Comment intégrer des logiciels issus de la recherche en EIAH ?

Marilyne Rosselle, Marie-Noelle Bessagnet, Thibault Carron

Sciences et Technologies de l´Information et de la Communication pour l´Éducation et la Formation

Volume 12, 2005
Article de recherche

Comment intégrer des logiciels issus de la recherche en EIAH ?

Marilyne Rosselle [SaSo, Amiens]

Marie-Noëlle Bessagnet [Liuppa, Pau]

Thibault Carron [SysCom, Chambéry]

■ RÉSUMÉ : Cet article aborde le problème de la ré-utilisation, dans un but de recherche, de logiciels EIAH sous forme de prototypes ou d'environnements développés dans d'autres équipes de recherche. En nous appuyant sur des travaux traitant des problèmes d'interopérabilité et d'intégration, nous tentons d'illustrer cette problématique étudiée collectivement au sein de l'AS « Plates-formes technologiques pour la recherche en EIAH ». Après avoir analysé les besoins d'intégrer des logiciels EIAH pour construire une activité d'apprentissage, nous argumentons autour d'une proposition d'architecture d'intégration. Notre analyse des besoins débouche sur une présentation des différentes dimensions d'intégration possible. Au regard de ces dimensions d'intégration, nous présentons cinq études de cas issus de la communauté française. Nous concluons en proposant un guide des bonnes pratiques pour l'intégration.

■ MOTS CLÉS : recommandations, approche par composants, plate-forme d'intégration, interopérabilité, EIAH, études de cas

■ ABSTRACT : This article tackles the problem of educational software re-use in research purpose. Theses pieces of software are prototypes or environments developed by other research teams. After having analyzed the requirements to integrate educational software in order to build an experiment, we argue around an integration architecture proposition. Our requirement analysis ensues on a presentation of the various possible dimensions of integration. Towards these dimensions, we present five case studies stemming from the French community. Based on theses case studies, handling interoperability and integration issues, we illustrate the set of problems collectively studied within the "Action Spécifique Plateformes" group. We conclude by proposing a guide of good practices for integration.

■ KEYWORDS : guidelines, component-based learning, integration platform, interoperability, case studies

1. Introduction

La recherche sur les Environnements Informatiques d'Apprentissage Humain (EIAH) a produit des Logiciels Educatifs Intelligents et Interactifs pour l'enseignant et l'apprenant. Ces logiciels, communément appelés EIAH (comme le domaine de recherche) implantent des fonctionnalités variées. Par exemple, "l'édition de figure géométrique" et "l'assistance dans la rédaction d'une preuve" sont des fonctionnalités actuellement implantées dans des EIAH de géométrie. Une fonctionnalité aide l'apprenant dans la réalisation d'une activité d'apprentissage. Elle correspond à un besoin : manipuler une figure, consulter un rappel de cours... Ces fonctionnalités sont aussi des "actions élémentaires" qu'un enseignant peut identifier et prescrire dans une activité pédagogique.

La synthèse des travaux présentés dans cet article est issue de collaborations et de constats réalisés dans le cadre de l'Action Spécifique plate-forme visant au support d'une communauté de recherche en EIAH (SiteASPF, 2005). Ces travaux partent du constat suivant : peu d'EIAH quittent les laboratoires pour une utilisation effective dans les classes. Outre les raisons liées à la nature expérimentale des approches ou des moyens matériels mis en œuvre, la raison la plus importante pour notre propos est que chaque logiciel ne couvre souvent qu'une partie des fonctionnalités nécessaires. Dans ces conditions, il n'est aisé ni à l'enseignant de proposer une activité pédagogique impliquant plusieurs fonctionnalités complémentaires, ni à l'apprenant de repérer quelle fonctionnalité de quel logiciel utiliser dans les tâches qu'il doit accomplir. Les chercheurs rencontrent les mêmes difficultés pour évaluer les logiciels, expérimenter une nouvelle fonctionnalité ou encore en observer les usages.

Pour remédier aux difficultés des chercheurs et dans un objectif de capitalisation des efforts et de réutilisation, nous aimerions qu'ils puissent employer un ensemble de fonctionnalités déjà implantées dans différents logiciels afin de tirer partie de leurs fonctionnalités complémentaires. Or, ces logiciels n'ont souvent été conçus ni pour être intégrés ni pour coopérer, ils doivent donc être adaptés. On aimerait alors pouvoir considérer chaque logiciel voire chaque fonctionnalité comme un composant. Ainsi "il suffirait" d'assembler des composants pour obtenir un logiciel qui ait les fonctionnalités souhaitées. De nombreux auteurs privilégient cette approche en proposant des architectures à base de composants.

Notre but n'est pas de spécifier et concevoir des composants éducatifs génériques mais de réutiliser des composants existants afin de les intégrer dans des dispositifs d'apprentissage. En effet, la réutilisation se définit comme une nouvelle approche de l'ingénierie des systèmes selon laquelle il est possible de développer un système à partir de composants existants. Cette approche s'oppose aux démarches usuelles de développement dans lesquelles la construction d'un nouveau système part de rien ("from scratch") et nécessite de réinventer tout à chaque fois (Barbier et Al, 2002). En génie logiciel, il existe deux démarches quant à la réutilisation dans l'étape de conception : on parle de "design for reuse" (a) et "design by reuse" (b). La stratégie (a) concerne le concepteur de composants qui contribue à bâtir une bibliothèque de composants. Elle s'intéresse plutôt à un niveau d'interopérabilité sémantique. La stratégie (b) concerne l'ingénieur d'applications qui réutilise des "composants" existants pour bâtir son application. Elle s'intéresse plutôt à un niveau d'interopérabilité du code. Nous nous sommes centrés dans cet article sur la démarche (b) en tentant de définir des bonnes pratiques pour que les prochains développements en EIAH tendent vers la démarche (a).

Nous proposons d'aborder cette problématique à partir de l'étude systématique de la conception du dispositif qu'il faut mettre *autour* et *entre* les logiciels ou composants pour permettre leur utilisation conjointe. Cette utilisation conjointe nécessite d'étudier pour chaque dimension de l'intégration, les types de réification pour chaque composant et les moyens de faire converger ces réifications (section 2). Cette étude est ensuite illustrée par des études de cas dans lesquelles nous avons souligné les problèmes d'interopérabilité et d'intégration d'EIAH hétérogènes ainsi que les dispositifs mis en place. Ces études de cas permettent aussi d'illustrer comment différents auteurs ont résolu le problème de dispositifs autour

et entre les EIAH qu'ils ont utilisés conjointement (section 3). Nous tentons alors d'en tirer des enseignements pour la conception et l'intégration des EIAH.

2. Conception d'un dispositif d'intégration d'EIAH

Précisons le dispositif technique qu'il faut mettre autour des composants existants et entre ceux-ci pour permettre leur intégration dans la réalisation d'une activité. Dans sa caractérisation technologique, un EIAH est un artefact informatique, support d'une activité pédagogique avec l'intention de faire apprendre. Nous commençons par poser le problème et quelques définitions (section 2.1). Puis nous proposons d'identifier les propriétés nécessaires aux composants et les fonctionnalités nécessaires au dispositif (section 2.2).

