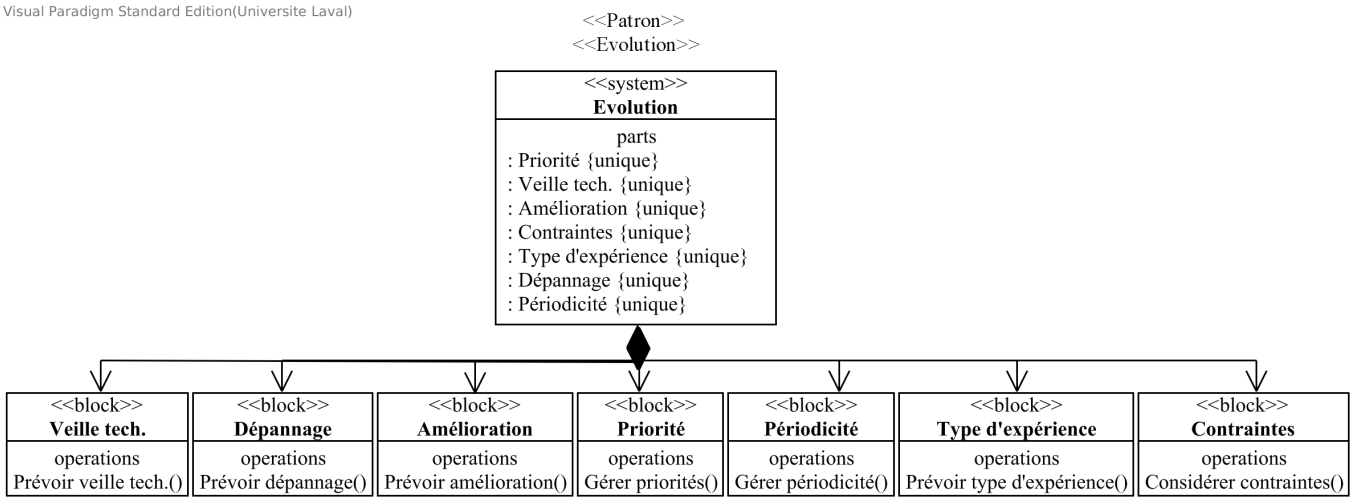
P101 *Performance :* il s'agit de déterminer les enjeux des modifications à effectuer sur la performance établie du logiciel.

P102 *Interopérabilité :* c'est l'ensemble des règles permettant de déterminer d'une part si le problème à corriger interagit sur d'autres modules, et d'autre par le degré d'interopérabilité du module ou des modules modifiés avec le reste du système.

P103 *Sécurité_ évoluation* : il s'agit de l'ensemble des règles à implémenter pour assurer une bonne évolutivité du LEA.

Les exigences du paquetage *Evolution* reprenant la majeure partie des exigences des autres paquetages, nous ne présentons pas ici de nouveaux exemples issus du diagramme des exigences de ce paquetage.

4.8.3 Diagramme de définition de blocs



4.8.3.1 Patrons recensés

P104 *Veille technologique :* stratégie de veille technologique à définir.

P105 *Dépannage :* type de dépannage à effectuer.

P106 *Amélioration :* stratégie d'évolutivité à définir.

P107 *Priorité :* stratégie de priorisation des dépannages et des améliorations à définir.

P108 *Périodicité :* périodicité des améliorations à définir.

P109 *Type d'expérience :* niveau d'expérience requis pour effectuer un dépannage ou une amélioration à définir.

P110 *Contraintes :* ensemble des contraintes particulières pouvant affecter le processus d'évolution.

4.8.3.2 Exemple : le patron *Veille technologique*

Ce patron permet de décrire les activités à mener dans le cadre d'un processus de veille technologique visant à maintenir le LEA ou à le faire évoluer.

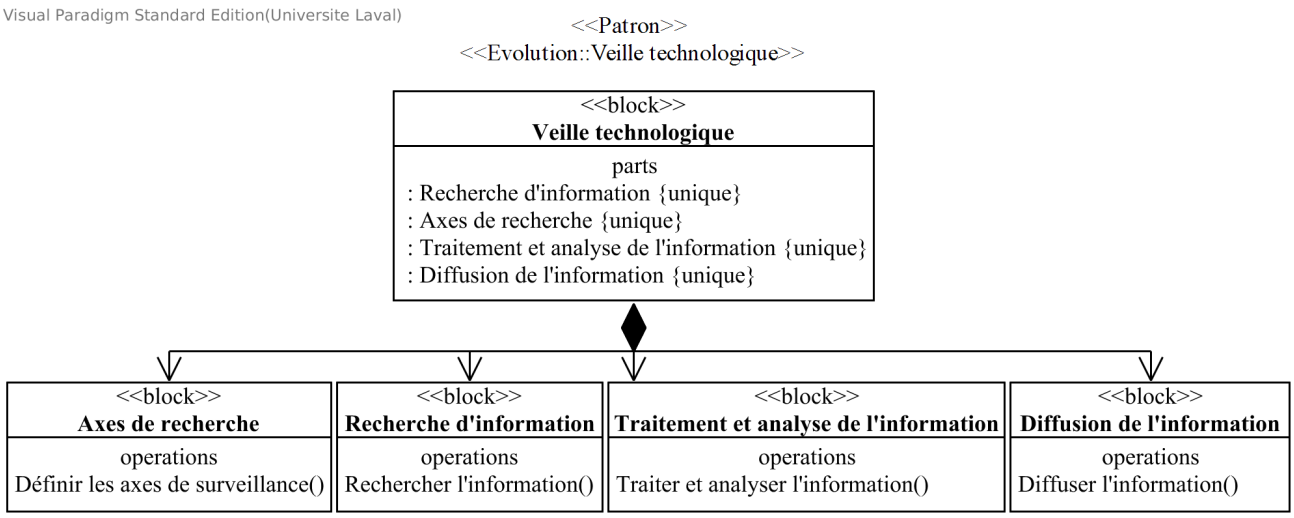


Diagramme de définition de blocs du patron *Veille technologique*.

Il s'agit d'abord de définir les axes sur lesquels doivent porter la veille technologique, de rechercher l'information, de la traiter et de l'analyser pour pouvoir la diffuser.

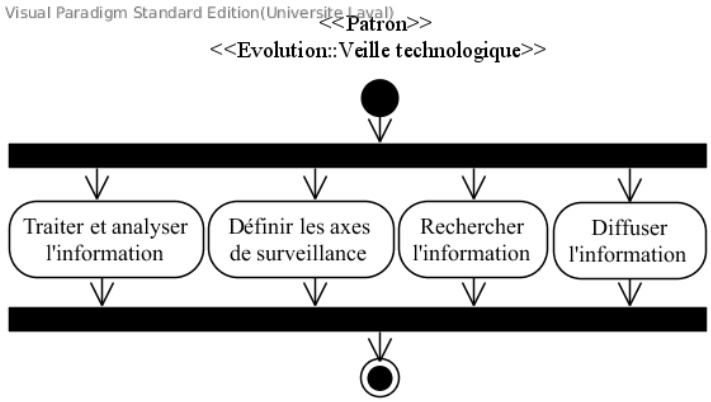


Diagramme d'activités du patron *Veille technologique*.

4.8.3.3 Exemple : le patron *Dépannage*

Le diagramme de définition de blocs du patron *Dépannage* représente les principales structures qui par leurs interactions permettent à l'utilisateur de pouvoir personaliser ses interfaces graphiques.

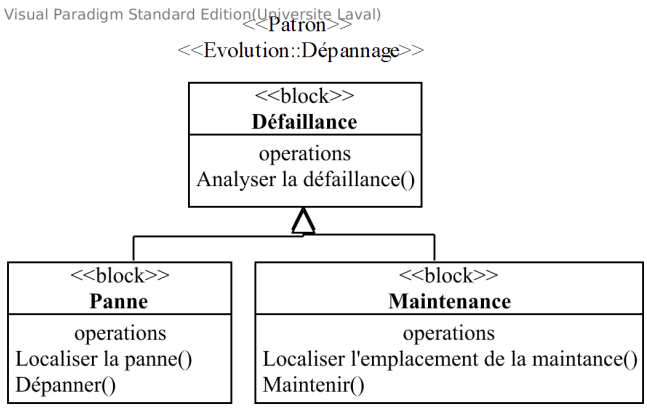


Diagramme de définition de blocs du patron *Dépannage*.

Le diagramme d'activités du patron *Dépannage* indique comme première activité d'analyser le cas de défaillance afin de savoir s'il s'agit d'un cas de panne ou de maintenance. Il faut ensuite soit dépanner le système ou le maintenir et ensuite effectuer des tests afin d'évaluer son état fonctionnel après les travaux effectués.

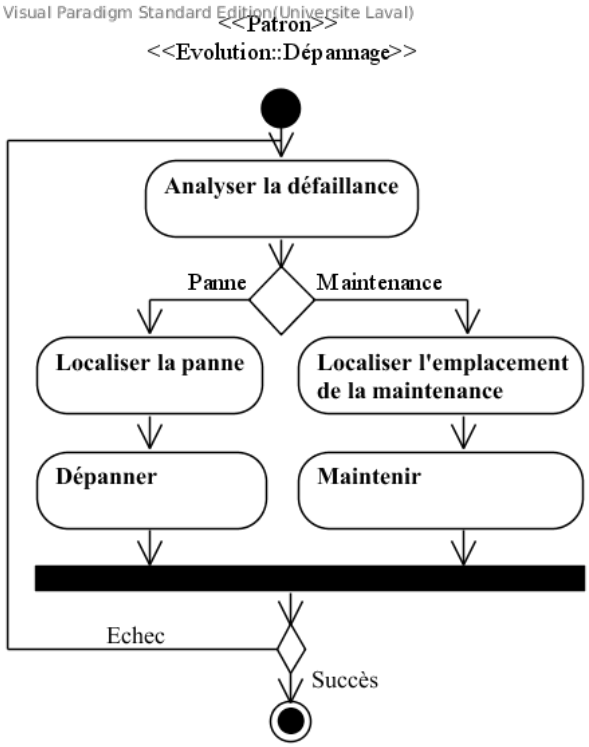


Diagramme d'activités du patron *Dépannage*.

4.9 Conclusion

Dans ce chapitre, nous avons présenté le catalogue constitué de 110 patrons adéquats sur lequel se base l'approche de conception des LEA que nous avons présentée au chapitre 3.

Pour chaque groupe ou paquetage de patrons, nous avons présenté sa représentation textuelle,

son diagramme des exigences et son diagramme de définition de blocs. Pour chacun des diagrammes présentés, nous avons aussi présenté des exemples de patrons décrits eux aussi à l'aide d'un diagramme de définition de blocs et d'un diagramme d'activités.

Dans le prochain chapitre, nous procèderons à la validation de l'approche de conception des LEA proposée au chapitre 3, et à la validation des patrons proposés dans le présent chapitre. Ce processus de validation nous permettra d'évaluer le réalisme et la faisabilité de la démarche de conception que nous avons proposée, et nous permettra aussi de vérifier que les patrons proposés sont valides et fonctionnels.

Chapitre 5

Cadre de validation de la démarche de conception des LEA basée sur les patrons

Ce chapitre présente le cadre de validation de la démarche de conception des LEA basée sur les patrons. La validation est le processus de détermination du degré de représentativité du monde réel par le modèle proposé (Frey & Dym, 2006) et permet de tester l'approche que nous proposons dans une situation de conception d'un LEA.

Le processus de validation dans le cadre de cette thèse répond aux objectifs suivants :

1. évaluer le réalisme et la faisabilité de la démarche de conception proposée : s'assurer du fait qu'un modèle d'architecture, puis un LEA peut être developpé en suivant les phases proposées dans la démarche ;
2. vérifier que les patrons proposés sont valides et fonctionnels.

La méthode de validation a consisté à suivre de façon systématique les phases de la démarche de conception des LEA, et les patrons, proposés dans le cadre du chapitre 4. Pour chacune des phases de la démarche de conception, nous présentons la suite d'activités à mener, les patrons associés à ces activités et faisant partie du répertoire de patrons que nous avons proposé, et s'il y a lieu les exigences associées aux patrons présentés.

Nous présentons deux cas d'application de l'approche, ainsi que les patrons que nous avons utilisés. Le premier cas concerne la conception d'un module logiciel d'analyse et d'évaluation de la participation estudiantine devant être intégré au forum de discussion Knowledge Forum 6 (KF6), présenté à la section 5.1. Le deuxième cas présenté à la section 5.2 consiste en la conception d'un forum de discussion d'enseignement et d'apprentissage. Ce chapitre s'achève par la présentation d'un guide d'utilisation du répertoire de patrons, présenté à la section 3.2.2.

5.1 Etude de cas 1 : conception d'un module logiciel d'analyse et d'évaluation de la participation estudiantine

Cette première étude de cas concerne la conception d'un module logiciel d'analyse et d'évaluation de la participation estudiantine. Cette étude de cas fait suite à un besoin formulé par Mme Thérèse Laferrière et son équipe. Mme Laferrière est professeure à la Faculté des sciences de l'éducation de l'Université Laval, au département d'études sur l'enseignement et l'apprentissage et responsable du centre de recherche *TéléApprentissage Communautaire et Transformatif* (TACT) qui *s'intéresse aux communautés qui renforcent leur capacité d'apprendre, de faire apprendre et d'élaborer ensemble des connaissances en utilisant des technologies numériques pour augmenter et bonifier leurs interactions en vue de la production d'un savoir collectif*[1]. A notre demande, la professeure Laferrière a accepté de nous proposer un contexte réel pour la validation de notre approche de conception des LEA.

Dans la section 5.1.1 nous présentons le contexte, c'est-à dire les motivations ayant conduit au développement du logiciel d'analyse et d'évaluation de la participation estudiantine que nous avons développé. Puis dans la section 5.1.2, nous présentons les différentes étapes de conception du logiciel.

5.1.1 Contexte

Knowledge Forum est un forum de discussion conçu pour soutenir une démarche de *coélaboration*[2]. Il est développé par un consortium d'universités dont l'Université de Toronto et l'Université Laval à travers l'équipe du centre de recherche TACT.

L'équipe du centre de recherche TACT travaille sur la version 6 en cours de développement de Knowledge Forum, et souhaite lui intégrer des outils d'analyse et d'évaluation de la participation estudiantine. La version 6 de Knowledge Forum permet actuellement aux enseignants de créer des catégories et des fils de discussion relatifs au cadre de leurs cours respectifs, et aux étudiants de poser des questions de compréhension sur le matériel du cours, ou de communiquer avec leurs pairs.

Knowledge Forum 6 est developpé en langage JavaScript, à l'aide de la pile d'outils MEAN 0.5.6[3], constituée de l'ensemble des plateformes de développement suivantes :

- MongoDB 3.0[4] : un Système de Gestion de Bases de Données (SGBD) orienté documents ;

1. http://tact.ulaval.ca/page/%C3%A0-propos-de-tact
2. Coélaboration (de l'anglais *knowledge building)* : processus infini d'amélioration et de création de connaissances.
3. http://mean.io
4. https://www.mongodb.org/

- ExpressJS 4.0[5] : une plateforme de développement d'applications web en *Nodejs* ;
- Node.js 4.2.0[6] : une plateforme de développement d'applications évènementielle en JavaScript ;
- AngularJS 1.4.7[7] : une plateforme de développement d'applications en JavaScript développé par Google.

