« Architectures Logicielles Ouvertes appliquées au Learning Design »

Encadrant principal :

- Mourad Oussalah – Equipe MOC, LINA CNRS FRE 2729
  Courriel : Mourad.Oussalah@univ-nantes.fr
  Tél. : 02 51 12 58 47

CO-Encadrants :

- Christophe Choquet – LIUM CNRS FRE 2730
  Courriel : Christophe.Choquet@lium.univ-lemans.fr
  Tél. : 02 43 59 49 27

- Dalila Tamzalit – Equipe MOC, LINA CNRS FRE 2729
  Courriel : Dalila.Tamzalit@univ-nantes.fr
  Tél. : 02 51 12 58 31

Résumé : le sujet proposé se résume en deux principaux points :

- La modélisation des différents concepts du langage de scénarisation pédagogique Learning Design selon une approche architectures logicielles à base de composants.
- La gestion, toujours selon l'approche composant, des dispositifs d'apprentissage lors de la modification du scénario d'apprentissage modélisé en Learning Design.

Introduction :

Le sujet de Master Recherche proposé s'inscrit dans le cadre d'une collaboration scientifique déjà existante entre deux groupes[1] : équipe MOC du LINA et groupe de travail REDIM du LIUM. Cette collaboration met en commun leurs compétences respectives : l'Enseignement A Distance pour le groupe REDIM, plus précisément la conception de scénarios pédagogiques, et l'approche composants dans les architectures logicielles de l'équipe MOC, plus précisément l'évolution de modèles à base de composants.

Les systèmes à base de composants sont des systèmes développés par assemblage de composants réutilisables, préfabriqués et pré-testés (à la manière de composants électroniques). Ils sont définis généralement par un ensemble de composants représentant les fonctionnalités du système et un ensemble d'interactions entre ces composants. Cette nouvelle manière de concevoir des systèmes a émergé de la communauté des architectures logicielles, qui a ainsi souligné l'importance de la description de l'architecture logicielle pour la conception de tels systèmes.

D'un autre coté un système n'est jamais figé, il est toujours amené à évoluer soit pour prendre en compte de nouveaux besoins fonctionnels, techniques ou matériels, soit pour adapter les besoins déjà exprimés, et cela à n'importe quelle phase de son cycle de développement. Aussi, outre la description d'un système sous forme d'architectures logicielles, un tel système nécessite d'évoluer et de voir son évolution gérée. De plus, ces architectures logicielles sont dites *ouvertes* car elles permettent d'être enrichies avec de nouvelles spécifications mais également en gardant trace des nouvelles situations d'évolution.

Contexte de travail :

Une architecture logicielle évolue pour répondre aux différents changements dans le système qu'elle décrit. Pour une architecture donnée, on peut être amené par exemple à lui ajouter de nouveaux composants pour décrire de nouvelles fonctionnalités, remplacer un composant par un autre ou encore réorganiser l'interaction entre ses différents composants. Ainsi tout changement dans

[1] Cette collaboration s'inscrit dans le cadre plus vaste d'un projet Etat/Région.

l'architecture doit être identifié, et géré afin d'éviter d'introduire des incohérences dans l'architecture amenée à évoluer. Pour ce faire, le modèle d'évolution SAEV a été développé au sein de l'équipe MOC. Il permet :

- L'abstraction de l'évolution, en la distinguant du comportement propre à tout élément de l'architecture.
- L'ajout de nouvelles méthodes d'évolutions, notamment celles non prévues initialement.
- La prise en compte de l'évolution aussi bien statique (lors des spécifications de l'architecture) que dynamique (lors de l'exécution de l'application) car l 'évolution peut intervenir à n'importe quelle phase du cycle du développement du logiciel.
- La sauvegarde de la cohérence de l'architecture au cours de l'évolution.

Tout langage de scénarisation pédagogique (tel que Learning Design), tout scénario pédagogique exprimé dans ces langages, peut être modélisé comme une architecture logicielle à base de composants. A titre d'exemple, un *scénario* peut être vu comme une *configuration* de composants et une *activité* de cette configuration comme un *composant*. Tous les composants sont décrits au sein de leur configuration et sont reliés entre eux par des interactions. Ainsi, un dispositif d'enseignement à distance peut s'appuyer sur cette description, notamment sur ces interactions, pour exécuter un ou plusieurs scénarios d'apprentissages. En considérant que les scénarios pédagogiques sont des objets évolutifs (adaptation dynamique en fonction de l'interaction, adaptation a posteriori dans une démarche de réingénierie), l'objectif est de permettre la modification et l'adaptation d'un scénario en tirant parti du modèle d'évolution SAEV.

Travail à réaliser :

Un premier travail a été fait dans le cadre de la collaboration LINA-LIUM. Le stage proposé s'inscrit pleinement dans ce cadre et consiste, au vu de ce qui a été présenté, à approfondir les premiers résultats en modélisant :

- Les différents concepts du langage Learning Design selon une approche architectures logicielles à base de composants.
- Les dispositifs d'apprentissage, au vu des mécanismes d'évolution de SAEV, lors de la modification d'un scénario d'apprentissage.

Il est à noter que ce travail pourra bénéficier de résultats d'expérimentation permettant de tester la validité des résultats.

Mots-clés : Composants logiciels, Evolution, Scénarisation pédagogique.

Références bibliographiques

Référence principale :

- N.Sadou, D.Tamzalit, M.Oussalah : « How to manage uniformly Software Architecture at different abstraction levels », A paraître dans les proceeding of the 24th ACM international conference on conceptual Modeling ER 2005 ;

Référence secondaire :

- Hassina El-Kechai, Christophe Choquet, « Approche pragmatique de conception d'un EIAH : Réingénierie pédagogique dirigée par les modèles », In: EIAH 2005, 25-27 mai 2005, Montpellier (France), 189-200.

Bibliographie complémentaire :

- N. Medvidovic, D.S.Rosenblum, and R.N. Taylor: A Language and Environment for Architecture-based Software Development and evolution. In proceeding of the 21st international conference en Software engineering , pp.44-53, may 1999.