2.1 Position du problème et définitions

Le dispositif est conçu pour une activité impliquant des utilisateurs. Nous décrivons d'abord les problèmes posés aux utilisateurs lors de l'expérimentation. Puis nous définissons notre vocabulaire avant de décrire le concept d'intégration. Enfin nous expliquons les dimensions que nous allons explorer dans la section suivante.

2.1.1 Description de problèmes posés lors de l'utilisation de différents EIAH au sein d'une même activité.

Dans (Rosselle et Granbastien, 2004), l'illustration d'une situation mettant en œuvre un utilisateur désirant réaliser une activité – résolution d'un exercice de géométrie – sans référence à un dispositif montre qu'il est très difficile de:

- choisir le composant qui dispose des fonctionnalités adéquates,
- passer d'un composant à un autre pour l'utilisateur de l'expérimentation,
- connaître les modalités de cet environnement non intégré quant aux interactions possibles entre composants,
- réutiliser cette situation non formalisée,
- caractériser et évaluer cette expérimentation.

Par exemple, de manière plus pragmatique, sans dispositif, l'utilisateur doit "jongler" entre différents EIAH. Il est également possible que ces derniers s'appuient sur des fonctionnalités système courantes, communes ou obligatoires (identification, authentification, espace de travail personnel, historique) qui s'avèreront redondantes, incompatibles ou tout simplement peu pratiques. La mise en œuvre et la cohabitation des différents éléments resteront à la charge d'un utilisateur pas forcément à même d'en saisir les subtilités techniques ou de conception. L'intégration dans un dispositif doit permettre de mieux prendre en compte ces problèmes afin de réduire ces inconvénients.

2.1.2 Définitions

Définissons le rôle des utilisateurs du dispositif, l'activité menée avec le dispositif et l'intégration du dispositif dans un contexte plus général.

Les rôles

Les dispositifs d'expérimentation des EIAH impliquent divers utilisateurs entre autres des apprenants, des tuteurs, des enseignants-auteurs, des enseignants prescripteurs, des pédagogues concepteurs, des ingénieurs pédagogiques, des ingénieurs informaticiens, des chercheurs en EIAH (sciences de

l'éducation, didactique, etc.) dont nous allons caractériser les rôles. Pour notre étude, nous nous intéressons aux cas des chercheurs en EIAH qui souhaitent réaliser une expérimentation. D'où, nous appelons *prescripteur* le rôle de l'utilisateur qui prescrit l'activité (l'expérimentation). C'est un chercheur en EIAH. Nous appelons *sujet* le rôle de l'utilisateur qui réalise l'activité. C'est un apprenant, un tuteur ... Nous appelons *administrateur* le rôle de l'utilisateur qui prend en charge tous les aspects techniques du dispositif. C'est un chercheur en EIAH, un ingénieur en informatique ... Il règle la configuration et les paramètres du dispositif technique en fonction des caractéristiques matérielles : nombre et types de machines, type de réseau, composants en présence. Pour (ASPF, 2005), il est un "architecte d'environnement d'apprentissage" qui assemble, intègre les divers composants. Dans cet article, nous nous focalisons sur le rôle de l'administrateur. Nous mettons en évidence les problèmes qu'il rencontre et qu'il doit résoudre pour rendre l'expérimentation possible.

L'expérimentation

Pendant l'expérimentation, le prescripteur demande au sujet de réaliser une activité. Quel type d'activité le dispositif permet-il de réaliser ?

L'activité peut être une activité dirigée. Ainsi, le prescripteur définit les étapes successives par lesquelles le sujet doit passer. Nous considérons qu'à un instant donné, le sujet n'utilise qu'un seul composant. En effet, même s'il utilise à la fois un éditeur de texte et un éditeur de figure, il passe de l'un à l'autre pour soit entrer des caractères, soit manipuler la figure. Pour chaque étape, le prescripteur définit une consigne, une fonctionnalité à utiliser et le logiciel/composant dans lequel cette fonctionnalité est offerte (elle peut être proposée par plusieurs logiciels). Des boucles peuvent être définies. Par exemple, tant que la figure géométrique n'est pas validée positivement, faire "éditer la figure" (étape n), la transition d'une étape à l'autre est franchie à la demande du sujet ou déclenchée automatiquement.

De la même manière, nous pouvons utiliser le dispositif pour réaliser une activité à postes *i.e.* une activité où le sujet a à sa disposition des postes (fonctionnalités) par lesquels il peut passer dans l'ordre qu'il choisit pour atteindre un but donné. Ainsi le sujet peut disposer de toutes les fonctionnalités disponibles de tous les composants. Afin de réaliser un objectif pédagogique, « rédiger une note » par exemple, le sujet utilise la fonctionnalité qui lui sied. Cependant, le scénario pédagogique peut contraindre l'accès à telle ou telle fonctionnalité. Ce type d'activité est propice au déroulement de projets. De même, le prescripteur peut ne pas avoir anticipé le besoin d'une fonctionnalité particulière. Ainsi le sujet, pour répondre à ce besoin, pourrait aller à la recherche du composant souhaité et l'ajouter au dispositif (seul ou avec l'aide de l'administrateur).

Le dispositif doit supporter tout type d'activités pédagogiques, individuelles, collectives, coopératives ou encore collaboratives. Dans la conception du dispositif technique, une composante doit permettre d'évaluer et de réguler l'activité.

En EIAH, le concept de scénario recouvre des réalités différentes. Il semble qu'il soit indispensable d'adjoindre un qualificatif afin de mieux préciser le contexte. Nous rencontrons par exemple dans la littérature (Brassard et Daele, 2003), (Pernin et Lejeune, 2004), (Guéraud et Al, 2004), les expressions de scénarios pédagogiques, scénarios d'apprentissage, scénarios d'usage en classe, scénarios ciblés en tant que pré-composant, scénarios en tant que contrôle de l'activité de l'élève, scénarios didactiques, scénarios de navigation, scénarios de communication, scénarios d'activité, scénarios d'utilisation, scénarios d'interaction, scénarios relatifs aux contenus et scénarios de référence... Il est à noter que même ainsi contextualisés, les définitions restent multiples. A l'instar de (Brassard et Daele, 2003), nous avons choisi de définir simplement un scénario pédagogique comme « le résultat du processus de conception d'une activité d'apprentissage », processus s'inscrivant dans un temps donné et aboutissant à la mise en oeuvre du scénario.

Les activités dirigées et les activités à postes sont une partie des activités. Suivant les types de

composants impliqués (exerciseurs, tutoriels, micromondes, simulateurs), les types de pédagogie qui peuvent être mis en œuvre semblent nombreux. Cependant, des limitations pourraient être mises en évidence par des praticiens, des enseignants, des pédagogues ou des didacticiens. Nous ne maîtrisons pas en effet, les présupposés pédagogiques que notre approche présume.

Définition d'un dispositif d'intégration d'EIAHs et contexte général

Qu'entend-on par dispositif d'intégration d'EIAH ? G. Jacquinot et L. Monnoyer (Jacquinot et Monnoyer, 2002) donnent la définition suivante du terme de "dispositif" : il s'agit d'"un ensemble de pièces constituant un mécanisme, un appareil quelconque" et donc de manière plus générale : "tout agencement d'éléments humains ou matériels, réalisé en fonction d'un but à atteindre". Nous pouvons donc le définir comme un support médiateur permettant d'atteindre un objectif. Dans les travaux présentés, le dispositif vise à l'intégration de composants logiciels particuliers, des EIAH, afin de construire un nouveau « dispositif » qualifié de dispositif d'expérimentation. Le dispositif qui permet de réaliser une expérimentation peut être un dispositif technique isolé sur un ordinateur individuel ou sur un ensemble d'ordinateurs en réseaux. Il peut aussi être accessible au travers d'un LMS (Learning Management System), un CMS (un Content Management System) ou un C3MS (Community, Content, and Collaboration Management System).