5.1.1.1 Existant

Un forum de discussion Knowledge Forum 6 asynchrone dont les messages sont archivés dans une base de données d'un SGBD orienté documents.

5.1.1.2 Besoin de l'équipe du centre de recherche TACT

L'équipe du centre de recherche TACT souhaite outiller les différentes équipes pédagogiques utilisant Knowledge Forum 6 en les dotant d'un logiciel d'analyse et d'évaluation de la participation estudiantine sur le forum Knowledge Forum 6 aux fins de rétroaction pour orienter la démarche de coélaboration de connaissances.

5.1.1.3 Résultat escompté

Le logiciel d'analyse et d'évaluation de la participation estudiantine doit être intégré à Knowledge Forum 6 et permettre aux utilisateurs que sont l'apprenant, l'enseignant et le gestionnaire, selon le niveau de privilèges de :

- consulter le nombre de contributions par personne ;
- consulter la densité des contributions par groupe ;
- consulter la densité des mots utilisés dans les messages de contribution ;
- consulter ces statistiques à partir de n'importe quel terminal.

Nous décrivons dans la section 5.1.2 le processus de conception du logiciel d'analyse de la participation estudiantine. Les éléments conceptuels des cas d'utilisation réalisés sont graphiquement représentés à l'aide de l'outil de modélisation `Visual paradigm 12.1`[8] dont la version utilisée dans le cadre de ce processus de validation est la version 12.1 et la licence d'utilisation, la licence académique standard.

5.1.2 Approche de conception

Dans le cadre du processus de validation, nous avons appliqué les paquetages issus de la démarche de conception des LEA proposée au chapitre 4. Les patrons utilisés dans le cadre de la

5. http://expressjs.com/
6. https://nodejs.org/
7. https://angularjs.org/
8. http://www.visual-paradigm.com/

démarche de conception des LEA sont regroupés en paquetages correspondants logiquement aux phases de la démarche de conception. Ces paquetages sont les paquetages *Exigences pédagogiques*, *Choix technologiques*, *Evolution*, *Architecture logicielle*, *Acteurs*, *Validation*, *Interface utilisateur*, *Implémentation* et *Déploiement* conformément à la prescription du chapitre 4.

5.1.2.1 Méthode de sélection des patrons

Pour sélectionner un patron dans le répertoire de patrons, lors d'une phase donnée du processus de conception, nous procédons par une réduction littéraire de la liste des exigences fonctionnelles et des exigences spécifiques à la phase donnée pour ne conserver que la phrase clé. Puis en effectuant ensuite une réduction de cette phrase réduite, nous obtenons des mots-clés qui serviront à effectuer une recherche dans le guide de recherche de patrons présenté à la section 3.2.2.

Par exemple l'exigence fonctionnelle libellée ainsi *Consulter le nombre de contributions par personne* peut permettre d'obtenir les mots-clés ou expressions-clés suivants : *Consulter*, *Nombre*, *Contributions*, *Personne*. Nous pouvons aussi utiliser des synonymes de ces termes. Par exemple en lieu et place du mot *Contributions*, nous pouvons utiliser le mot-clé *Message* puisque de façon courante les deux termes sont invariablement utilisés pour désigner des messages postés sur un forum de discussion.

Dans les paragraphes suivants nous détaillons les différentes phases de conception du logiciel d'analyse et d'évaluation de la participation estudiantine de Knowledge Forum 6.

5.1.2.2 Phase d'analyse et de spécification des exigences pédagogiques

Dans cette première phase du processus de conception, nous avons observé la situation d'enseignement et d'apprentissage à travers diverses rencontres avec Mme Laferrière et son équipe. Nous avons ainsi déterminé un scénario pédagogique qui nous a permis d'éliciter les exigences pédagogiques à considérer. Plus précisément, il s'agit des trois exigences suivantes :

- consulter le nombre de contributions par personne ;
- consulter la densité des mots utilisés dans les messages de contribution ;
- consulter la densité des contributions par groupe.

En plus du paquetage *Analyse et spécification des exigences pédagogiques*, nous avons utilisé le paquetage *Acteurs* afin de définir les différents acteurs intervenants dans le processus de conception ainsi que leurs interactions.

Un seul type d'acteur du logiciel d'analyse et d'évaluation de la participation estudiantine a été identifié comme étant l'utilisateur du logiciel : il s'agit de l'acteur *Enseignant*. Toutefois étant dans le cadre d'un processus de validation de notre approche de conception, nous présentons aussi un autre type d'acteur, l'acteur *Concepteur*, intervenant majeur du processus de

conception. L'acteur *Concepteur* représente l'équipe de conception du logiciel, constituée de l'expert en pédagogie, de l'analyste, de l'architecte, de l'expert en interface personne-machine et du développeur logiciel. Ces deux catégories d'acteurs sont autonomes, sont des entités concurrentes et sont proactives, c'est-à-dire que chaque acteur poursuit un but sur le logiciel.

Les membres de l'équipe de conception n'interviennent pas nécessairement tous de façon simultanée lors des différentes phases du processus de conception. Ils seront plus ou moins sollicités, selon leurs compétences et la phase du processus de conception en cours. Par exemple, les analystes seront sollicités durant la phase d'analyse et de spécification du logiciel tandis que les développeurs seront sollicités durant la phase d'implémentation du logiciel.

Les concepteurs concernés par l'activité *Elicitation des exigences pédagogiques* sont l'analyste et l'expert en pédagogie.

Les interactions entre les différents acteurs intervenant sur le logiciel d'analyse et d'évaluation de la participation estudiantine et les différents cas d'utilisation du logiciel sont représentés à la figure 5.1.

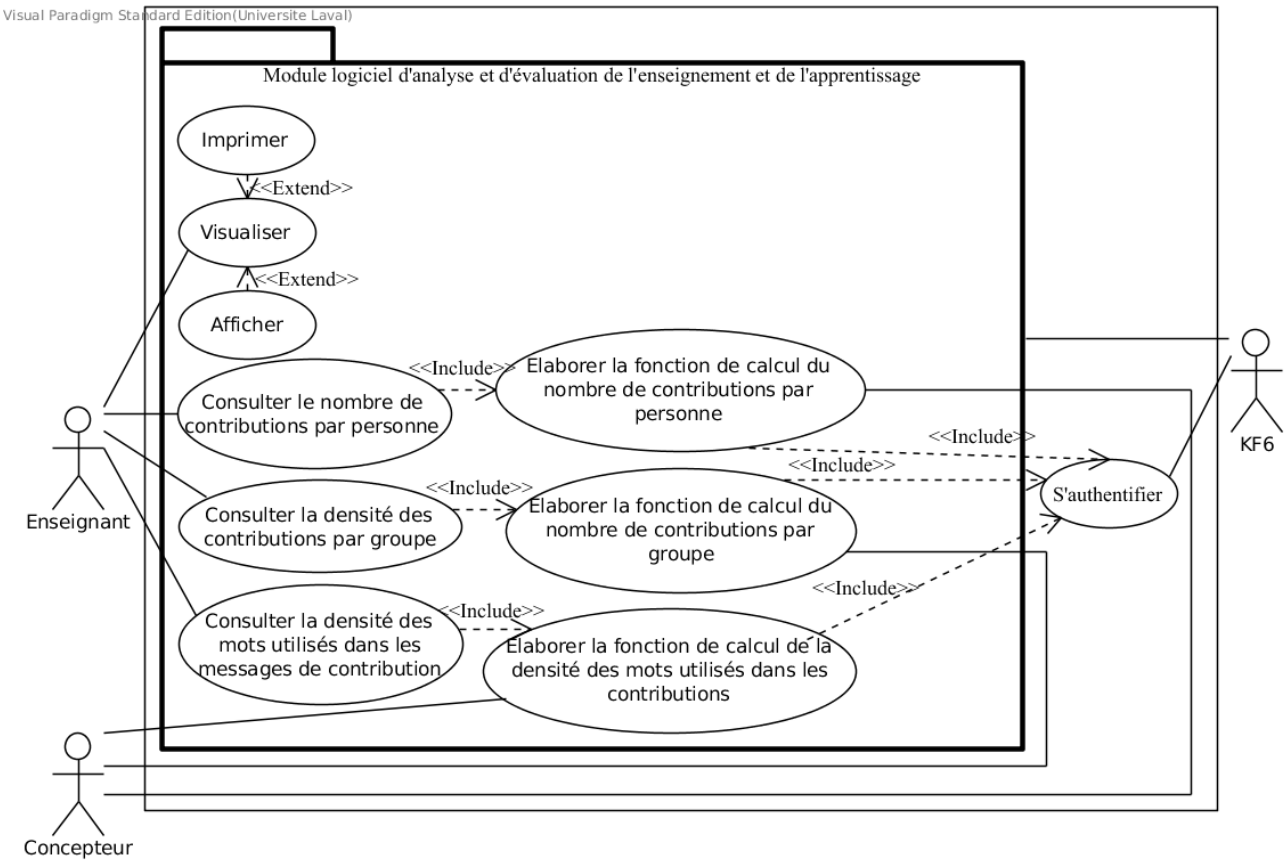


FIGURE 5.1 – Diagramme des cas d'utilisation du logiciel d'analyse et d'évaluation de l'enseignement et de l'apprentissage.

Ce diagramme montre que l'enseignant consulte de manière visuelle, à l'aide de graphiques, des statistiques sur la participation estudiantine dans le forum Knowledge Forum 6. Ses cas d'utilisation du logiciel sont décrits dans le tableau 5.1.

	Activités de consultation des statistiques	**Description**
1	Consulter le nombre de contributions par personne	Fonctions statistiques à élaborer par le concepteur
2	Consulter la densité des contributions par groupe	
3	Consulter la densité des mots utilisés dans les messages de contribution	
4	Visualiser	
5	Imprimer	

TABLE 5.1 – Description des cas d'utilisation de l'acteur *Enseignant*.

Cette analyse de la participation estudiantine peut permettre à l'enseignant de détecter les étudiants en difficultés en vue de leur apporter une rétroaction adéquate, d'évaluer le degré de contribution personnelle et collective des étudiants au processus de création de connaissances sur Knowledge Forum 6, ou encore d'apporter une rétroaction globale sur la qualité de l'enseignement des cours enseignés dans une unité d'enseignement.

Le *Concepteur* élabore les fonctions permettant à l'enseignant d'avoir accès aux informations visuelles dont il a besoin. Ses cas d'utilisation avec le logiciel d'analyse sont décrits dans le tableau 5.2.

Knowledge Forum 6 est le forum auquel doit être intégré le logiciel d'analyse et d'évaluation de la participation estudiantine. L'enseignant s'authentifie sur Knowledge Forum 6 afin de pouvoir effectuer les activités d'analyse des messages des étudiants. La source des données d'où sont extraites les statistiques nécessaires à l'analyse et l'évaluation de la participation estudiantine sur Knowledge Forum est celle de la base de données de Knowledge Forum 6.

Etant donné le fait que nous sommes dans le cadre d'un processus de validation d'une démarche de conception logicielle nous privilègerons la perspective de l'acteur *Concepteur* afin de mettre en évidence les tâches à accomplir par le concepteur.

Le tableau 5.3 montre la relation existant entre les fonctionnalités du logiciel et les patrons sélectionnés. Aux trois exigences fonctionnelles de consultation précédemment définies correspondent les trois tâches d'élaboration du concepteur, soit les tâches *Elaborer la fonction de calcul du nombre de contributions par personne*, *Elaborer la fonction de calcul du nombre de contributions par groupe*, et *Elaborer la fonction de calcul de la densité des mots utilisés dans les contributions*. Aux deux premières tâches, correspond le patron *Evaluation de contenu*, sélectionné de notre répertoire de patrons à l'aide de la méthode décrite à la section 5.1.2.1. A la dernière exigence correspond le patron *Centration* sélectionné suivant le même procédé.

	Tâches du concepteur	Description
1	■ Elaborer la fonction de calcul du nombre de contributions par personne	1. Lire la source de données (base de données de Knowledge Forum 6) par enregistrement. 2. Calculer de façon cumulative le nombre de message postés par un étudiant précisé par l'utilisateur. 3. Afficher le nombre de message postés par étudiant sous la forme d'un graphique : une courbe, histogramme ou autre.
2	■ Elaborer la fonction de calcul du nombre de contributions par groupe d'étudiants	1. Faire les opérations 1 et 2 pécédentes. 2. Afficher le nombre de message postés par les étudiants d'une même équipe de travail sous la forme d'un graphique : une courbe, histogramme ou autre.
3	■ Elaborer la fonction de calcul de la densité des mots utilisés dans les contributions	1. Lire la source de données (base de données de Knowledge Forum 6) par enregistrement. 2. Sélectionner l'ensemble des messages postés sur une période donnée précisée par l'utilisateur. 3. Compter pour un mot donné par l'utilisateur son nombre d'occurrences. 4. Afficher le nombre d'occurrences comptées de ce mot. 5. Afficher en les soulignant dans les messages sélectionnées les occurences du mot dont les occurrences ont été comptées.

TABLE 5.2 – Description des cas d'utilisation de l'acteur *Concepteur*.

Fonctionnalités	Patrons par fonctionnalité	Patrons généraux
Elaborer la fonction de calcul du nombre de contributions par personne	Evaluation de contenu	■ Mode de livraison ■ Validation.Test_exig_pédagogiques
Elaborer la fonction de calcul du nombre de contributions par groupe	Evaluation de contenu	
Elaborer la fonction de calcul de la densité des mots utilisés dans les contributions	Centration	

TABLE 5.3 – Association des patrons recensés dans la phase d'analyse et de spécification des exigences pédagogiques et fonctionnalités correspondantes.

5.1.2.3 Phase de conception de l'architecture logicielle

Dans cette phase, les patrons sélectionnés sont les patrons *MVC* qui provient du paquetage *Architecture* et qui permet de représenter de façon logique et de différencier les différentes entités logiques d'une interface utilisateur, et *Test_ exig_ architecture* qui provient du paquetage *Validation* et qui permet de valider les exigences d'architecture.

Le patron *MVC* décrit un modèle d'organisation en trois parties de l'interface du logiciel que nous concevons. Ce modèle est représenté à la figure 5.2. La première partie, la *Vue*, décrit comment le concepteur doit présenter les données à l'utilisateur. Le cadre de la présentation des données est représenté par des pages web conçues en langage HTML. La troisième partie, le *modèle* manipule les données. Il est représenté par le code qui permet de faire des requêtes à la base de données de Knowledge Forum 6. La deuxième partie, le *Contrôleur*, permet de lier le modèle et la vue.