2.1.3 Description du dispositif et de son architecture

Le dispositif est un environnement informatique dans lequel différents composants logiciels sont mis à la disposition d'un sujet. Chaque composant peut tourner sur une machine distincte de celle sur laquelle le sujet réalise son activité. Par conséquent, l'architecture du dispositif est distribuée. Certaines fonctions sont offertes par le dispositif : *e.g.* supporter l'interaction entre composants, le partage de ressources et l'échange de données. De plus, certaines propriétés permettent aux composants d'être utilisés par le dispositif : *e.g.* une capacité à interagir et à échanger des données.

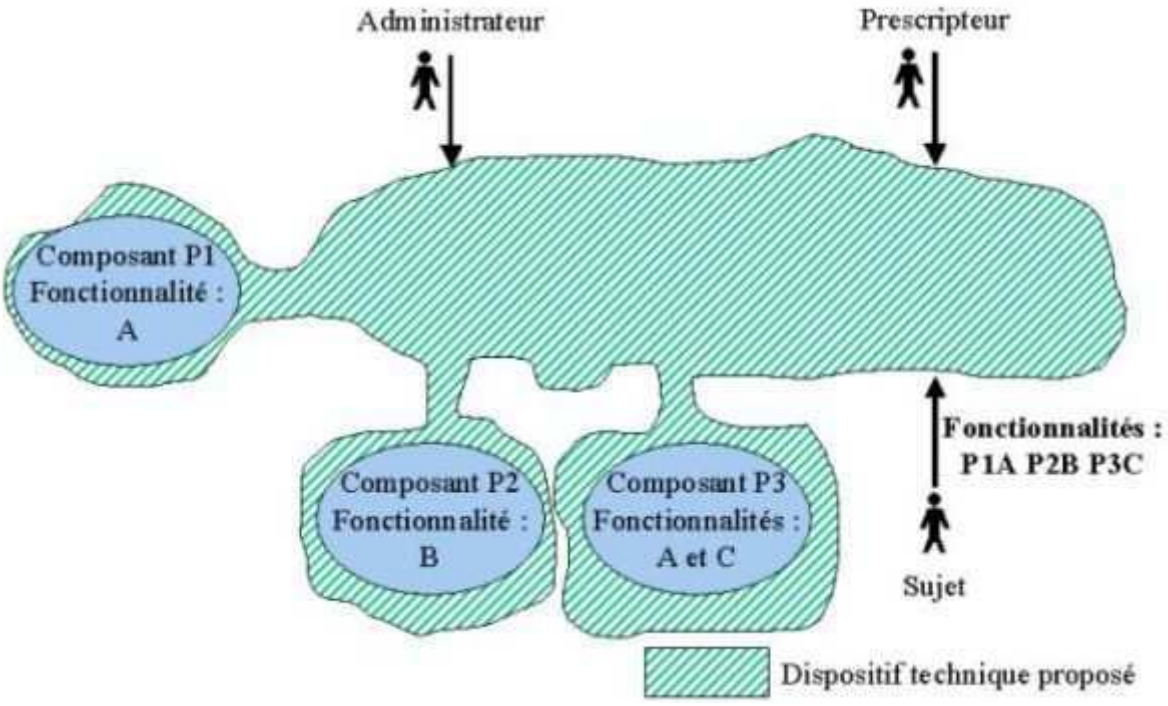

Figure 1 : Le dispositif technique proposé

Sur la figure 1, nous avons, par exemple, trois composants P1, P2 et P3, fournissant les fonctionnalités A,

B et C. L'objectif est d'offrir au sujet un "EIAH" dont les fonctionnalités sont toutes celles nécessaires. Autrement dit, tout se passe comme s'il avait sur son ordinateur toutes les fonctionnalités choisies dans les différents composants : ici la fonctionnalité 'A' de P1, la fonctionnalité 'B' de P2 et la fonctionnalité 'C' de P3. Ainsi, si l'intégration est réussie, le sujet n'a pas conscience d'utiliser les trois composants.

La problématique de l'intégration de composants hétérogènes apparaît ici clairement. Sa mise en œuvre doit être envisagée de manière globale. Pour gérer la communication et les autres aspects de l'utilisation conjointe des composants P1, P2, P3, nous définissons un dispositif technique. Pour ne pas présager de la technologie adoptée, nous avons choisi une forme quelconque pour représenter ce dernier. Nous verrons par la suite dans les études de cas, qu'il peut être un dispositif technique *ad hoc*, utiliser les modèles de composant CORBA, J2EE et EJB, les web services, *etc.* Il porte des noms aussi variés que les technologies employées (routeur, médiateur, agent facilitateur, service de nommage, service de type pages jaunes, middleware). Dans la suite, nous appelons "patatoïde" la partie du dispositif technique qui est entre les composants. Par ailleurs, les composants P1, P2 et P3 ayant été conçus indépendamment dans des technologies indépendantes, il faut prévoir une adaptation de ceux-ci pour qu'ils puissent coopérer entre eux par l'intermédiaire de la patatoïde. Nous appelons "glu", la partie du dispositif technique qui est autour des composants pour leur permettre d'être utilisés conjointement via la patatoïde.

Mais que signifie au juste l'intégration ? Nous pouvons rencontrer dans la littérature des utilisations du terme intégration dans des sens différents. C'est pourquoi, nous allons préciser maintenant la définition de ce terme.

2.1.4 Qu'entendons-nous par "Intégration" ?

Notre définition repose à la fois sur les types d'intégration et sur les dimensions de celle-ci.

Les types d'intégration

Précisons les modalités de l'intégration des composants. Notre utilisation du mot "intégration" a le sens "d'utilisation conjointe de différents outils dans un environnement donné" (plate-forme, dispositif). A partir de l'étude de l'existant, (Rosselle et Granbastien, 2004) identifient quatre types d'utilisation conjointe : la juxtaposition, la transmission, la coopération et l'interopération. Dans la littérature, on rencontre également les termes d'assemblage ou de composition de composants. Ainsi dans la partie 3, nous essayerons d'identifier quel est le type d'intégration obtenu.

Les dimensions de l'intégration

Lors de l'intégration, différents types d'intégration peuvent être envisagés simultanément. Nous pouvons par exemple rencontrer une intégration de type interopération pour l'activité à réaliser (le sujet est guidé pour l'activité sans devoir passer d'un composant à l'autre par l'activation volontaire d'un composant) alors que l'intégration pour l'interface n'est que la juxtaposition des interfaces graphiques des différents composants. Ces différents niveaux sont appelés "dimensions". En Génie logiciel (GL), trois dimensions de base sont généralement identifiées quels que soient les auteurs : présentation, données, contrôle. Pour d'autres auteurs, les dimensions de plate-forme, de processus et de communication sont ajoutées (Wicks, 2004). L'intégration de la **présentation** assure que les outils disposent d'interfaces cohérentes (i.e. leur "look and feel" est commun). Concernant les **données**, les outils se communiquent les données à partager, via un système (appelé "dépôt") qui fait interface entre eux. Pour le **contrôle**, les outils se notifient des événements pour se coordonner. Dans la dimension **plate-forme,** les outils sont exécutés sur le même environnement d'exploitation virtuel (i.e. les réseaux et les systèmes d'exploitation sont transparents pour les outils). Le **processus**, le développement du système en GL (l'expérimentation pour nous) est géré du début à la fin et les outils pertinents sont associés au bon moment. Enfin la dimension **communication** est celle qui supporte les équipes durant le développement du système.

Les technologies évoluant, nous n'avons – en théorie – plus besoin de nous préoccuper de savoir sur quel

système d'exploitation (plate-forme) a été initialement développé le logiciel ou le composant à intégrer. Les technologies qui permettent cela sont nombreuses. Par exemple, une application ou une applet développée en java ne pose pas le problème du système d'exploitation si elle n'appelle pas de fonction spécifique au système. Ou encore, pour une application client/serveur, il est possible de ne pas connaître le système d'exploitation sur lequel tourne le serveur. Le client peut être développé pour tout système. Les technologies de web services et les applications développées en trois tiers (au minimum) permettent aussi de s'affranchir de la connaissance des systèmes d'exploitation, en particulier en donnant l'accès aux interfaces utilisateurs via un navigateur web. En pratique, nous pouvons rencontrer des difficultés inattendues (version de la machine virtuelle java, du navigateur, du serveur d'application, etc.). Aussi faut-il être prudent et annoncer dans quel contexte a été développé et utilisé le composant. Par ailleurs, nous parlons plutôt de la dimension "gestion des interfaces" de l'utilisateur plutôt que de la dimension présentation : l'objectif étant de garder, de réutiliser l'existant plutôt qu'adapter des logiciels sur des formats pour lesquels ils n'ont pas forcément été conçus. Enfin, la spécificité des EIAH nécessite l'étude de dimensions plus fines. En particulier, l'indexation des EIAH et la communication des données et des modèles (*e.g.* d'apprenant) prennent une grande importance (*cf.* (Rosselle et Granbastien, 2004)). En effet, afin de réutiliser des composants EIAH avec une haute qualité de service, il est important d'avoir des mécanismes d'indexation permettant aux intégrateurs de trouver aisément le composant adéquat et les fonctionnalités qu'il rend. Ainsi, dans la dimension "contrôle", nous distinguons d'une part l'"indexation des composants" et d'autre part la "commande de plusieurs composants". De même dans la dimension "données" nous distinguons d'une part la "communication des données" entre les composants et d'autre part la "collecte des données" résultat de l'activité menée avec le dispositif. Enfin, le "processus" est une expérimentation impliquant des composants complémentaires et pour cela il est nécessaire de définir un "protocole de coopération des composants". En résumé, dans la suite nous étudierons les dimensions : *indexation des composants, communication de données, commande de plusieurs composants, protocole de coopération pour les composants, gestion des interfaces et collecte des données*. Ces dimensions nous semblent intéressantes pour systématiser l'étude, en se posant des questions spécifiques pour chacune lors de l'intégration des EIAH. Qu'a-t-on en commun comparé à ce que l'on a de différent ? Que partage-t-on ? Que se communique-t-on ?

2.2 Les enjeux dans chaque dimension de l'intégration dans le cas général

L'intégration de composants ne se fait pas sans résoudre un certain nombre de difficultés. Ces difficultés sont regroupées selon les dimensions rappelées ci-dessus. L'étude de chacune de ces dimensions dans (Rosselle et Granbastien, 2004) débouche en premier lieu sur la définition de 7 propriétés demandées aux composants. Nous les redonnons ci-dessous. Le composant doit :

- permettre l'observation de certains de ses mécanismes, états, objets etc. (il est *inspectable*)

- fournir les traces intelligibles de l'interaction de l'utilisateur avec lui (il est *traçable)*;

- permettre une certaine forme de prise de contrôle, comme pouvoir être rendu actif ou inactif, totalement ou partiellement et, par ailleurs, permettre de collecter uniquement les interactions jugées utiles (il est *scriptable)* ;

- pouvoir être décrit afin d'être retrouvé et de proposer ses fonctionnalités (il est *indexable)*;

- pouvoir exporter son interface graphique, permettre son adaptation ou sa reprogrammation (il est *intégrable au niveau IHM* ou *plastique* (Calvary et Coutaz, 2002)*)*;

- permettre d'accéder indépendamment à ses différentes fonctionnalités (il est

décomposable en *fonctionnalités indépendantes ou élémentaires)*;

- assurer la communication des données dans un format adéquat (il utilise des formats normalisés, des standards et des spécifications pour lesquelles il y a un consensus).

Cette étude débouche ensuite sur des fonctions identifiées pour le dispositif afin de favoriser la communication et la coopération inter-logicielle. Ces fonctions sont :

- posséder les mêmes propriétés qu'un composant,
- posséder un moteur pour définir et exécuter un protocole de coopération pour les prototypes,
- posséder un mécanisme pour scruter des états ou des variables d'un prototype,
- implanter un langage de commande des prototypes,
- gérer les formats de données,
- assurer la communication des données,
- gérer une base d'informations sur les composants pour indexer leurs fonctionnalités intéressantes,
- posséder un mécanisme pour sélectionner les interfaces pertinentes.

Pour les logiciels et composants existants, il faut se demander ce qu'il faut leur ajouter pour les doter de ces propriétés et des fonctionnalités nécessaires. Des solutions dépendantes des évolutions techniques existent. Les composants et dispositifs futurs devraient prendre ces recommandations en compte dès leur spécification.

Enfin, cette étude a mis en perspective des pistes de recherche nécessaires dans chaque dimension. Nous les énonçons à la section 2.3.

2.3 Les verrous scientifiques liés à chaque dimension de l'intégration

Dans cette section nous reprenons les verrous scientifiques liés à chaque dimension.

2.3.1 Indexation des composants

Les verrous liés à cette dimension concernent deux aspects : d'une part, la normalisation de la description d'un composant avec des méta-données et d'autre part l'aspect "génie logiciel" *i.e.* les technologies proprement dites permettant d'atteindre les composants dans le dispositif. Cette description est souvent *ad hoc* car elle reste actuellement difficile à prendre en compte dans les normes existantes ou en cours d'élaboration. La recherche sur les méta-données produit des modèles pour décrire (et indexer) une ressource pédagogique, plus particulièrement un composant, afin de la retrouver et de faire appel à ses fonctionnalités. Le format de description doit permettre de décrire à la fois les aspects techniques et les considérations liées à l'usage pédagogique du composant (Grandbastien et Al, 2002), (Rebaï et Labat, 2004). Pour l'aspect génie logiciel, plusieurs technologies permettent aux composants d'afficher leurs fonctionnalités (*e.g.* l'interface IDL en CORBA) ou encore d'en connaître les fonctionnalités disponibles (*e.g.* méthode List_initial_services de l'ORB).