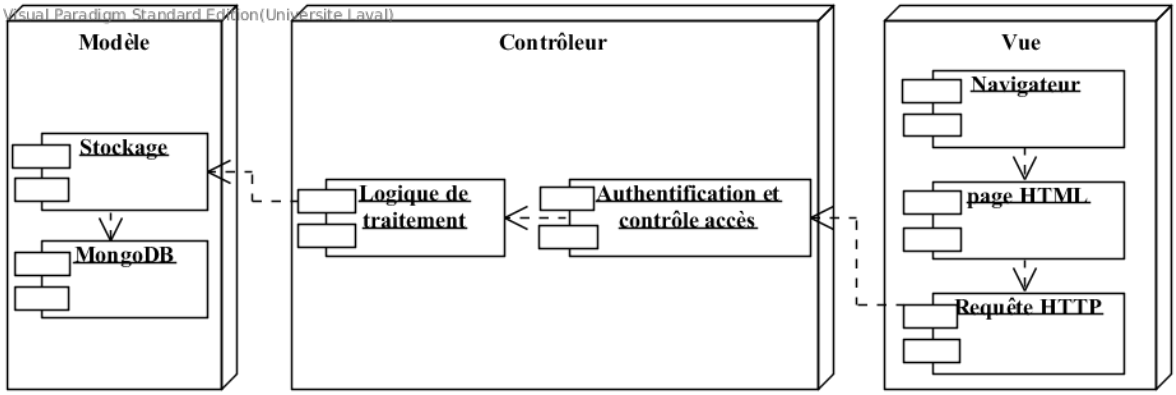


FIGURE 5.2 – Modèle d'architecture de type MVC du logiciel d'analyse et d'évaluation de la participation estudiantine.

Voici ci-dessous un exemple de fonctionnement de l'architecture MVC du logiciel d'analyse de la participation estudiantine. Un utilisateur entre l'adresse du forum Knowledge Forum 6 dans la barre d'adresse de son navigateur qui effectue de ce fait une requête de type HTTP. La requête est transmise au contrôleur concerné qui exécute la logique du logiciel ; il vérifie par exemple si l'utilisateur s'est authentifié sur Knowledge Forum 6. Le contrôleur utilise ensuite le modèle pour obtenir l'accès aux données de la base de données c'est-à-dire les statistiques sur la participation estudiantine sur le forum Knowledge Forum 6. Il effectue les calculs et traitements nécessaires sur les données et il les transmet à une vue par une réponse de type HTTP. La vue récupère ces données et les formate pour les présenter au navigateur de l'utilisaeur.

Deux critères ont guidés le choix du patron *MVC* :

- Le logiciel d'analyse est une application indépendante qui sera intégrée au forum Knowledge Forum 6 en tant qu'une fonctionnalité de celui-ci. De ce fait, nous souhaitons que le code du logiciel d'analyse soit indépendant du reste de l'application pour sa bonne maintenabilité.
- Il doit y avoir une compatibilité parfaite du logiciel d'analyse avec les technologies utilisées pour le développement de Knowledge Forum 6, notamment le langage JavaScript, présentées à la section 5.1.1. Le fait que ces technologies soient orientées-client induit

l'intégration logique du logiciel d'analyse dans la couche *présentation* de Knowledge Forum 6.

5.1.2.4 Phase de conception de l'interface

Dans cette phase, nous avons conçu les interfaces d'accès aux différentes fonctionnalités du logiciel d'analyse de la participation estudiantine.

Nous avons tout d'abord déterminé les exigences des utilisateurs en vue de dresser la liste des exigences spécifiques des interfaces utilisateur du logiciel d'analyse et d'évaluation de la participation estudiantine. Seule l'exigence d'interface *Accès au logiciel à partir de n'importe quel type d'interface* a été retenue.

Nous avons ensuite sélectionné les patrons correspondant à l'exigence retenue soit les patrons *Affichage*, *Systèmes mobiles* et *Systèmes natifs*.

Le patron *Affichage* permet de détecter le type de média utilisé par l'utilisateur et de s'assurer de l'adaptabilité de l'interface en conséquence. Les patrons *Systèmes mobiles* et *Systèmes natifs*, selon le type de terminal détecté par le patron *Affichage* décrivent les spécifications d'affichage respectivement pour les terminaux mobiles ou les terminaux natifs.

La figure 5.3 montre l'interface d'accueil du logiciel d'analyse et d'évaluation de la participation estudiantine et les liens d'accès aux trois fonctionnalités *Consulter le nombre de contributions par personne*, *Consulter le nombre de contributions par groupe* et *Consulter la densité des mots utilisés dans les messages de contribution* du forum de discussion tandis que la figure 5.4 illustre une page web affichant un histogramme représentant le nombre de messages postés par étudiant sur une base journalière durant une certaine période.

FIGURE 5.3 – Capture d'écran de la page d'accueil du logiciel d'analyse et d'évaluation de la participation estudiantine.

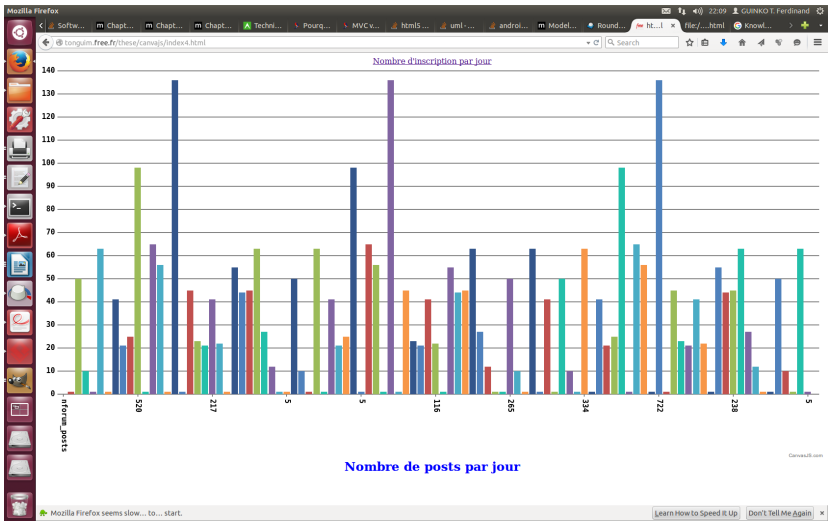


FIGURE 5.4 – Page web affichant un histogramme représentant le nombre de posts par jour généré à partir de l'interface du module d'analyse de l'enseignement et de l'apprentissage.

Tous les patrons liés à la conception de l'interface font partie du paquetage *Interface* à l'exception du patron *Test_ exig_ utilisateur* qui provient du paquetage *Validation*.

5.1.2.5 Phase des choix technologiques

Dans cette phase nous avons d'abord sélectionné les patrons dont nous avions besoin soit les patrons *Elaboration des règles des choix technologiques* décrivant le processus d'élaboration des règles des choix technologiques, *Normes locales* décrivant la liste des normes locales (préférence pour une marque ou un fournisseur, ...) dont il faut tenir compte pour effectuer les choix technologiques et *Aspects sécuritaires* décrivant les aspects sécuritaires des choix technologiques à effectuer. Nous avons ensuite vérifié sur le Web l'existence éventuelle de solutions logicielles déjà existantes et compatibles avec Knowledge Forum 6. Cette recherche nous a permis de découvrir la librairie de conception de graphiques en langage JavaScript, CanvaJs [9] compatible avec les spécifications de notre projet.

Quand à la source de données, le Système de Gestion de Bases de données (SGBD) qui sera utilisé est celui de Knowledge Forum 6 qui contient les statistiques devant êtres lues et affichées à l'écran à l'aide de graphiques. Knowledge Forum 6 utilise le SGBD MongoDB.

5.1.2.6 Phase d'implémentation

Dans cette phase nous avons sélectionné les patrons suivants : *Modularité*, *Portabilité*, *Compatibilité avec le Web* et *Validation.Test_ exig_ utilisateur*. Ces patrons liés à l'implémentation font naturellement partie du paquetage *Implémentation* à l'exception du patron *Test_ exig_ utilisateur* qui provient du paquetage *Validation*.

9. CanvaJS 1.7.0 : `http://canvasjs.com/`

Nous avons ensuite écrit le code dont on peut voir un aperçu pour la fonction de calcul du nombre de messages postés par étudiant à la figure 5.5, puis nous l'avons vérifié et testé à l'aide du patron *Validation.Test_ exig_ utilisateur*, qui permet de décrire le processus de conformité des exigences des utilisateurs avec le code écrit.

```
18      function processData( allText )
19      {
20         var allLinesArray = allText.split("\n");
21         if( allLinesArray.length > 0 )
22         {
23            var dataPoints = [];
24            for (var i = 0; i <= allLinesArray.length-1; i++)
25            {
26               var rowData = allLinesArray[i].split(",");
27               dataPoints.push({ label:rowData[11], y:parseInt(rowData[2]) });
28            }
29            drawChart(dataPoints);
30         }
31      }
```

FIGURE 5.5 – Vue partielle du code implémentant la fonction de calcul du nombre de messages postés par étudiant.

Le logiciel développé permet à l'enseignant de consulter la participation estudiantine relative au forum de discussion de son cours afin de pouvoir s'il y a lieu orienter la démarche de coélaboration de connaissances. Seule la fonctionnalité *Consulter le nombre de contributions par étudiant* du logiciel d'analyse a été développée ; les deux autres ne l'ont pas été car nous n'avons pu obtenir une source de données sur laquelle implémenter ces fonctionnalités.

5.1.2.7 Phase de validation

Dans cette phase nous vérifions que le logiciel d'analyse et d'évaluation de la participation estudiantine répond bien aux exigences initiales. Les résultats des tests de validation sont présentés dans le tableau 5.4. Les exigences élicitées à la section 5.1.1.3 ont été testées l'une après l'autre. Pour chacune des exigences testées la procédure de test, le résultat attendu, le résultat obtenu et l'appréciation finale sont précisées. Faute d'avoir obtenu les données de la base de données de Knowledge Forum 6, nous n'avons pu répondre qu'à l'exigence *Consulter le nombre de contributions par étudiant* ; c'est par conséquent la seule fonctionnalité du logiciel qui a été implémentée.

Source de données

Nous n'avons pu avoir accès aux données réelles de la base de données de Knowledge Forum 6 pour des raisons de contraintes administratives et de délais. En conséquence, l'ensemble de données utilisé pour la validation du modèle conceptuel de cette étude de cas est la base de données *MITx and HarvardX Dataverse* (MITx & HarvardX, 2014) disponible gratuitement sur le Web. Cette base de données contient l'ensemble des données collectées pendant l'année académique 2012-2013 sur une plateforme logicielle d'enseignement et d'apprentissage détenue

Exigences	Procédure de test	Résultat attendu	Résultat obtenu	Appréciation
Consulter le nombre de contributions par étudiant	Test des interfaces	Graphique représentant le nombre de contributions par étudiant	Conforme au résultat attendu	Réalisé
Consulter la densité des contributions par groupe	Test des interfaces	Graphique représentant la densité des contributions par groupe d'étudiants	-	Non réalisé
Consulter la densité des mots utilisés dans les messages de contribution	Test des interfaces	Graphique représentant la densité des mots utilisés dans les messages de contribution	-	Non réalisé

TABLE 5.4 – Résultat de la validation du logiciel d'analyse et d'évaluation de la participation estudiantine.

conjointement par les universités Harvard et le MIT. Les données collectées concernent les informations sur les étudiants utilisant la plateforme, les cours accessibles sur la plateforme, la fréquence d'envois de messages, de lecture des modules de cours etc., et sont stockées dans un fichier au format *.csv* contenant 641139 enregistrements. Une vue partielle de cet ensemble de données est illustrée à la figure 5.6. Pour des raisons de lisibilité, seuls 11 des 17 champs du fichier *.csv* apparaissent sur cette figure.

X94

	A	B	C	D	E	F	G	H	I	J	K
1	course_id	userid_DI	registered	viewed	explored	certified	final_cc_cname_DI	LoE_DI	YoB	gender	grad
2	HarvardX/CB22x/2013_Spring	MHxPC130442623	1	0	0	0	United States	NA	NA	NA	
3	HarvardX/CS50x/2012	MHxPC130442623	1	1	0	0	United States	NA	NA	NA	
4	HarvardX/CB22x/2013_Spring	MHxPC130275857	1	0	0	0	United States	NA	NA	NA	
5	HarvardX/CS50x/2012	MHxPC130275857	1	0	0	0	United States	NA	NA	NA	
6	HarvardX/ER22x/2013_Spring	MHxPC130275857	1	0	0	0	United States	NA	NA	NA	

FIGURE 5.6 – Vue partielle des données de l'ensemble de données *MITx and HarvardX Dataverse.*

Par exemple, les champs extraits de cette source de données pour le calcul du nombre de contributions par étudiant sont les champs *userid_ DI*, stocké à la colonne 2, stockant l'identifiant d'un étudiant à chacune de ses connexions sur la plateforme logicielle, et *nforum_ posts* stocké à la colonne 11, contenant le nombre de contributions fait par un étudiant à chacune de ses connexions sur la plateforme logicielle.

La fonctionnalité *Consulter la densité des mots utilisés dans les messages de contribution* se base sur le calcul de l'occurrence d'un mot donné. La base de données *MITx and HarvardX Dataverse* ne stockant aucune information relative aux mots utilisés dans le forum de discussion, la fonctionnalité *Consulter la densité des mots utilisés dans les messages de contribution* n'a pu être implémentée.

5.1.2.8 Phase de déploiement

Cette phase de déploiement permet de décrire le processus d'accueil du logiciel développé dans son environnemt d'exploitation prévu. Nous avons d'abord recherché les patrons pertinents pour cette phase.

Nous avons ensuite sélectionné le patron *Configuration* pour aider à définir l'ensemble des configurations à effectuer en vue du déploiement du logiciel d'analyse. Le diagramme de déploiement du logiciel d'analyse et d'évaluation de la participation estudiantine est représenté à la figure 5.7.

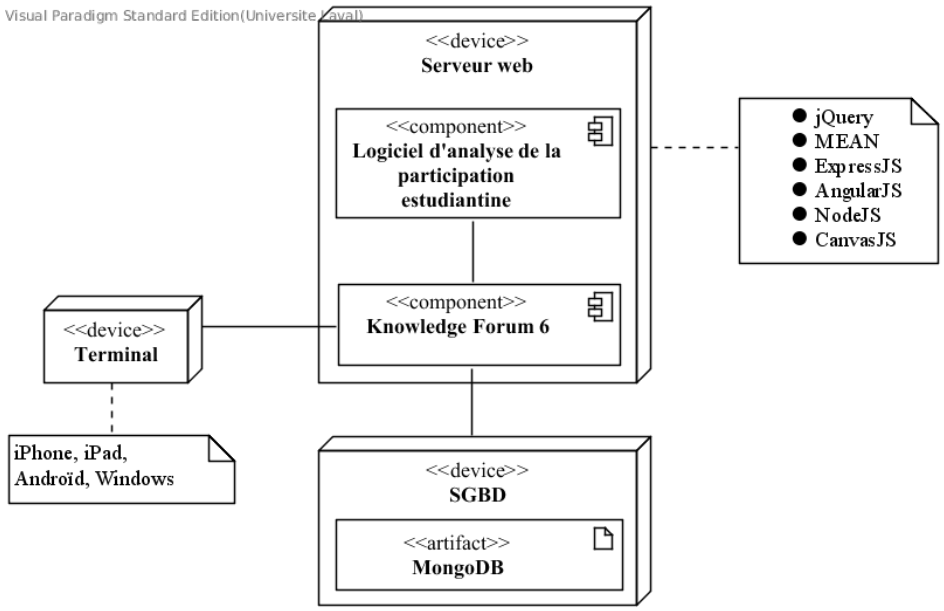


FIGURE 5.7 – Diagramme de déploiement du logiciel d'analyse et d'évaluation de la participation estudiantine.