2.3.2 Communication des données

Deux verrous sont également identifiés : d'une part la gestion des formats de données entre différents composants et d'autre part la gestion des données proprement dites. Pour *la gestion des formats de données*, deux possibilités sont envisageables. Premièrement, il existe un format standard de représentation des données. Deuxièmement, il existe un moyen de traduire les connaissances à échanger

dans le langage de représentation de chaque composant cible de ces données. Ces deux possibilités peuvent être utilisées conjointement. Elles ne sont pas bien sûr exclusives. Se pose aussi le problème récurrent de la gestion des données. Les données doivent-elles être réparties dans les prototypes qui en ont besoin ? Doivent-elles être stockées dans un dépôt (repository) ? Les technologies actuelles (*e.g.* J2EE, Web services) permettent d'associer les deux choix en fonction de chaque type de données. Le rôle de dépôt de données peut d'ailleurs être joué par l'un des composants du dispositif.

2.3.3 Commande de plusieurs composants

Nous distinguons deux questions primordiales pour cette dimension. Comment définir un ensemble de commandes pour agir sur tous les composants ? Et comment transmettre ces commandes ? La recherche d'un ensemble de commandes a produit un standard (LTSC,2005) pour agir sur un composant de l'extérieur. Le but est de pouvoir scripter tous les composants avec le même langage de scripts. Les noms des commandes sont fortement inspirés des commandes d'appelScript (MacOS) et de script de OLE (WindowsOS). De son côté, la transmission des commandes repose sur l'invocation de méthode (RMI, RPC) ou le scriptage des composants (*e.g.* la technologie des web services SOAP/WSDL et des flux XML).

2.3.4 Protocole de coopération pour les composants

Deux points particuliers doivent être abordés dans cette dimension : la définition "d'inspectables" communs à un ensemble de composants et la définition d'un protocole de coopération entre les composants. En effet, il faut dire quel prototype est lancé, où, avec quelle configuration (interface), quels paramètres. Que reçoit-il en entrée, que transmet-il en sortie, émet-il des événements ? Ce protocole pour spécifier le déroulement de l'activité peut aussi utiliser IMS-LD (IMSLD, 2005) ou utiliser un moteur de workflow.

2.3.5 Gestion des interfaces

De nombreux travaux ont montré l'importance de l'IHM et la spécificité de cette gestion en EIAH, en particulier pour l'acceptabilité d'un EIAH. Elle nécessite des concepts, des langages pour exporter ou re-programmer les interfaces d'un logiciel. L'intégration à ce niveau peut être facilitée par le respect de bonnes pratiques liées à la plasticité (Calvary et Al, 2003) ou de modèles de conception (design patterns).

2.3.6 Collecte de données pour le suivi et la régulation de l'activité

Cette dimension est un objectif principal du dispositif. Les fonctionnalités de suivi et de régulation dans l'apprentissage sont primordiales. Ainsi, il faut être capable premièrement de définir des traces pertinentes à sauvegarder dans le but de collecter des données exploitables sur l'activité, deuxièmement d'acheminer ces traces et troisièmement de les exploiter. Des concepts, des langages et des outils pour la collecte de traces d'interactions didactiques pertinentes et de granularité adéquate doivent être développés. La transmission des données entre les composants peut s'effectuer grâce aux mécanismes d'abonnement aux flux d'événements (*e.g.* dans CORBA et J2EE).

2.4 Discussion

La section 2.3 montre que les questions à se poser sont nombreuses et que des solutions existent (spécification, standard, normes, technologies). Cependant nous pouvons nous questionner sur nos choix. Comment faire un bon choix ? Répondre à cette question nécessite encore de nombreuses recherches. Néanmoins, que pouvons-nous retenir pour la recherche en EIAH ? A court terme, intégrer des EIAH consiste à comparer ces derniers suivant chaque dimension et à tenter de faire converger les choix faits sur chaque dimension. Par exemple, si pour décrire le protocole de coopération de deux EIAH, l'un implante IMS-LD et l'autre utilise un moteur de workflow, nous pouvons par exemple opter dans le dispositif pour la spécification IMS-LD et adapter l'interface entre le dispositif et le moteur de workflow

(*e.g.* verrue, composant externe), parce qu'il faut bien faire un choix pour expérimenter. Ce choix pragmatique n'est pas forcément idéal et pose de nombreuses questions du point de vue de la recherche en EIAH (cf. (EIAH, 2005). Cependant, ce choix est *stratégiquement* le meilleur compromis en attendant que notre communauté amende ces spécifications ou que nos recherches fassent émerger un consensus. Cet exemple illustre un type de difficultés que nous rencontrons dans les autres dimensions. Prendre conscience que le choix fait est stratégique permet d'identifier les limites de nos choix. En particulier, nous recommandons l'utilisation des spécifications, des normes, des standards et des technologies leaders, tout en prenant la distance qui s'impose.

3. Etudes de cas

Nous présentons maintenant des études de cas pour lesquelles nous décrivons le dispositif et pour chaque dimension les choix faits et les types d'intégration. Les deux premières études sont détaillées car elles résultent des recherches des auteurs. En revanche, les suivantes reposent uniquement sur l'étude de la bibliographie et sont par conséquent moins fournies.

3.1 Atelier d'intégration d'EIAHs de géométrie

L'atelier d'intégration d'EIAH de géométrie (Rosselle, 2001) (Rosselle et Granbastien, 2004) a pour but de permettre l'utilisation conjointe de plusieurs EIAH ayant des fonctionnalités complémentaires en géométrie. Ces composants sont au départ des prototypes de logiciels issus de la recherche en EIAH. Ils n'ont pas été prévus pour être intégrés.

3.1.1 Description

Le dispositif est appelé atelier, par analogie avec les ateliers de génie logiciel, avec comme 'processus' '*l'activité'*, et comme 'fonctionnalités' (compilation, édition, spécification, en génie logiciel) '*les fonctionnalités disponibles en géométrie'* (e. g. édition de figure). La figure 2 présente l'architecture du dispositif. Par rapport à l'architecture présentée à la figure 1, chaque logiciel est incorporé dans un composant (glu) avec une granularité plus ou moins grande suivant le degré de contrôle que permet le logiciel. La patatoïde est divisée en deux équipements : "médiateur" qui sert d'intermédiaire (partie hachurée) entre les différents composants impliqués et le "gestionnaire d''activité" (dans l'hexagone). Les fonctions du dispositif recensées à la section 2.2 sont réparties entre ces deux composants. Le médiateur prend en charge les fonctions d'indexation des composants, de communication de données et de commande de plusieurs composants de la patatoïde. Il incorpore autour de l'ORB, des composants qui remplissent des services comme la gestion des formats. Le gestionnaire d'activité prend en charge le protocole de coopération pour les composants, la gestion des interfaces et la collecte des données. Il est lui-même un composant CORBA. Les flèches sur cette figure correspondent à une mise en relation entre les divers éléments (interfaces de communication de chaque côté). Pour valider cette approche, le médiateur et le gestionnaire d'activité ont été implantés avec les technologies CORBA 2.0 et java 1.2, où chaque logiciel était inclus dans un composant CORBA ayant les "bonnes propriétés".

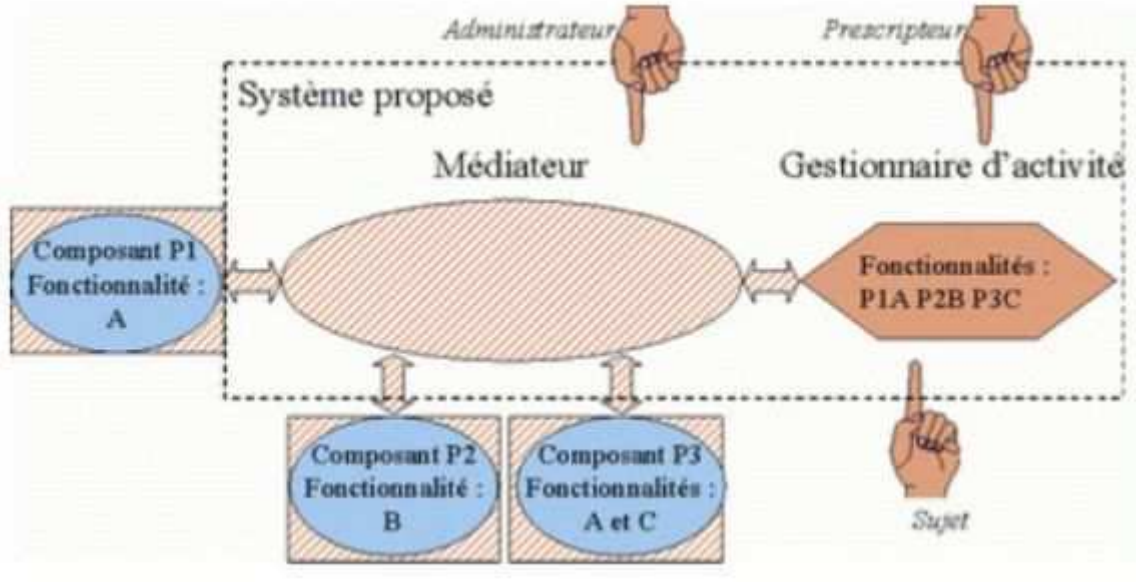

Figure 2 : Le système proposé

3.1.2 Choix adoptés pour l'intégration

Pour *l'indexation des composants*, la contribution de cette étude de cas consiste à recenser les fonctionnalités liées à l'activité en résolution de problèmes de géométrie en classe de 4^{iéme}. La liste (Rosselle, 2001) obtenue sert à décrire les composants de géométrie pour les indexer dans le dispositif. Cette liste reste actuellement difficile à prendre en compte dans les normes existantes ou en cours d'élaboration. Par conséquent, les auteurs ont utilisé le champ cinq (educational) de la LOM (LTSC, 2005). Concrètement, ils ont utilisé une structure de donnée *ad hoc*, basée sur XML, pour décrire leurs méta-données. L'indexation proprement dite est réalisée pour chaque composant CORBA via la définition de méthodes dans l'interface IDL du composant. Le nommage des méthodes prend en compte les méta-données définies. Du côté du gestionnaire d'activité, ils utilisent la méthode "List_initial_services" du BOA (Basic Object Adapter) de l'ORB en invocation dynamique pour connaître les fonctionnalités disponibles.

Pour la *communication des données*, afin d'éviter la saisie multiple des connaissances du domaine, la contribution des auteurs consiste en la définition d'un interprète généraliste. Il permet de traduire les connaissances d'une représentation vers une autre à l'aide de macro-définitions. Ce composant est appelé quand une traduction est nécessaire. Il est accolé à la patatoïde comme le sont les composants CORBA incluant les EIAH.

Pour la *commande de plusieurs composants*, la contribution consiste d'une part en une étude de faisabilité via l'application du principe "défaire et refaire". D'autre part, les auteurs de l'étude ont défini un ensemble d'ordres, reposant sur (LTSC, 2005). Ces commandes correspondent à des méthodes déclarées dans l'IDL des composants.

Pour le *protocole de coopération* pour les composants, la contribution des auteurs consiste à expliquer d'une part comment rendre scriptable un composant et rendre inspectables des variables ou objets et d'autre part à définir le protocole via un scénario d'activité implanté en java dans un gestionnaire de scénario côté gestionnaire d'activité. Pour les auteurs, un scénario d'activité est la formalisation informatique d'une trame d'activité. Pour expliciter la trame, le prescripteur identifie des tâches que le sujet doit réaliser. Il précise comment les systèmes informatiques peuvent aider le sujet comme par exemple : "le sujet dessine la figure correspondant à l'énoncé avec Cabri. Puis ...". La structure de donnée pour décrire le scénario est une structure XML *ad hoc*. Le gestionnaire de scénarios, permettant de les

décrire et de les exécuter est un composant.

Gestion des interfaces

Pour la *gestion des interfaces*, le domaine d'application étant la géométrie, les auteurs ont privilégié l'affichage à l'écran. Ils ont adopté une solution *ad hoc* (utilisant VNC - Virtual Network Computing) afin de permettre le test des diverses propositions qu'ils ont faites. L'intégration est du type juxtaposition des interfaces (dans le temps et dans l'espace). Leur contribution ici consiste à définir précisément la spécification du gestionnaire d'interface graphique, dont ils ont besoin. Il faut bien sûr étudier la validité de leurs propositions et leur généralisation à d'autres IHM.

Pour la *collecte de données*, leur contribution consiste à proposer un mécanisme pour filtrer et structurer des traces d'interaction. Dans ce but, ils réutilisent l'interprète construit pour la gestion des connaissances du domaine avec des macro-définitions en "observables". Ils définissent pour cela des macro-textes pour chaque composant.

3.1.3 Conclusion

Cette recherche a spécialement consisté à débroussailler le terrain pour déterminer quels étaient les problèmes à résoudre. Les auteurs ne prétendent pas présenter un dispositif qui résolve définitivement les problèmes soulevés lors de sa conception. Cependant, en identifiant les directions de recherche, ils formulent ici les problèmes qui se sont posés, en restant aussi indépendants que possible des technologies d'implantation. Ils ont eu une approche pragmatique afin de montrer la faisabilité de leur approche et la valider. Aujourd'hui le contexte technologique est quelque peu différent, de nouvelles approches et de nouveaux services sont apparus. En particulier, l'architecture choisie a pu être conservée dans l'évolution de CORBA vers J2EE. Elle semble pouvoir être préservée dans les évolutions futures. Par ailleurs, l'indexation des composants (*ad hoc* ici) pourrait bénéficier des modèles du web sémantique (les langages XML et RDF et les outils relatifs à ces langages) ainsi que des recherches en cours visant la définition d'une ontologie des termes que nous utilisons pour décrire une activité pédagogique (TCAN-OURAL (OURAL, 2005). De plus, le protocole de coopération pour les composants (scénario d'activité *ad hoc* ici) devrait bénéficier des apports des recherches sur les workflow ou sur les langages tels que IMS-LD.

3.2 Le dispositif EVACOLEN

Ce travail de recherche (Bessagnet et Al, 2005) prend comme exemple l'intégration de deux applications éducatives :

- un système de support de collaboration intelligent (SSCI) - Intelligent Collaborative Support System (ICSS) – qui gère entre autres les activités de dialogue et de négociation. Il permet d'évaluer les habiletés de collaboration (Aiken et Al, 2005).

- un espace d'activités partagées, le système open source Tulka WhiteBoard (Tulka, 2005). Il permet d'évaluer les habiletés de résolutions de problèmes.

En intégrant ces deux systèmes, les auteurs construisent un environnement qui permet aux étudiants aussi bien de collaborer sur une activité d'apprentissage spécifique que de recevoir un feedback sur cette collaboration pendant qu'ils travaillent ensemble.