Sur cette figure, les formes rectangulaires portant l'étiquette *«device»* désignent les ressources physiques computationnelles dotées d'une capacité de traitement et d'exécution de programmes, tel que les serveurs web, de données ou les terminaux comme les ordinateurs de travail, les téléphones mobiles, etc. Les formes rectangulaires portant l'étiquette *«component»* désignent un sous système ou un module doté d'une interface de communication avec le reste du système. Le SGBD gère toutes les données stockées du système déployé.

Dans le cadre du cas présent, nous n'avons pu implémenter qu'une seule des trois fonctionnalités élaborées dans la phase de conception de l'architecture logicielle. Cette expérience de conception s'avère toutefois suffisante pour nous permettre de constater que le modèle d'architecture du LEA proposé peut bien être developpé en suivant les phases proposées dans notre démarche de conception d'un LEA, et que les patrons utilisés à cette fin sont valides et fonctionnels.

5.2 Etude de cas 2 : conception d'un forum de discussion pédagogique

Cette section décrit l'étude de cas 2 : il s'agit de la conception d'un forum de discussion à finalité pédagogique.

5.2.1 Contexte

Il s'agit ici d'un projet de conception d'un LEA s'inscrivant dans le cadre du Programme d'appui au développement de cours hybrides, à l'Université Laval. Ce programme est une initiative du Vice-Rectorat aux Etudes et aux Activités Internationales de l'Université Laval à travers laquelle l'Université souhaite encourager et appuyer les efforts de son personnel enseignant par la création d'une offre de formation adaptée aux nouvelles réalités de sa population étudiante (Université Laval, 2015).

Rappelons qu'en effet *l'Université Laval a inscrit en 2005 la formation à distance parmi ses priorités* (Université Laval, 2012). Partant du constat que les réalités socio-économiques et académiques de ses étudiants correspondent à une grande variabilité de scénarios pédagogiques elle a initié le programme d'appui au développement de cours hybrides qui permettra d'offrir aux étudiants concernés des formules de scénarios pédagogiques flexibles et à mi-chemin entre les formations en présentiel classiques et celles à distance.

Le présent processus de conception d'un logiciel d'enseignement et d'apprentissage est une réponse qui s'inscrit dans le cadre de ce projet.

5.2.1.1 Besoin

Le Programme d'appui au développement de cours hybrides, à l'Université Laval, a besoin d'un logiciel d'enseignement et d'apprentissage offrant une formule d'enseignement et d'apprentissage hybride entre l'enseignement en présenciel et l'enseignement à distance et permettant une interaction entre l'apprenant et l'enseignant.

5.2.1.2 Résultats attendus

Le logiciel doit permettre :

- une interaction, qu'elle soit réactive ou proactive entre l'enseignant et l'étudiant ;
- une interaction entre les étudiants afin qu'ils puissent initier des discussions ou discuter d'une question proposée par l'enseignant ;
- à la fois un enseignement individualisé ou collectif : l'enseignant peut proposer des activités pédagogiques à un seul étudiant ou à un groupe d'étudiants ;

- aux étudiants de poser des questions sur le contenu du cours et les travaux pendant toute la durée du cours ;
- à l'étudiant de sentir qu'il fait partie d'un groupe et de poser des questions pendant tout le long du cours dès qu'il en ressent le besoin, notamment dans le cas d'un enseignement à distance notamment.

5.2.2 Approche de conception

Dans cette section, nous décrivons le processus de conception du LEA.

5.2.2.1 Méthode de sélection des patrons

Tout au long du processus de conception du LEA, pour sélectionner un patron dans le répertoire de patrons nous utilisons la méthode de recherche et de sélection de patrons décrite à la section 3.2.2.

Comme dans le cadre du cas d'étude présenté à la section 5.1, nous avons appliqué au présent contexte la démarche de conception des LEA et ses huit phases proposée au chapitre 4. Nous détaillons ce processus en suivant les différentes phases de conception du LEA.

5.2.2.2 Phase d'analyse et de spécification des exigences pédagogiques

Il existe différentes possibilités de LEA pouvant répondre au besoin du Programme d'appui au développement de cours hybrides, à l'Université Laval. Il faut donc déterminer quel type de LEA correspond au présent contexte et sélectionner les patrons appropriés.

A l'aide de la la méthode de recherche et de sélection de patrons décrite à la section 5.1.2.1, et du mot-clé *interaction* nous avons sélectionné l'ensemble de patrons nommé *Cours interactif* du paquetage *Michael Derntl*, c'est-à-dire le patron *Michael Derntl.Cours interactif*.

C'est donc une plateforme interactive d'enseignement et d'apprentissage que nous concevrons comme LEA répondant au besoin du Programme d'appui au développement de cours hybrides de l'Université Laval. Dans notre répertoire de patrons, plusieurs patrons permettent de décrire différents modules pouvant être intégré à la plateforme interactive d'enseignement et d'apprentissage. Parmi ces possibilités nous avons choisi le patron *Michael_Derntl.element_interactif.discussion en ligne*. Nous intègrerons donc un forum de discussion en ligne à notre plateforme interactive. D'autres modules comme celui du *chat* pourront être intégrés ultérieurement à la plateforme interactive d'enseignement et d'apprentissage s'il y a lieu.

Dans le cadre de la conception de ce forum de discussion, nous avons retenu le scénario d'encadrement conformément à la recommandation formulée par le guide des bonnes pratiques enseignantes du Bureau des Services Pédagogiques de l'Université Laval (Université Laval,

2013) qui préconise le scénario d'encadrement dans le cas où l'enseignant, sur le forum de discussion, doit offrir la possibilité aux étudiants de poser des questions sur le contenu du cours ou les travaux.

Etant donné le scénario pédagogique d'encadrement, et compte tenu du fait que nous sommes dans le cadre d'un processus de validation d'une démarche de conception logicielle nous présenterons les différents cas d'utilisation du logiciel en considérant la perspective de l'acteur *Concepteur* afin de mettre en évidence les tâches à accomplir par le concepteur.

Le *Concepteur*, seul acteur décrit sur le diagramme, représente les différentes personnes et compétences intervenant dans le processus de conception du forum de disussion, c'est-à-dire l'expert en pédagogie, l'analyste, l'architecte, l'expert en interface personne-machine, et le développeur logiciel.

Ce processus de conception se base sur une approche par fonctions. Une telle approche permet de simplifier le processus de conception, mais aussi la maintenance du logiciel en développement. Les trois fonctions sur lesquelles reposent l'approche sont les fonctions *Gestion du curriculum*, *Gestion des évaluations* et *Gestion de l'interaction*.

L'acteur *Concepteur* élabore ces trois fonctionnalités représentées schématiquement sur les diagrammes des cas d'utilisation illustrés à la figure 5.8.

Les éléments conceptuels des cas d'utilisation sont graphiquement représentés à l'aide de l'outil de modélisation `Visual paradigm`. La version de l'outil utilisée dans le cadre de ce processus de validation est la version 12.1 et la licence d'utilisation est la licence académique standard.

L'association entre les patrons recensés dans cette phase d'analyse et de spécification des exigences pédagogiques et les trois fonctions mises en évidence au niveau du diagramme des cas d'utilisation à la figure 5.8 est exprimée sur le tableau 5.5. Ce tableau fournit, pour chaque fonctionnalité mise en évidence, la liste des patrons associés. Par exemple le patron *MichaelDerntl.Cours* permet spécifiquement de décrire la fonctionnalité *Gestion du curriculum*. D'autres patrons sélectionnés tel que le patron *Evaluation de contenu* s'appliquent de façon transversale à l'ensemble des trois fonctionnalités ; ces patrons sont classés dans la colonne nommée *Patrons généraux*.

5.2.2.3 Phase de conception de l'architecture logicielle

Pour cette phase de conception de l'architecture logicielle, nous avons sélectionné de notre répertoire les patrons ci-après nommés :

- *Architecture 3 tiers :* il permet d'une part d'effectuer une séparation des processus de conception et d'implémentation en trois parties que sont le *Stockage des données*, la *Logique applicative*, et la *Présentation*. D'autre part il permet aussi une plus grande évolutivité de

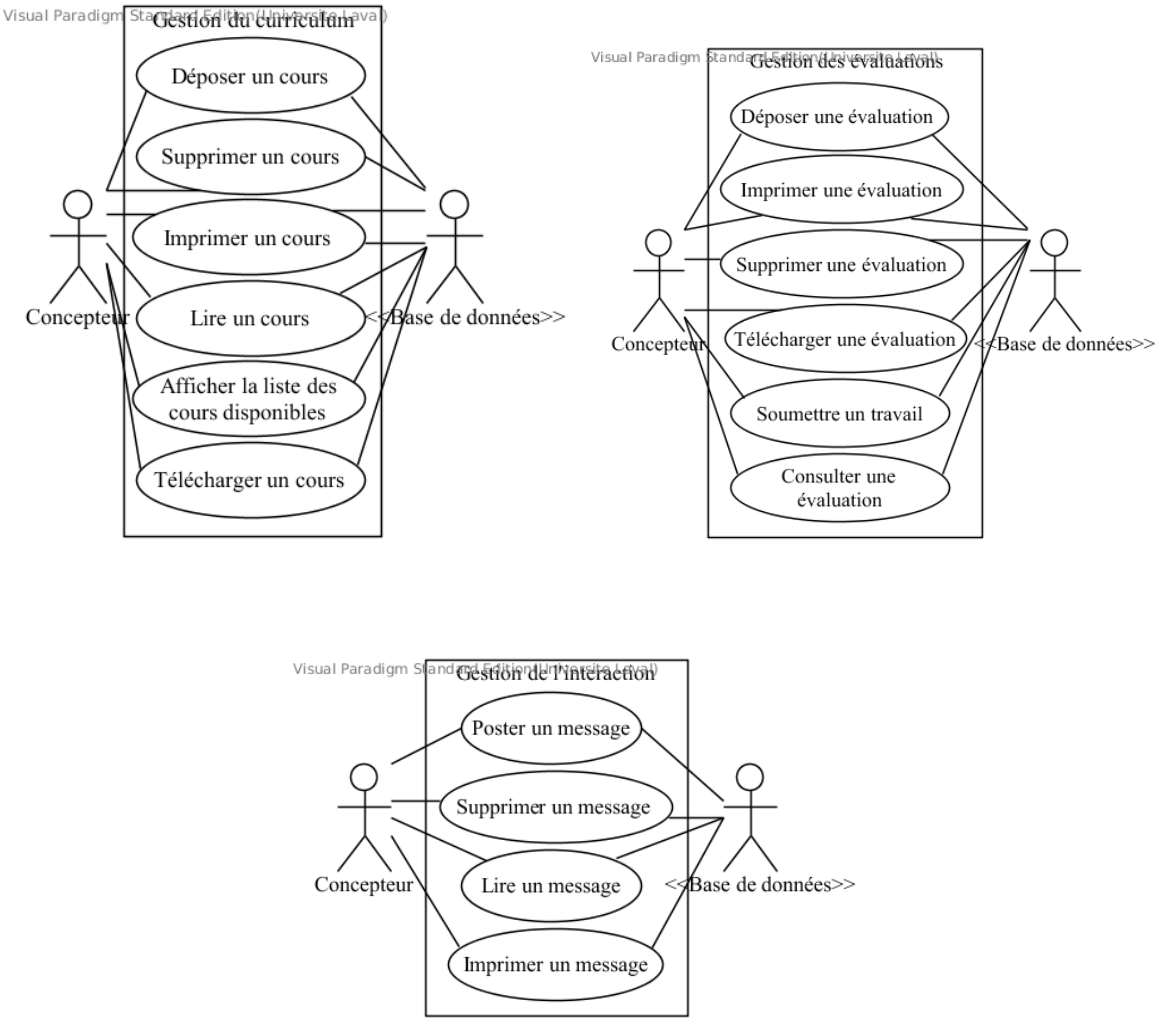


FIGURE 5.8 – Scénario d'encadrement de la plateforme d'interaction d'enseignement et d'apprentissage : diagrammes des cas d'utilisation

Fonctionnalités	**Patrons par fonctionnalité**	**Patrons généraux**
Gestion du curriculum	MichaelDerntl.Cours	Evaluation de contenu Centration Mode de livraison Validation.Test_exig_pédagogiques
Gestion des évaluations	MichaelDerntl.auto-examen, MichaelDerntl.examen-sujet de l'enseignant	
Gestion de l'interaction	Echange des contributions et Discussions en ligne	

TABLE 5.5 – Association des patrons recensés dans la phase d'analyse et de spécification des exigences pédagogiques et fonctionnalités correspondantes.

la plateforme interactive d'enseignement. L'architecture représentée à la figure 5.9 a les trois tiers que sont les tiers *Données* basé sur le Système de Gestion de Bases de Données MySQL, *Logique métier* où sont effectués tous les traitements applicatifs du forum de discussion et *Service web* qui offre à l'utilisateur les interfaces d'interaction avec le forum. Le tiers service web assure les fonctions de présentation et est constitué des parties visibles et interactives du forum de discussion, c'est-à-dire des interfaces utilisateur. L'utilisateur saisit l'adresse de la page d'accueil du forum de discussion dans la barre d'adresse du navigateur web. Celui-ci transmet l'adresse sous la forme d'une requête de type HTTP au tiers applicatif chargé de la logique métier. Ce tiers s'assure des fonctions applicatives telle que la validation des données entrées par l'utilisateur et la modélisation des processus métiers comme les requêtes formulées par l'utilisateur. Le serveur web analyse donc la requête HTTP reçue puis transmet au serveur PHP la sous-requête qui le concerne et au tiers *Données* la sous-requête qui le concerne sous la forme d'une requête de type SQL. Les données requises sont renvoyées au tiers de la *Logique métiers* qui à l'aide des patrons *Modèle d'intégration sémantique* et *Modèle d'intégration de l'uniformité* renvoie au navigateur web grâce à une réponse de type HTTP la page web d'accueil du forum de discussion.

- *MVC :* les différentes couches de ce patron ont été présentées à la section 5.1.2.3. Il permet de simplifier l'organisation et l'implémentation du tiers *Présentation* de l'architecture 3 tiers.

- *Performance :* il sert à décrire les compromis fonctionnalités/vitesse d'exécution à réaliser.

- *Modèle d'intégration sémantique :* il est utile pour s'assurer que les données échangées entre le forum de discussion, la plateforme interactive et le reste de l'environnement numérique de hérarchie supérieure ont la même signification.