3.2.1 Description

Le dispositif qui permet l'intégration de SSCI et de Tulka WhiteBoard peut-être considéré comme un atelier. Afin de montrer la faisabilité de leur démarche, les auteurs ont conçu un premier prototype qui résulte de plusieurs choix dont les principaux sont décrits ci-dessous. Cet environnement est appelé EVACOLEN pour Evaluation de la Collaboration en Ligne (Aiken et Al, 2005). La figure 3 basée sur des packages UML montre l'environnement complet. La méthode d'intégration choisie est la composition des

deux environnements.

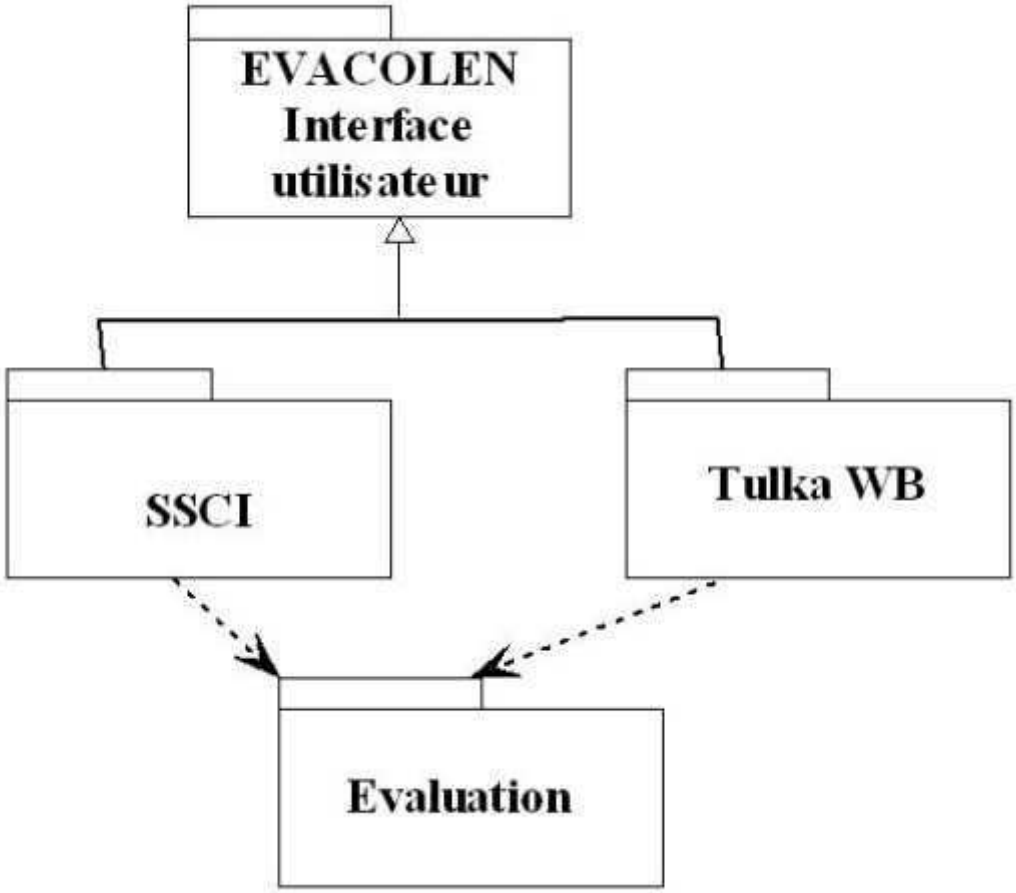

Figure 3. Les packages de l'environnement

Afin d'intégrer les deux environnements, les auteurs ont fait les choix d'implémentation décrits ci-après.

Choix 1 : 1 ou 2 applets pour l'interface homme machine ?

Afin d'intégrer les 2 environnements, les auteurs pouvaient opter pour deux possibilités :

- Soit à l'aide du navigateur Internet qui télécharge deux applets java correspondant aux clients des deux environnements. La communication entre celles-ci se fait via des variables globales déclarées en java script ou à l'aide de messages échangés entre les deux serveurs.

- Soit en développant une interface graphique unique regroupant les deux environnements clients. La communication entre ceux-ci se fait via des procédures et fonctions à l'intérieur du code java regroupant les deux interfaces.

La première implémentation comporte le risque que l'utilisateur se retrouve avec la moitié de l'environnement. En effet, il suffit que l'une des deux applets ne soit pas disponible pour que la partie communication ainsi que l'environnement global soit inutilisable. Dans la deuxième implémentation, la communication entre les deux environnements se fait à travers une applet unique. Ainsi, l'utilisateur a la garantie d'avoir la totalité de l'environnement. Les auteurs ont donc décidé d'implémenter une interface unique intégrant les deux applications.

Choix 2 : un ou deux middleware pour communiquer ?

Afin de gérer la communication entre les deux applications, les auteurs ont privilégié une approche à

deux Middleware séparés et donc créés à l'origine avec l'interface graphique commune servant de tunnel de communication. En effet, une approche « un middleware intégrant la communication des 2 applications » obligeait à écrire complètement une nouvelle couche middleware, pour la communication des deux environnements. Cette solution implique des temps de développement importants.

Choix 3 : intégration des parties clientes

Concernant l'intégration des parties clientes des deux applications, après avoir étudié les codes sources des 2 environnements, deux implémentations possibles s'offraient aux auteurs. La classe **TulkaWhiteBoard** hérite de "**Frame**" dans le package TulkaWB, la classe **MICLSClient** hérite de "**MessageCollaborator**" dans le package SSCI. La deuxième option a été choisie ; en redéfinissant la méthode *notify(message)* de la classe **MessageCollaborator.**

Comment les composants sont-ils réutilisés ?

Ne disposant pas à proprement parler de deux composants logiciels intégrables sur une plate forme J2EE et ne voulant pas transformer ces deux applications en composants, les auteurs se sont orientés vers une composition (Stafford et Wallnau, 2002) des deux environnements, avec une partie cliente et une interface commune.

Qu'est-ce qui constitue la glu ?

Dans ce dispositif, le SSCI est un composant invariant qui doit s'interfacer avec un espace d'activités partagées. La glu est constituée des interactions entre les deux applications qui sont factorisables à d'autres espaces d'activités partagées.

3.2.2 Choix adoptés pour l'intégration

Afin d'intégrer ces deux environnements, les auteurs ont dû régler des problèmes de synchronisation de messages entre les applications, de renommage et de développement de classes Java spécifiques. Toutes les dimensions listées en partie 2 n'ont pas été abordées dans cette étude.

Indexation des composants

Dans le cadre de ces travaux, cette dimension n'a pas été abordée. Cependant, elle est essentielle dans le cadre de la réutilisation de modèles pour les situations pédagogiques particulières abordées : les *situations-problèmes*.

Gestion des interfaces

L'interface graphique commune a amené les auteurs à créer un nouveau type de messages gérant l'interaction entre les deux espaces. En effet, l'accès à l'espace d'activités partagées peut être bloqué par la mise en œuvre de règles pédagogiques analysant les dialogues entre le Group Leader et les utilisateurs. Ainsi l'envoi de messages d'ordre respectant deux règles pédagogiques est permis afin de bloquer ou débloquer le composant Tulka. Le Group Leader peut de cette manière interagir avec la partie Tableau Blanc de chaque utilisateur.