- *Validation.Test_ exig_ architecture :* il sert à effectuer les tests validation de l'architecture.

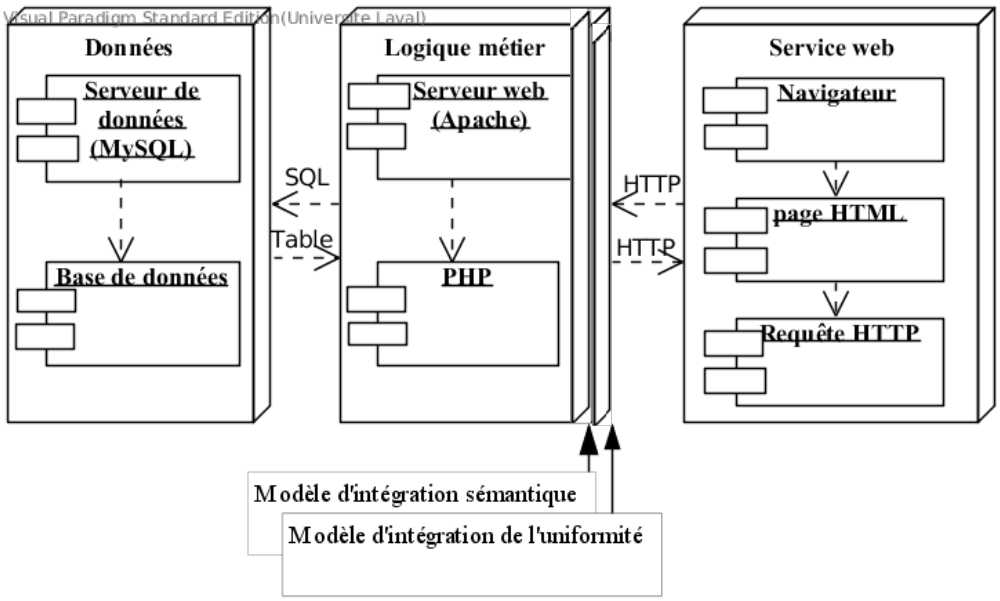


FIGURE 5.9 – Représentation de l'architecture trois tiers du forum de discussion d'encadrement pédagogique.

5.2.2.4 Phase de conception de l'interface utilisateur

Dans cette phase nous avons conçu les interfaces d'accès aux différentes fonctionnalités du forum de discussion, présentées aux figures 5.10, 5.11, 5.12, et 5.13. Nous avons d'abord procédé à la détermination des utilisateurs et ensuite à la sélection des patrons pertinents dans le répertoire des patrons.

Les patrons séléctionnés sont les suivants :

- *Personnalisation :* ce patron est utilisé pour décrire comment l'étudiant peut personnaliser son interface selon ses préférences. Cela permettra d'améliorer l'expérience en matière d'apprentissage de l'utilisateur.
- *Affichage :* il permet de décrire comment doivent être capturées et traitées les données au niveau du forum.
- *Systèmes mobiles :* il permet de décrire comment doit être réalisé l'affichage des pages du forum sur un appareil mobile, par exemple un teléphone intelligent.
- *Systèmes natifs :* il permet de décrire comment doit être réalisé l'affichage des pages du forum sur un ordinateur traditionnel tel un poste de travail ou un ordinateur portable.
- *Test_ exig_ utilisateur :* il permet de décrire comment doivent être effectués les tests des différentes interfaces du forum.

Les patrons liés à la conception des interfaces font partie du paquetage *Interface*.

La figure 5.10 montre l'interface d'accueil du forum de discussion d'encadrement pédagogique et les liens d'accès aux trois fonctionnalités *Curriculum*, *Evaluations*, et *Messages* du forum de discussion.

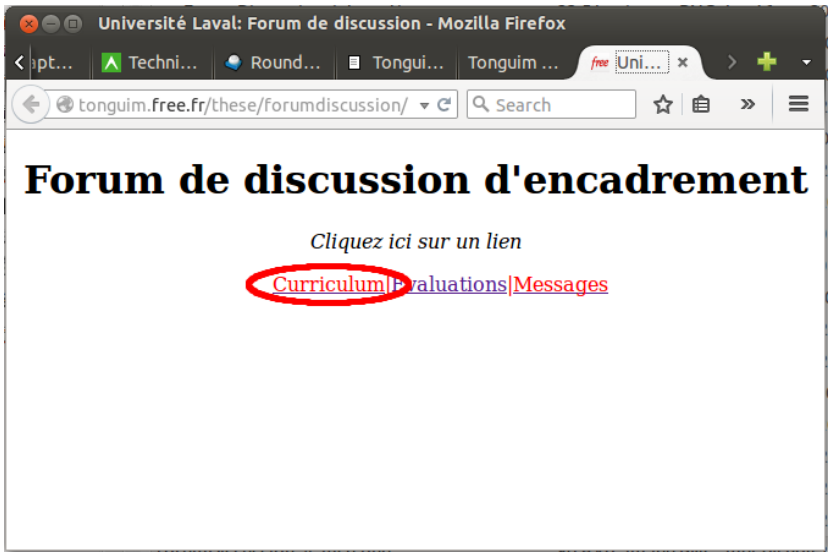


FIGURE 5.10 – Interface d'accueil du forum de discussion d'encadrement pédagogique.

La figure 5.11 montre l'interface d'accès aux différentes sous-fonctionnalités de la fonction *Curriculum* du forum de discussion d'encadrement.

FIGURE 5.11 – Interface d'accès aux fonctionalités de gestion du forum de discussion d'encadrement pédagogique.

La figure 5.12 montre l'interface permettant d'accéder à la fonction de téléversement d'un document de cours sur le forum.

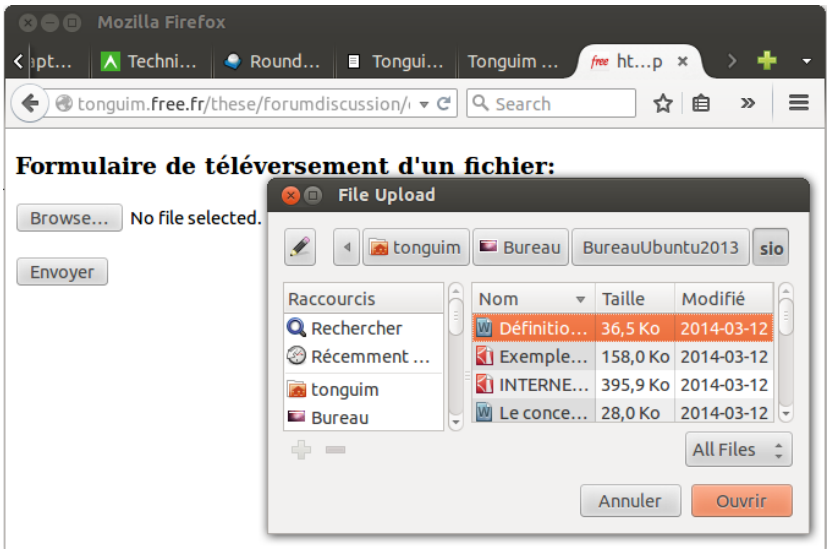


FIGURE 5.12 – Interface de téléversement d'un cours sur le forum de discussion d'encadrement pédagogique.

La figure 5.13 montre l'interface affichant la liste des cours disponibles sur la plateforme interactive.

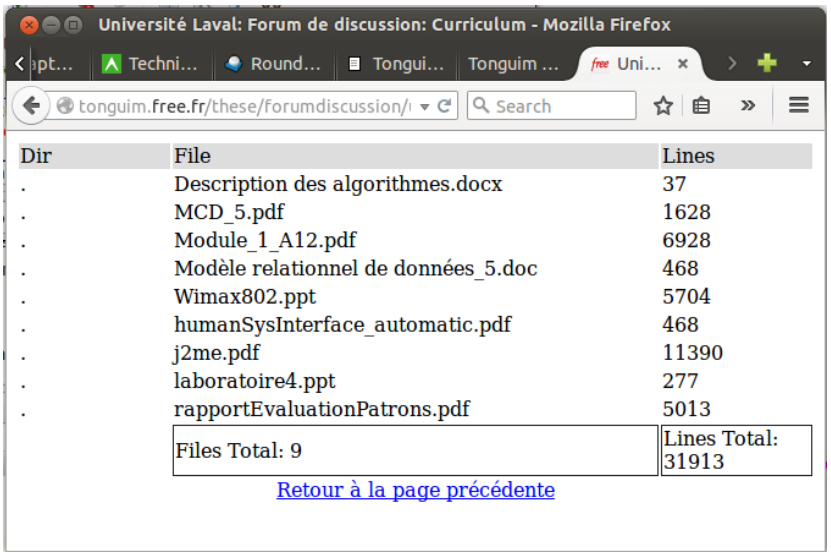


FIGURE 5.13 – Liste des cours disponibles sur le forum de discussion d'encadrement pédagogique.

5.2.2.5 Phase des choix technologiques

Dans cette phase nous avons d'abord sélectionné les patrons pertinents dans le répertoire des patrons, c'est-à-dire les patrons *Elaboration des règles des choix technologiques* décrivant le processus d'élaboration des règles des choix technologiques, *Normes locales*, décrivant la liste des normes locales (préférence pour une marque ou un fournisseur par exemple) dont il

FIGURE 5.14 – Aperçu du code écrit en langage PHP.

faut tenir compte pour effectuer les choix technologiques et *Aspects sécuritaires* décrivant les aspects sécuritaires des choix technologiques à effectuer. Nous avons ensuite vérifié l'existence de solutions similaires. La recherche de solutions similaires existantes n'a pas permis de trouver une solution répondant aux exigences de l'architecture logicielle.

Nous avons donc procédé au développement de la plateforme d'interaction en utilisant le langage HTML comme langage de conception des pages web, le langage PHP comme langage de script permettant de faire l'interface entre les pages web et la base de données et comme Système de Gestion de Bases de données, MySQL.

Les patrons liés aux choix technologiques font partie du paquetage *Choix technologiques*.

5.2.2.6 Phase d'implémentation

Dans cette phase nous avons procédé à la sélection des patrons pertinents que sont les patrons *Portabilité*, *Implémenter les spécifications de l'architecture logicielle*, *Implémenter les spécifications de l'interface*, *Test_ exig_ utilisateur*. Les patrons liés à l'implémentation font partie du paquetage *Implémentation* à l'exception du patron *Test_ exig_ utilisateur*.

Nous avons ensuite développé le forum de discussion. Sur la figure 5.14 apparaît un aperçu du code en langage PHP de la page d'affichage de la liste des cours disponibles.

Les différentes interfaces réalisées et leur enchainement sont représentés à la figure 5.15.

Selon ses trois fonctionnalités définies plus haut, le forum développé permet les tâches suivantes :

- *Curriculum :* dépôt par un enseignant de son matériel de cours, affichage de la liste des cours disponibles.
- *Evaluations :* soumission des travaux des étudiants.

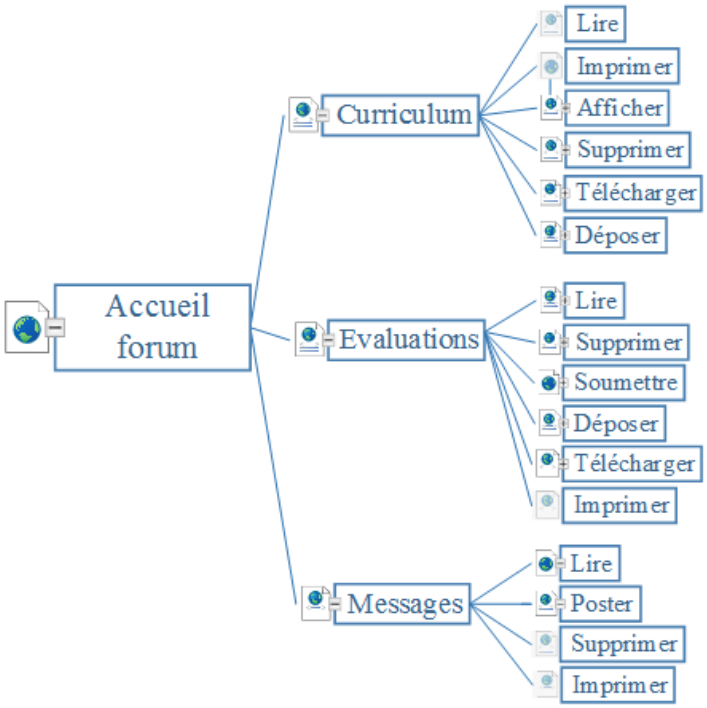


FIGURE 5.15 – Carte du forum de discussion.

- *Messages :* envoi et lecture de messages.

5.2.2.7 Phase de validation

Dans cette phase nous vérifions que la plateforme d'enseignement et d'apprentissage répond bien aux exigences initiales. Les résultats des tests de validation sont présentés dans le tableau 5.6. Les exigences élicitées à la section 5.2.1.2 ont été testées l'une après l'autre. Pour chacune des exigences testées la procédure de test, le résultat attendu, le résultat obtenu et l'appréciation finale sont précisées. Nous avons pu répondre à toutes les exigences à l'exception des exigences *Permettre une interaction entre l'apprenant et l'enseignant* et *Permettre à l'étudiant de sentir qu'il fait partie d'un groupe et de poser des questions pendant tout le long du cours dès qu'il en ressent le besoin* auxquelles nous n'avons pu satisfaire que partiellement par manque de temps.

La plateforme d'enseignement et d'apprentissage développée répond bien à ces exigences.

Exigences	Procédure de test	Résultat attendu	Résultat obtenu	Appréciation
Offrir une formule d'enseignement et d'apprentissage hybride entre l'enseignement en présenciel et l'enseignement à distance	Test des interfaces. Test des fonctionnalités *Gestion de l'interaction, Gestion du curriculum, Gestion des évaluations*	Publier le matériel du cours. Déposer une évaluation. Poursuivre la discussion du cours sur le forum.	Conforme au résultat attendu	Réalisé
Permettre une interaction entre l'apprenant et l'enseignant	Test de la fonction *Gestion de l'interaction* Envoi de messages privés.	Déposer une évaluation. Soumettre un travail.	Poster un message. Lire un message.	Partiellement réalisé
Permettre une interaction entre les étudiants afin qu'ils puissent initier des discussions ou discuter d'une question proposée par l'enseignant	Test de la fonction *Gestion de l'interaction*	Poster un message. Lire un message.	Conforme au résultat attendu	Réalisé
Permettre à la fois un enseignement individualisé ou collectif	Test des fonctionnalités *Gestion de l'interaction, Gestion du curriculum, Gestion des évaluations*	Publier le matériel du cours. Déposer une évaluation. Poursuivre la discussion du cours sur le forum.	Conforme au résultat attendu	Réalisé
Permettre aux étudiants de poser des questions sur le contenu du cours et les travaux pendant toute la durée du cours	Test de la fonction *Gestion de l'interaction*	Poster un message. Lire un message.	Conforme au résultat attendu	Réalisé
Permettre à l'étudiant de sentir qu'il fait partie d'un groupe et de poser des questions pendant tout le long du cours dès qu'il en ressent le besoin	Test de la fonction *Gestion de l'interaction*	Personnalisation de l'interface. Poster un message. Lire un message.	Conforme au résultat attendu	Partiellement réalisé

TABLE 5.6 – Résultat de la validation du forum de discussion.

5.2.2.8 Phase de déploiement

Dans le cadre de la phase de déploiement, deux activités ont été réalisées : il s'agit de *Sélectionner les patrons pertinents dans le répertoire des patrons* et *Description des configurations à effectuer.*

Le diagramme de déploiement du logiciel d'analyse et d'évaluation de la participation estudiantine est représenté à la figure 5.16.

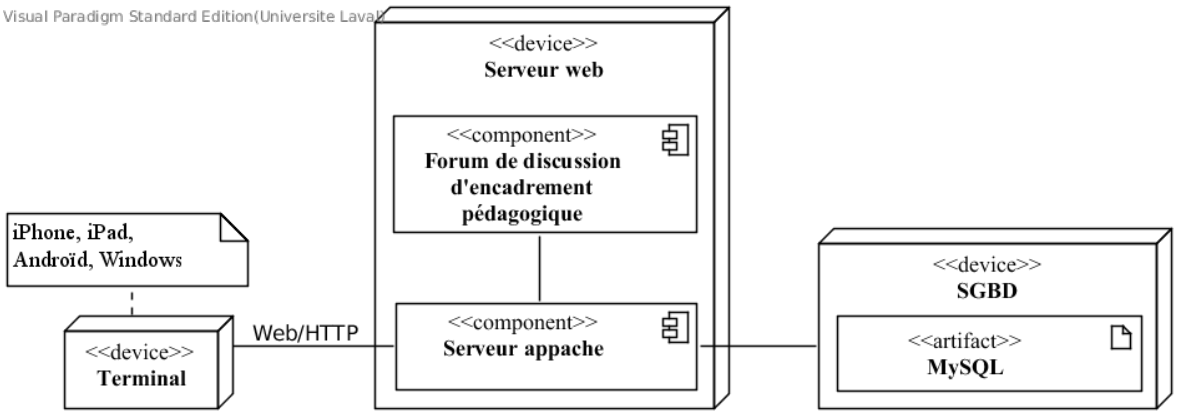


FIGURE 5.16 – Diagramme de déploiement du forum de discussion d'encadrement pédagogique.

Dans le cadre du cas 2, quatre sur six fonctionnalités élaborées dans la phase de conception de l'architecture logicielle ont été entièrement implémentées. Les deux autres l'ont été partiellement faute de temps. Cette deuxième expérience de conception nous a permis de constater une fois de plus que le modèle d'architecture du LEA proposé a pu bien être developpé en suivant les phases proposées dans notre démarche de conception d'un LEA, et que les patrons utilisés à cette fin sont valides et fonctionnels.

5.3 Conclusion

L'objectif majeur de ce chapitre était de présenter un cadre de validation de la démarche de conception des logiciels d'enseignement et d'apprentissage basée sur les patrons présentée au chapitre 4. Cet objectif se déclinait en les sous objectifs suivants : d'abord évaluer le réalisme et la faisabilité de l'approche de conception proposée, puis ensuite vérifier que les patrons proposés sont valides et fonctionnels.

Deux exemples ont servi de cadre d'application de la démarche de conception des LEA et d'utilisation des patrons proposés. Il s'agit premièrement de la conception d'un logiciel d'analyse et d'évaluation de la participation estudiantine sur le forum de discussion Knowledge Forum 6, et deuxièmement de la conception d'un forum de discution d'enseigement et d'apprentissage.

Ce processus de validation nous a permis d'observer que la simplicité de notre démarche de conception des LEA, basée sur les patrons, ouvre de nouvelles perspectives pour les experts et toutes les compétences impliqués dans le développement des LEA. D'une part la démarche basée sur les patrons permet aux experts du génie logiciel peu expérimentés en matière de développement des LEA d'apprendre de l'expérience de leurs pairs, d'éviter des erreurs dues éventuellement à leur manque d'expérience et aussi de gagner du temps en performance. D'autre part la simplicité de la démarche permet à des experts dont les compétences sont étrangères au domaine du génie logiciel, par exemple les experts en pédagogie, de pouvoir intervenir sur des aspects du processus de conception plus techniques et liés au domaine du

génie logiciel. Le guide de recherche de patrons se montre à cet effet très intéressant puisqu'il permet de sélectionner un patron sachant les exigences de la phase de conception concernée. A terme en poursuivant sur cette lancée, on peut même envisager la création d'un outil logiciel doté d'une interface graphique qui permettra aux experts en pédagogie de créer des LEA selon leurs besoins.

Ce processus de validation permet de conclure premièrement que la démarche de conception des LEA présentée au chapitre 4 permet effectivement de concevoir un LEA en suivant les phases proposées dans la démarche. Deuxièmement ce processus de validation permet aussi de conclure que les patrons proposés sont bien valides et fonctionnels car les patrons utilisés dans les deux cas présentés ont permis d'obtenir dans chacun des cas un logiciel fonctionnel.

Toutefois alors que les patrons que nous avons proposés sont liés surtout aux aspects technologiques et conceptuels du processus de développement des LEA, nous constatons à l'issue de ce processus de validation que l'intégration harmonieuse des exigences pédagogiques, à travers le processus de conception intégral doit être davantage élaborée. En effet, les exigences pédagogiques ne sont pas encore bien latentes au niveau d'un grand nombre de patrons que nous avons proposés. De plus, le répertoire de patrons doit être davantage spécifié afin de ne contenir que les patrons directement liés à la fonction d'enseignement et d'apprentissage des logiciels d'enseignement et d'apprentissage.

Chapitre 6

Discussion et conclusion

Dans le cadre de cette thèse, nous nous sommes attelé à proposer une solution pouvant contribuer à faire évoluer le processus de conception des LEA. Nous sommes partis de questions de recherche et de trois objectifs que nous avons développés. Nous avons ensuite validé la solution que nous avons proposée, observé ses limites et noté des perspectives de recherche pour la suite.

6.1 Résumé

Nous avons opté d'organiser notre thèse autour des cinq chapitres ci-dessous résumés.

Dans le chapitre 1, il s'est agit de définir le contexte du sujet et les concepts de base permettant de le cerner, relevant surtout du domaine de la pédagogie. Cet exercice s'est révélé fort délicat car il a fallu sélectionner des définitions, sinon essayer de définir des termes et des concepts dans un champs de connaissance où les experts entre eux n'accordent pas toujours leurs violons. Cette entrée en matière nous a permis de passer à un niveau plus technique celui de l'étude des modèles d'ingénierie.

Au chapitre 2 dédié à l'état de l'art de cette thèse, l'étude des principaux modèles d'ingénierie pédagogique nous a permis de comprendre comment les scénarios pédagogiques sont interprétés, modélisés et développés par les experts en pédagogie que ceux du génie logiciel. Nous avons présenté une liste de travaux visant à concevoir ou améliorer les LEA. Nous avons observé qu'au coeur des développements pédagogiques et informatiques des processus de conception de logiciels d'enseignement et d'apprentissage, se trouve le concept patron que nous retrouvons dans la longue liste de travaux que nous avons étudiés à cet effet. Cela nous a permis de conclure que l'utilisation du concept *patron* dans le but de concevoir ou d'améliorer les LEA dans les différents travaux présentés exprime le souci des chercheurs de capitaliser l'experience de la chaîne d'experts impliqués dans le processus de conception de ces systèmes. L'étude comparative effectuée nous a permis d'identifier avec précision la pierre angulaire de

la contribution de cette thèse.

Au chapitre 3, nous avons proposé une approche complète, séquentielle et itérative de de conception d'un LEA constituée de huit étapes. Nous avons aussi présenté les formalismes de représentation des patrons qui constituent notre approche, et avons ensuite proposé et présenté trois niveaux d'abstraction logique de ces patrons, parmi dont le niveau paquetage. Enfin nous avons aussi proposé un guide de recherche de patrons basé sur un arbre de dépendance des patrons qui permet de rechercher et de sélectionner des patrons du répertoire de patrons de façon méthodique.

Au chapitre 4, nous avons proposé un répertoire constitué de 110 patrons. Nous avons présenté les paquetages identifiés au chapitre précédent, et pour chacun des paquetages avons proposé des patrons pédagogiques, des patrons pour la conception de l'architecture logicielle, des patrons pour la conception de l'interface utilisateur, des patrons pour les choix technologiques à effectuer, des patrons pour l'implémentation à réaliser, des patrons pour le processus de validation, des patrons pour le déploiement à mettre en oeuvre, et des patrons pour la phase d'évolution du logiciel. Pour évaluer le réalisme et la faisabilité de la démarche de conception proposée et vérifier que les patrons proposés sont pertinents et complets nous avons dû soumettre notre démarche et nos patrons à un processus de validation.

Au chapitre 5, un cadre de validation de la démarche proposée dans cette thèse de conception des LEA basée sur les patrons a été proposé. Deux exemples ont servi de cadre d'application de la démarche de conception des LEA et d'utilisation des patrons proposés. Ce processus de validation a permis de conclure d'une part au réalisme de notre démarche de conception des LEA et d'autre part que les patrons proposés sont bien valides et fonctionnels.

6.2 Discussion des résultats

Au début de cette thèse, nous avons identifié les trois objectifs de recherche suivants auxquels des réponses ont été apportées.

Proposer une démarche méthodique basée sur le concept de *patron*, pour concevoir un LEA. Il s'agira de montrer quelles doivent être les étapes séquentielles de conception d'un LEA prenant en compte les exigences pédagogiques

Cet objectif a été atteint. Nous avons effectivement défini une nouvelle approche complète, séquentielle et itérative de conception des LEA basée sur les patrons et l'ingénierie dirigée par les modèles. Cette approche de conception permet de prendre en compte les exigences pédagogiques dès le début du processus de conception. Les exigences pédagogiques formulées par les experts en pédagogie sont décrites chacune à l'aide d'un diagramme d'exigences puis d'un diagramme de définition de blocs et d'un diagramme d'activités. Les formalismes ainsi

obtenus sont intégrés au reste du processus de conception du LEA. Le processus proposé de conception d'un LEA est constitué d'un ensemble de huit phases formant une chaîne de progression chronologique et itérative. Nous avons aussi présenté deux formalismes, textuel et graphique de représentation des patrons de notre approche. Enfin, nous avons proposé un guide de recherche de patrons qui permettra de rechercher et de sélectionner des patrons adéquats du répertoire de patrons de façon méthodique. Dans le chapitre 5, à travers les deux études de cas présentés, nous avons démontré le réalisme et la faisabilité de l'approche de conception proposée.

Définir le répertoire de patrons qui seront intégrés dans la démarche

Nous avons aussi atteint cet objectif. En effet pour chacune des phases de l'approche de conception définie, nous avons recensé un ensemble de patrons adéquats organisés en huit paquetages. Le répertoire de patrons ainsi obtenu permettra à l'équipe de conception d'un LEA de savoir sur quelles bases débuter le processus de conception d'un LEA. Un tel catalogue de patrons permet à l'équipe en charge de la conception du LEA d'apprendre de l'expérience de ses pairs, d'éviter des erreurs dues éventuellement à son manque d'expérience et aussi de gagner du temps et de la performance. Le répertoire de patrons ainsi obtenu n'est pas exhaustif et peut être continuellement enrichi. Dans le chapitre 5, à travers les deux études de cas présentés, nous avons montré que les patrons proposés sont valides et fonctionnels.

Proposer une interface multipartite des experts en pédagogie, en technologie éducative et en ingénierie logicielle leur permettant de communiquer dans le but de développer un LEA.

Cet objectif a été partiellement atteint par la définition d'un cadre de définition des exigences pédagogiques pouvant être exploitées par les ingénieurs logiciels. Nous avons montré comment des exigences pédagogiques peuvent être définies et traduites à l'aide des formalismes du langage SySML dans un format pouvant être exploité par les ingénieurs logiciels. Toutefois le processus de traduction des exigences pédagogiques tel que présenté n'est pas encore très intuitif et n'est pas encore très linéaire. Il y a encore des efforts à fournir afin de pouvoir traduire en artéfacts exploitables par les ingénieurs logiciels les exgicences pédagogiques les plus complexes et les plus abstraites.

6.3 Avantages et limites

Comme avantage, nous observons que l'approche que nous avons proposée est générique car elle est indépendante de toute plateforme logicielle ou matérielle. Ensuite bien que l'intention au départ était précisément de proposer une approche de conception des LEA, nous constatons que la solution proposée peut être étendue et généralisée au processus de conception d'un logiciel en général. De plus les patrons proposés permettent de capitaliser l'expérience des

experts en conception de LEA aux fins de constitution d'une mémoire de la connaissance, et aussi à l'intention des experts moins expérimentés en matière de développement de LEA. D'autre part comme nous l'avons montré lors du processus de validation les patrons proposés simplifient d'une belle manière le processus de conception logicielle et le rend plus accessible aux personnes qui n'ont pas tous les rudiments en la matière.

Outre l'approche de conception, nous avons aussi proposé un repertoire de 110 patrons pouvant aider à la conception de LEA. Ce répertoire de patrons n'est pas exhaustif et peut éventuellement être enrichi.

Cependant sachant que *toute oeuvre humaine est par définition imparfaite et est par conséquent perfectible*, nous sommes donc conscients du caractère imparfait de notre oeuvre. Ci-dessous nous dressons une liste des limites de cette thèse.

Alors que les patrons que nous avons proposés sont liés surtout aux aspects technologiques et conceptuels du processus de développement des LEA, nous constatons à l'issue du processus de validation que la vectorisation des exigences pédagogiques à travers le processus de conception intégral doit être davantage élaborée. En effet, les exigences pédagogiques ne sont pas encore bien latentes au niveau d'un grand nombre de patrons que nous avons proposés. De plus le répertoire de patrons doit être davantage spécifié afin de ne contenir que les patrons ayant traits à la nature spécifique des LEA, c'est-à-dire ceux directement liés à la fonction d'enseignement et d'apprentissage des LEA. Cela permettra de diminuer le nombre plétorique de patrons. Nous observons aussi d'autres limites de notre thèse. Il s'agit d'abord de l'absence d'un avis critique d'un expert en pédagogie concernant les aspects relevant du domaine pédagogique de cette thèse alors que le sujet traité dans le cadre de cette thèse est bien multidisciplinaire et fait s'entrecroiser les domaines de la pédagogie et du génie logiciel. Ensuite nous notons le fait que nous n'ayons pas pu effectuer de cas d'étude à l'échelle réelle et industrielle pour la validation, faute d'avoir eu accès à des personnes ressources pour les informations et à des données réelles pour le processus de validation. Enfin nous observons le fait que les patrons que nous avons proposé sont des patrons de haut niveau, abstraits : ils sont non instanciés par exemple sous la forme de classes rendant leur utilisabilité difficile.

6.4 Perspectives de recherche

Plusieurs perspectives à nos travaux peuvent être envisagées. D'abord le travail peut être poursuivi par l'instantiation des patrons proposés dans le cadre de cette thèse car ceux-ci sont de haut niveau et très abstraits. Il faudrait donc les formaliser en utilisant par exemple le formalisme de la *classe* en langage UML ou le formalisme du *block* en langage SysML. Ensuite il est possible d'aller plus loin jusqu'à la proposition d'un métamodèle puis d'un méta-métamodèle basé sur des patrons pour la conception des LEA. Cet ensemble modèle, métamodèle et méta-métamodèle pourra permettre la définition d'un langage de modélisation

typique des LEA. Une troisième perspective est le développement d'une méthode plus évoluée et intelligente de recherche de patrons dans un grand répertoire permettant de retrouver facilement un patron. Il est aussi possible d'ajouter une couche sémantique au niveau du LEA qui permettra non seulement d'adapter les scénarios pédagogiques, mais aussi d'automatiser le processus d'adaptation, au besoin d'un étudiant en particulier. Enfin nous entrevoyons encore comme perspective la transformation des patrons proposés en ontologies pouvant permettre de faciliter l'évaluation des connaissances de l'étudiant, de lui communiquer des informations structurées utiles pour son apprentissage et correspondant à son besoin d'apprentissage.

Liste des publications

Conférences internationales avec comité de lecture

1. *Toward a Pattern-based e-learning software design approach* TF Guinko, L Capus, H Ben Sta - EdMedia 2015 - Montreal, Quebec, Canada, June 22-24, 2015 - learntechlib.org
2. *How to Take Pedagogical Requirements into account to Design E-Learning Platforms by Using Patterns ?* F Guinko, L Capus, H Ben Sta - E-Learn 2015 - Kona, Hawaii, United States, October 19-22, 2015 - learntechlib.org

Bibliographie

Agarwal, Shalabh, Nath, Asoke, & Chowdhury, Dipayan. 2012. Sustainable approaches and good practices in green software engineering. *International Journal of Research and Reviews in Computer Science (IJRRCS)*, **3**(1).

Avgeriou, Paris, & Zdun, Uwe. 2005. Architectural patterns revisited – a pattern language. *Pages 1–39 of : In 10th European Conference on Pattern Languages of Programs (EuroPlop 2005), Irsee.*

Avgeriou, Paris, Papasalouros, Andreas, & Retalis, Simos. 2003. Patterns for Designing Learning Management Systems. *Pages 115–132 of : EuroPLoP.*

Barré, Vincent. 2004. EMLs : case study in distance learning education. *In : Proceedings of International Conference on Computer Aided Learning in Engineering education CALIE04, 16-18 February 2004.*

Barré, Vincent, Choquet, Christophe, Corbiere, Alain, Cottier, Philippe, Dubourg, Xavier, & Gounon, Patricia. 2003. MOCA, une approche expérimentale de l'ingénierie des EIAH. *Pages 55–66 of : Environnements Informatiques pour l'Apprentissage Humain 2003.*

Basque, Josianne. 2011. TED 6312. *Ingénierie pédagogique et technologies éducatives. Téléuniversité. Université du Québec à Montréal.*

Beck, Kent, & Johnson, Ralph. 1994. Patterns generate architectures. *Object-Oriented Programming*, 139–149.

Berczuk, Steve. 1994. Finding solutions through pattern languages. *Computer*, **27**(12), 75–76.

Bergin, Joseph. 2000. Fourteen Pedagogical Patterns. *Pages 1–49 of : EuroPLoP.* Citeseer.

Bergin, Joseph. 2004. *Two pedagogical patterns for course design.* Disponible à l'adresse : https://www.semanticscholar.org/paper/Two-Pedagogical-Patterns-for-Course-Design-Bergin/c0bdc3e96e3fca4d75d4dff3045ed9a2fe4e0990/pdf. Consulté le : 05-05-2011.

Borchers, Jan O. 2001. A pattern approach to interaction design. *AI & SOCIETY*, **15**(4), 359–376.

Bouabib, Mohamed El Amine. 2005. *Apports de la modélisation IMS LD Dans le projet MEPA 2D*. M.Phil. thesis, Université du Maine, France.

Branch, Robert Maribe, & Gustafson, Kent L. 1998. *Re-visioning models of instructional development*. Springer.

Burgos, Daniel, Arnaud, Michel, Neuhauser, Patrick, & Koper, Rob. 2006. *IMS Learning Design : la flexibilité pédagogique au service des besoins de l'e-formation, 2006*.

Buschmann, Frank, Meunier, Regine, Rohnert, Hans, Sommerlad, Peter, & Stal, Michael. 1996. A system of patterns : pattern-oriented software architecture. *West Sussex, England : John Wiley & Sons*.

Carle, Andy, Clancy, Michael, & Canny, John. 2007. Working with pedagogical patterns in PACT : initial applications and observations. *ACM SIGCSE Bulletin*, **39**(1), 238–242.

Chen, Irene. 2008. Instructional design methodologies. *Handbook of research on instructional systems and technology*, 1–14.

Corbière, Alain, & Choquet, Christophe. 2004. Designer integration in training cycles : IEEE LTSA model adaptation. *In : Proceedings of CALIE 2004, International Conference on Computer Aided Learning in Engineering Education, 16-18 February 2004, Grenoble, France*.

Coutinho Anacleto, Junia, Talarico Neto, Americo, & Neris, Vania Paula de Almeida. 2009. Cog-Learn : An e-Learning Pattern Language for Web-based Learning Design. *eLearn*, **2009**(8), 1.

Culatta, Richard. 2005. *Instructional Design Models*. Disponible à l'adresse : `http://www.instructionaldesign.org/models/`. Consulté le : 22-01-2012.

de Moura Filho, César Olavo, & Derycke, Alain. 2007. Concevoir des scénarios pédagogiques exécutables avec des patrons de conception pédagogiques. *In : Actes de la conférence EIAH 2007*. INRP.

de Moura Filho, César Olavo de. 2007. *MDEduc : conceiving and implementing a language-oriented approach for the design of automated learning scenarios*. Ph.D. thesis, Lille 1.

de Moura Filho, César Olavo de. 2009. *Designing e-learning applications : a language-oriented approach*. LAP Lambert Academic Publishing.

Derntl, Michael. 2004. The Person-Centered e-Learning pattern repository : Design for reuse and extensibility. *Pages 3856–3861 of : World Conference on Educational Multimedia, Hypermedia and Telecommunications*, vol. 2004.

Derntl, Michael. 2005. *Patterns for Person-Centered e-Learning.* Ph.D. thesis, Université de Vienne.

Devedzic, Vladan, & Harrer, Andreas. 2005. Software Patterns in ITS Architectures. *Int. J. Artif. Intell. Ed.*, **15**(2), 63–94.

Dick, Walter. 1996. The Dick and Carey model : Will it survive the decade ? *Educational Technology Research and Development*, **44**(3), 55–63.

Don, Clark. 2010. *ADDIE timeline.* Disponible à l'adresse : http://www.nwlink.com/~donclark/history_isd/addie.html. Consulté le : 09-01-2014.

Downes, Stephen. 2005. *E-learning 2.0.* Disponible à l'adresse : http://doi.acm.org/10.1145/1104966.1104968. Consulté le : 22-05-2010.

Enard, Quentin. 2013. *Développement d'applications logicielles sûres de fonctionnement : une approche dirigée par la conception.* Ph.D. thesis, Université Sciences et Technologies-Bordeaux I.

Fincher, Sally, & Utting, Ian. 2002. Pedagogical patterns : their place in the genre. *Pages 199–202 of : ACM SIGCSE Bulletin*, vol. 34. ACM.

Finlay, Janet, Gray, John, Falconer, Isobel, Hensman, Jim, Mor, Yishay, & Warburton, Steven. 2009. *Planet : Pattern Language Network for Web 2.0 in Learning. Technical report.* Disponible à l'adresse : https://halshs.archives-ouvertes.fr/file/index/docid/592752/filename/PlanetFinalReportv1.3.pdf. Consulté le : 11-10-2011.

Forest, Ed. 2014. *The ADDIE Model : Instructional Design.* Disponible à l'adresse : http://educationaltechnology.net/the-addie-model-instructional-design/. Consulté le : 10-11-2016.

Frey, Daniel D., & Dym, Clive L. 2006. Validation of design methods : lessons from medicine. *Research in Engineering Design*, **17**(1), 45–57.

Gamma, Erich. 1995. *Design patterns : elements of reusable object-oriented software.* Addison-Wesley Professional.

Goodyear, Peter. 2005. Educational design and networked learning : Patterns, pattern languages and design practice. *Australasian Journal of Educational Technology*, **21**(1), 82–101.

Goodyear, Peter, Avgeriou, Paris, Baggetun, Rune, Bartoluzzi, Sonia, Retalis, Simeon, Ronteltap, Frans, & Rusman, Ellen. 2004. Towards a pattern language for networked learning. *Pages 449–455 of : proceedings of networked learning.*

Gustafson, Kent L., & Branch, Robert Maribe. 2002. *Survey of instructional development models.* ERIC Clearinghouse.

Haberman, Bruria. 2006. Pedagogical patterns : A means for communication within the CS teaching community of practice. *Computer Science Education*, **16**(2), 87–103.

Hachani, Ouafa. 2006. *Patrons de conception à base d'aspects pour l'ingénierie des systèmes d'information par réutilisation.* Ph.D. thesis, Université Joseph-Fourier-Grenoble I.

Hadjerrouit, Said. 2007. Applying a system development approach to translate educational requirements into e-learning. *Interdisciplinary Journal of E-Learning and Learning Objects*, **3**(1), 107–134.

Hadjerrouit, Saïd. 2005. Developing web-based learning systems. A skill-based approach. *Proceedings of ELEARN 2005*, 24–28.

Holz, Hilary J, Applin, Anne, Haberman, Bruria, Joyce, Donald, Purchase, Helen, & Reed, Catherine. 2006. Research Methods in Computing : What are they, and how should we teach them ? *Pages 96–114 of : ACM SIGCSE Bulletin*, vol. 38. ACM.

Iacob, Claudia. 2011. Identifying, relating, and evaluating design patterns for the design of software for synchronous collaboration. *Pages 323–326 of : Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems.* ACM.

Iacob, Claudia. 2012. Using design patterns in collaborative interaction design processes. *Pages 107–110 of : Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion.* ACM.

Jones, Paula, & Davis, Rita. 2011. Instructional design methods integrating instructional technology. *Instructional design : Concepts, methodologies, tools and applications*, 101–113.

Karampiperis, Pythagoras, & Sampson, Demetrios. 2005. Towards next generation activity-based web-based educational systems. *Pages 868–872 of : Advanced Learning Technologies, 2005. ICALT 2005. Fifth IEEE International Conference on.* IEEE.

Kohls, Christian. 2014. *The theories of design patterns and their practical implications exemplified for e-learning patterns.* Ph.D. thesis, Eichstätt-Ingolstadt, Katholische Universität Eichstätt-Ingolstadt, Diss., 2014.

Kompen, Ricardo Torres, Edirisingha, Palitha, & Mobbs, Richard. 2008. *Building Web 2.0-based personal learning environments-a conceptual framework.* Disponible à l'adresse : `http://hdl.handle.net/2381/4398`. Consulté le : 22-07-2011.

Köppe, Christian. 2012. Continuous activity : a pedagogical pattern for active learning. *Page 3 of : Proceedings of the 16th European Conference on Pattern Languages of Programs.* ACM.

Kumar, Prveen, Samaddar, Shefalika Ghoch, Samaddar, Arun B, & Misra, Arun Kumar. 2010. Extending IEEE LTSA e-Learning framework in secured SOA environment. *Pages V2–136 of : Education Technology and Computer (ICETC), 2010 2nd International Conference on*, vol. 2. IEEE.

Lebrun, Marcel. 2002. Courants pédagogiques et technologies de l'éducation. *Louvainla-Neuve : Institut de pédagogie universitaire et des multimédias.*

Lindner, Rolf. 2006. Architectures and frameworks. *Pages 193–208 of : Handbook on quality and standardisation in e-learning.* Springer.

Littlejohn, Allison, & Pegler, Chris. 2014. *Preparing for blended e-learning.* Routledge.

Lu, Peng, Cong, Xiao, & Zhou, Dongdai. 2015. E-learning-Oriented Software Architecture Design and Case Study. *International Journal of Emerging Technologies in Learning*, **10**(4).

Mannhardt, Axel. 2007. *Learning Technology Systems Architecture.* Disponible à l'adresse : http://www.fh-wedel.de/~si/seminare/ws07/Ausarbeitung/01.ilearn/node3.html. Consulté le : 09-01-2014.

Mitchell, Maria, & Tashjian, Susan. 2014. *Rapid Prototyping Model.* Disponible à l'adresse : http://insdsg619.wikispaces.com/Rapid. Consulté le : 03-02-2015.

MITx, & HarvardX. 2014. *HarvardX-MITx Person-Course Academic Year 2013 De-Identified dataset, version 2.0.* Disponible à l'adresse : http://dx.doi.org/10.7910/DVN/26147. Téléchargé le : 19-09-2015.

Mor, Yishay, Winters, Niall, Cerulli, Michele, & Björk, Steffan. 2006. *Literature review on the use of games in mathematical learning, Part I : Design. Report of the Learning Patterns for the Design and Deployment of Mathematical Games project.*

Mor, Yishay, Mellar, Harvey, Warburton, Steven, & Winters, Niall. 2014. *Practical Design Patterns for Teaching and Learning with Technology.* Vol. 8. Springer.

Moreno Ger, Pablo, Sancho Thomas, Pilar, Martínez Ortiz, Iván, Sierra, José Luis, & Fernández Manjón, Baltasar. 2007. Adaptive units of learning and educational videogames. *Journal of Interactive Media in Education*, **2007**(1).

Morrison, Gary R., Ross, Steven M., Kemp, Jerrold E., & Kalman, Howard. 2010. *Designing effective instruction.* John Wiley & Sons.

Murphy, Timothy, Price, Jerry, Stevens, William, Jackson, Kevin, Villareal, James A., & Way, Bob. 1994. Literacity : a multimedia adult literacity package combining NASA technology, recursive ID theory, and authentic instruction theory. *Page 365 of : Technology 2003 : conference proceedings : the Fourth National Technology Transfer Conference and*

Exposition, December 7-9, 1993, Anaheim, CA, vol. 2. National Aeronautics and Space Administration.

O'Droma, Mairtin S., Ganchev, Ivan, & McDonnell, Fergal. 2003. Architectural and functional design and evaluation of e-learning VUIS based on the proposed IEEE LTSA reference model. *The Internet and Higher Education*, **6**(3), 263–276.

Oubahssi, Lahcen, & Grandbastien, Monique. 2008. E-Learning Systems Re-Engineering : Functional Specifications and Component-Based Architecture. *Architecture Solutions for E-Learning Systems*, 175–194.

Paquette, Gilbert. 2002. *L'ingénierie pédagogique : pour construire l'apprentissage en réseau.* Puq.

Paquette, Gilbert, Léonard, Michel, de la Teja, Ileana, & Dessaint, M. Paule. 2001. Méthode d'ingénierie d'un système d'apprentissage MISA 4.0 Présentation de la méthode. *Une*, 39.

Project, The Pedagogical Patterns. 2002. *What are Pedagogical Patterns. Technical report.* Disponible à l'adresse : http://www.pedagogicalpatterns.org/. Consulté le : 11-10-2011.

Quintin, Jean-Jacques. 2011. *Modèle ADDIE.* Disponible à l'adresse : http://www.edu-tice.org/approche-m%C3%A9thodologique/mod%C3%A8le-addie/. Consulté le : 09-01-2014.

Qureshi, Elena. 2004. *Instructional Design Models.*

Randriamalaka, Noa. 2005. Design patterns approach for usage analysis in re-engineering process of learning systems. *Pages 308–310 of : Advanced Learning Technologies, 2005. ICALT 2005. Fifth IEEE International Conference on.* IEEE.

Randriamalaka, Noa. 2006. Construction d'une base de patterns pour la réingénierie de scénarios pédagogiques. *Rencontres Jeunes Chercheurs EIAH*, **2006**, 35–42.

Randriamalaka, Noa, & Iksal, Sébastien. 2006. Patterns approach in the re-engineering process of learning scenario. *Pages 799–803 of : Advanced Learning Technologies, 2006. Sixth International Conference on.* IEEE.

Randriamalaka, Noa, Iksal, Sébastien, & Choquet, Christophe. 2008. Elicitation des indicateurs pour la ré-ingénierie des scénarios pédagogiques : Approche à base de traces utilisant UTL. *In : INFORSID.*

Retalis, Symeon, Georgiakakis, Petros, & Dimitriadis, Yannis. 2006. Eliciting design patterns for e-learning systems. *Computer Science Education*, **16**(2), 105–118.

Riehle, Dirk, & Züllighoven, Heinz. 1996. Understanding and using patterns in software development. *TAPOS*, **2**(1), 3–13.

Roques, Pascal. 2013. *SysML par l'exemple-Un langage de modélisation pour systèmes complexes*. Editions Eyrolles.

Rudzki, Jakub, & Hammouda, Imed. 2004. *Pattern Classification. Technical report.*

Sauvé, Lucie. 1992. *Éléments d'une théorie du design pédagogique en éducation relative à l'environnement : élaboration d'un supramodèle pédagogique*. Montréal : Université du Québec à Montréal.

Schmidt, Douglas C. 1995. Using design patterns to develop reusable object-oriented communication software. *Communications of the ACM*, **38**(10), 65–74.

Schmolitzky, Axel, & Schümmer, Till. 2008. Patterns for Supervising Thesis Projects. *In : EuroPLoP*. Citeseer.

Seffah, Ahmed, & Grogono, Peter. 2002. Learner-centered software engineering education : From resources to skills and pedagogical patterns. *Pages 14–21 of : Software Engineering Education and Training, 2002.(CSEE&T 2002). Proceedings. 15th Conference on*. IEEE.

Shi, Lei, Cristea, Alexandra I, Awan, Malik Shahzad K, Hendrix, Maurice, & Stewart, Craig. 2013. Towards understanding learning behavior patterns in social adaptive personalized e-learning systems. *In : Proceedings of the 19th Americas Conference on Information Systems (AMCIS 2013)*. AMCIS.

Simmons, Shelby Elise. 2012. The Origins and Descriptions of ADDIE. *Origins*.

Smith, Patricia L., & Ragan, Tillman J. 1999. *Instructional design*. Wiley New York, NY.

Stracke, Christian M. 2006. Interoperability and Quality Development in e-Learning. Overview and Reference Model for e-Learning Standards. *Proceedings of the Asia-Europe e-Learning Colloquy. e-ASEM, Seoul*.

Tešanovic, Aleksandra. 2004. *What is a pattern ?* Disponible à l'adresse : `http://www.idi.ntnu.no/emner/dt8100/papers2005/P-a10-tesanovic04.pdf`. Consulté le : 25-04-2010.

Tripp, Steven D., & Bichelmeyer, Barbara. 1990. Rapid prototyping : An alternative instructional design strategy. *Educational Technology Research and Development*, **38**(1), 31–44.

Turani, Aiman, & Calvo, Rafael A. 2006. Beehive : A software application for synchronous collaborative learning. *Campus-Wide Information Systems*, **23**(3), 196–209.

Université Laval, Bureau des Services Pédagogiques (BSP). 2013. *Forum de discussion : guide des bonnes pratiques enseignantes*. Consulté le : 17-09-2015.

Université Laval, Bureau des Services Pédagogiques (BSP). 2015. *Programme d'appui au développement de cours hybrides 2015-2016*. Disponible à l'adresse : `http://www.bsp.ulaval.ca/docs/Programme-hybride_V30_juin_2015.pdf`. Consulté le : 17-09-2015.

Université Laval, Vice-Rectorat aux Etudes et aux Activités Internationales (VREAI). 2012. *Politique de la formation à distance.* Disponible à l'adresse : https://www2.ulaval.ca/fileadmin/Secretaire_general/Politiques/Politique_de_la_formation_a_distance.pdf. Consulté le : 17-09-2015.

Vyatkin, Valeriy. 2013. Software engineering in industrial automation : State-of-the-art review. *Industrial Informatics, IEEE Transactions on*, **9**(3), 1234–1249.

Willis, Jerry. 1995. A Recursive, Reflective Instructional Design Model Based on Constructivist-Interpretivist Theory. *Educational Technology*, **35**(6), 5–23.

Wilson, Scott, Liber, Oleg, Johnson, Mark W., Beauvoir, Philip, Sharples, Paul, & Milligan, Colin D. 2007. Personal Learning Environments : Challenging the dominant design of educational systems. *Journal of e-Learning and Knowledge Society*, **3**(2), 27–38.

Winn, Tiffany, & Calder, Paul. 2002. *Is This a Pattern ?* Disponible à l'adresse : http://dx.doi.org/10.1109/52.976942. Consulté le : 25-04-2010.

Winn, William. 1992. The assumptions of constructivism and instructional design. *Constructivism and the technology of instruction : A conversation*, 177–182.

Zeid, Amir, & Salah, Dina. 2010. Using Pattern Languages to Design Intelligent Tutoring Systems : A Case Study. *International Journal of Computer Science and Network Security*, **10**(9), 164–170.

Annexes

Annexe 1 : premier exemple de validation : décompte du nombre de mots postés par étudiant.

L'annexe 1 présente le code permettant de lire la base de données *MITx and HarvardX Dataverse* (MITx & HarvardX, 2014) et d'afficher le nombre de messages postés par jour. Cette base de données contient l'ensemble des données collectées pendant l'année académique 2012-2013 sur une plateforme logicielle d'enseignement et d'apprentissage détenue conjointement par les universités Harvard et le MIT.

La base de données *MITx and HarvardX Dataverse* a été utilisée en lieu et place des données réelles de la base de données de Knowledge Forum 6 auxquelles nous n'avons pas pu accéder.

```
1
2     <!DOCTYPE html>
3     <html>
4       <head>
5         <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
6         <script type="text/javascript" src="jquery-2.1.4.js"></script>
7         <script type="text/javascript">
8         $(document).ready(function ()
9         {
10
11          $.ajax({
12            type: "GET",
13            url:"figure3.csv",
14            dataType: "text",
15            success: function(data) {processData(data);}
16          });
17
18        function processData( allText )
19        {
20          var allLinesArray = allText.split("\n");
21          if( allLinesArray.length > 0 )
22          {
23            var dataPoints = [];
24            for (var i = 0; i <= allLinesArray.length-1; i++)
```

```
25              {
26                var rowData = allLinesArray[i].split(",");
27                dataPoints.push({ label:rowData[17], y:parseInt(rowData[2]) });
28                //dataPoints.push({ label:rowData[3], y:parseInt(rowData[2]) });
29              }
30              drawChart(dataPoints);
31            }
32          }
33
34          function drawChart(dataPoints)
35          {
36            var chart = new CanvasJS.Chart("chartContainer",
37            {
38                theme: "theme2", //theme1
39                title:
40                {
41                  //text: "Graphiques"
42                },
43
44                axisX:
45                {
46                  //interval: 5,
47                  labelAngle: 90,
48                  labelWrap:true,
49                  labelAutoFit: false,
50                  labelFontSize: 15,
51                  labelMaxWidth: 200,
52                  labelFontColor: "black"
53                },
54
55                axisY:
56                {
57                  interval: 10,
58                  labelAngle: 0,
59                  labelWrap:true,
60                  labelAutoFit: false,
61                  labelFontSize: 15,
62                  labelMaxWidth: 200,
63                  labelFontColor: "black"
64                },
65
66                animationEnabled: false,   // changer \'a true
67                zoomEnabled:true,
68                data: [
69                {
70                  type: "column",
71                  indexLabelPlacement: "outside",
72                  indexLabelFontWeight: "bold",
```

```
73                indexLabelFontColor: "black",
74                dataPoints: dataPoints
75              },
76
77              ]
78              });
79        chart.render();
80        }
81      });
82        </script>
83        <script type="text/javascript" src="canvasjs.min.js"></script>
84      </head>
85      <body style="background-color: #ffffff; background-image:url(../Images/
            bg_body_new.png); background-repeat: repeat-x;text-align:center">
86      <a href="http://tonguim.free.fr/these/canvajs/index2.html">Cliquez ici
            pour afficher le nombre d'inscription par jour</a>
87        <div id="chartContainer" style="height: 800px; width: 100%; background
              -image:url("fonto1.png"); background-repeat:no-repeat; background-
              position:center; background-size:100% 100%"></div>
88        <div style="text-align: center; color:blue; font-size:25px;"><b>Nombre
              de posts par jour</b></div>
89      </body>
90    </html>
```

Annexe 2 : deuxième exemple de validation : fonction «messages» : formulaire d'envoi d'un message sur le LEA

L'annexe 2 présente le code permettant de créer un formulaire d'écriture d'un message sur le forum de discussion à caractère pédagogique de l'étude de cas 2 présenté dans le chapitre 5.

```
1       <!DOCTYPE html>
2
3       <html>
4         <head>
5           <link rel="stylesheet" href="style.css">
6         </head>
7         <body>
8           <?php include ("connexion.php"); ?>
9           <h3><center>Sur cette, page, \'ecrivez le message que vous d\'esirez
                poster sur le forum de discussion du cours.</center></h3>
10          <form name="formulaire" method="POST" action="enregistrement.php">
11            <center><a href="Javascript:history.go(-1)">Retour \'a la page pr
                  \'ec\'edente</a></center><br>
12
13            <table class="box" width="700px" border="1" cellspacing="0"
                  cellpadding="0">
14             <tr>
```

```
15              <td><b>Nom et pr\'enom (s):</b></td>
16              <td><input name="txtnom" type="text" size="50"></td>
17             </tr>
18             <tr>
19              <td><b>Mon commentaire:</b> </td>
20              <td>  <textarea rows="10" cols="70" name="commentaires"> Entrez
                    votre commentaire ici. </textarea> </td>
21             </tr>
22             <tr>
23              <td colspan="2">
24            <table width="100\%" border="0" cellpadding="0" cellspacing="0">
25            <tr>
26              <td><div align="center">
27               <input type="submit" name="btonenvoyer" value="Envoyer">
28              </div></td>
29              <td><div align="center">
30               <input type="reset" name="btoneffacer" value="Effacer">
31              </div></td>
32              <td><div align="center">
33               <input type="button" name="btonannuler" value="Annuler">
34              </div></td>
35            </tr>
36              </table>
37            </td><!--fin de la fusion de colonnes dans le 1er tableau-->
38             </tr>
39            </table>
40          </form>
41        </body>
42      </html>
```

Annexe 3 : deuxième exemple de validation : formulaire d'envoi d'un message

L'annexe 3 présente le code permettant de valider le formulaire créé à l'aide du code présenté à l'annexe 2, et d'envoyer les données qu'il contient au serveur de base de données.

```
1
2     <!DOCTYPE html>
3     <html>
4       <head>
5         <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1
              " />
6         <meta http-equiv="Content-Language" content="fr">
7       </head>
8
9       <body>
10
```

```
<?php

  include "connexion.php";

  $nom=$_POST['txtnom'];
  $commentaires=$_POST['commentaires'];

    if(strlen($nom)<1 )
    {
      echo "<table>
            <tr>
            <td><center><img src=\"../images/divers/attention.gif\"></
                center></td><td><font face=\"Arial, Helvetica, sans-
                serif\" size=\"2\"><b>Attention!</b> V\'erifiez que le
                 champs NOM est effectivement remplit</font></td>
          </tr>
          <tr>
            <td colspan=\"2\">
              <font face=\"Arial, Helvetica, sans-serif\" size=\"2\"><
                 a href=\"javascript:history.go(-1)\"> <center>retour
                  au formulaire</center></a></font>
            </td>
          </tr>
       </table><br><br>";
    }

    $requete="insert into forumdiscussion (numero, nom, commentaire)
        values ('', '$nom', '$commentaires')";

    mysql_query($requete);

    ////////////////////// envoi du courriel

    ini_set("SMTP","smtp.mail.yahoo.fr");

    $sujet = "Nouveau message post\'e sur le forum de discussion";
        $de = stripslashes("$nom");
    $message="$nom vient d'envoyer le messsage suivant sur le forum de
         discussion du cours \n\n
    -<b>Nom:</b> $nom \n
    -<b>Commentaires:</b> $commentaires \n";

    $destinataire="adresseCourriel@yahoo.fr";

    if (mail($destinataire, $nom, $commentaires))
    {
              echo "<p><b>$nom, votre commentaire a \'et\'e envoy\'e
                 avec succ\`es!</b></p>";
```

```
51              echo"<br><center><a href='index.php'>Retour \'a la page d'
                    accueil</a></center><br>";
52                  //echo nl2br($message);
53                }
54            else
55            {
56                      echo "$nom, votre commentaire a pas \'et\'e enregistr\'e
                            ; je vous prie de reprendre la saisie de votre
                            commentaire";
57              echo"<br><center><a href='Javascript:history.go(-1)'>Retour \'a
                    la page pr\'ec\'edente</a></center><br>";
58            }
59        ?>
60      </body>
61    </html>
```

Annexe 4 : deuxième exemple de validation : fonction «curriculum» : Déposer un fichier

L'annexe 4 présente le code permettant de créer le formulaire d'envoi d'un fichier, par exemple un devoir à soumettre, sur le serveur de fichiers.

```
1
2     <!DOCTYPE html>
3     <html>
4     <head>
5         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6         <meta http-equiv="Content-Language" content="fr">
7
8     </head>
9       <body>
10        <form method="post" enctype="multipart/form-data" action="
              televersement.php">
11          <h3>Formulaire de t\'el\'eversement d'un fichier:</h3>
12          <input type="file" name="fich" id="fich" value="" /><br/><br/>
13          <input type="submit" id="importer" name="envoyer" value="Envoyer"/>
14        </form>
15      </body>
16    </html>
```