Protocole de coopération

Des difficultés ont été rencontrées lors de l'implémentation

- Un utilisateur peut verrouiller le tableau blanc pour un temps illimité. Les autres utilisateurs ne peuvent alors interagir sur le tableau blanc jusqu'à la libération du tableau.

- Lorsque le Group Leader bloque un utilisateur qui a verrouillé cette ressource, il faut avertir le serveur Tulka que l'utilisateur est bloqué ; faute de quoi, les autres utilisateurs se

retrouveraient en situation de famine sur la partie tableau blanc.

- Les auteurs ont utilisé des conditions booléennes pour réaliser ce contrôle (lors d'un blocage par le Group Leader).

- De même lorsque l'on débloque un utilisateur, il faut savoir si un autre ne verrouille pas le tableau blanc, afin de ne pas créer de conflit pour le serveur Tulka.

Collecte des données

Les fichiers de log issus de l'utilisation de l'environnement et les pré- et post- questionnaires sont analysés par une activité d'évaluation hors-ligne. Cette évaluation est importante dans un tel environnement pour d'une part améliorer l'environnement et d'autre part effectuer un feed-back sur l'activité pédagogique elle-même.

3.2.3 Conclusion

Cette étude EVACOLEN s'appuie sur l'implantation d'une interface graphique intégrant deux outils et deux middleware pour assurer la communication entre les deux outils. Cette approche est généralisable pour intégrer d'autres espaces d'activités partagées au système de support de collaboration intelligent. Cependant, elle requiert des développements *ad hoc* non automatisables pour gérer en fonction du scénario pédagogique, l'accès ou le non-accès dans notre cas au tableau blanc et à certaines de ses fonctionnalités.

L'objectif à terme est de s'intéresser à la conception d'un espace d'activités partagées que les auteurs pourraient spécialiser et facilement personnaliser afin de prendre en compte la multiplicité des espaces partagés. Dans les situations pédagogiques, l'espace d'activités partagées pourrait être : un espace de travail de conception informatique supportant la méthode UML ou encore un modèle Entité-Association, une feuille de calcul électronique, un éditeur de programme (travail collaboratif sur les langages de programmation), des Questionnaires à Choix Multiples, un site Web, un Tableau Blanc tel que démontré dans ce prototype. Il serait très avantageux de pouvoir réutiliser et intégrer aisément dans un tel environnement selon les modalités choisies par le pédagogue, lors de la conception du scénario pédagogique, des composants logiciels existants.

3.3 Intégration avec le moteur de Workflow COW (Cooperative Open Workflow) (Vantroys et Peter, 2003)

Cette étude de cas présente deux types d'expériences d'intégration du moteur de workflow COW, permettant de décrire et de faire exécuter des activités individuelles et collectives : intégration de l'outil collaboratif 3D SPIN avec le moteur COW d'une part et intégration de COW dans deux LMS d'autre part.

3.3.1 Description

Pour l'intégration de l'outil collaboratif 3D SPIN avec le moteur COW, l'approche repose sur une définition des interfaces sous forme de Web Service et la définition des paramètres passés entre les deux outils. Le dispositif permet de relier le moteur de workflow COW en technologie J2EE à un outil externe (SPIN – un client/serveur avec une interface cliente en 3D) via un connecteur. Il y a un SPIN sur chaque poste étudiant.

La seconde expérience d'intégration concerne COW et les LMS "Campus Virtuel" d'Archimed (CVA, 2005) (appelé par la suite CVA) et "OPEN Uss" (OpenUss, 2005) (appelé par la suite OU). Le dispositif pour intégrer COW avec le LMS et CMS CVA repose sur la connexion entre des méthodes Visual Basic du côté CVA et des méthodes java côté COW via SOAP. Le dispositif pour intégrer COW à OU repose sur les EJB et en particulier le serveur d'application Jonas.

3.3.2 Choix adoptés pour l'intégration

Concernant l'intégration de l'outil collaboratif 3D SPIN avec le moteur COW, COW active 3D SPIN lorsque l'utilisateur clique sur une URL. Le contrôle s'effectue par les commandes start et stop. Il lui fournit les paramètres pour installer la situation collaborative de départ, via la création d'une page web avec un lien sur un fichier du bon type mime. SPIN s'active. C'est ensuite une association entre le type mime approprié et l'application SPIN qui provoque l'ouverture de l'outil externe. L'échange d'informations est réalisé via une interface WSDL

Pour le dispositif COW-CVA, les auteurs ont été confrontés aux limitations des technologies de services web (les types ne pouvant pas être traités entre des systèmes hétérogènes). Ils ont réalisé une "sérialisation" des objets manipulés de part et d'autre en offrant une représentation XML des attributs (*i.e.* en envoyant une chaîne comportant la structure de données en XML). Cela simplifie la programmation et permet une intégration "lâche" entre le composant de séquencement et le LMS.

Le dispositif COW-OU semblait plus évident car reposant sur la même technologie. Cependant, il s'est avéré qu'ils n'avaient pas la même version d'EJB. Il a donc fallu lancer deux occurrences de Jonas (une pour chacune des versions compatibles avec les EJB de chacun). La conséquence est qu'il y avait donc deux services de nommage différents. Les auteurs ont pu n'en faire qu'un. Une autre conséquence est qu'il y avait deux services de sécurité aussi. En revanche, la version de ce service avait changé de l'un à l'autre. Là, aucune solution n'a pu être trouvée. Les auteurs envisagent d'essayer l'intégration en SOAP.

3.3.3 Conclusion

Cette étude de cas illustre une intégration d'un outil collaboratif grâce à une interface WSDL et une association type mime-application. C'est une intégration *ad hoc*. Elle illustre aussi des difficultés concrètes qui sont apparues lors de l'intégration d'un composant (COW est un EJB) avec un outil externe et deux LMS. Une approche basée sur les web services a été réalisée avec succès (COW-CVA) alors qu'une intégration plus profonde (COW-OU) avec un LMS basé sur la même technologie a été un échec pour des raisons technologiques (versions différentes du serveur d'application Jonas et du service de sécurité associé).

3.4 Intégration du logiciel de suivi d'activités Reflet

Reflet (Després et Coffinet, 2004) est un outil de suivi des activités destiné aux tuteurs et aux apprenants. Il permet de visualiser l'avancement du travail de l'étudiant en FOAD (Formation Ouverte et A Distance) : c'est l'étudiant qui valide les activités qu'il a effectuées.

3.4.1 Description

Dans le dispositif d'intégration – appelée routeur – Reflet est couplé à une base de données (Tomcat + MySQL).

3.4.2 Choix adoptés pour l'intégration

Son interface graphique est intégrée de manière *ad hoc* dans deux LMS existants : WebCT et Dokeos. Lors de l'intégration avec les LMS, des difficultés ont été identifiées.

La première concerne *la modélisation de la formation*. En effet, dans Reflet cette modélisation utilise un modèle propriétaire (MAT). Il fallait donc récupérer la formation définie dans le LMS et tout ressaisir. Les auteurs proposent d'utiliser IMS-LD dans Reflet ou de définir des routines de conversion. La seconde concerne *l'identification des utilisateurs* : le login et le mot de passe de l'utilisateur sont provisoirement passés en clair dans l'URL qui lance Reflet. La troisième concerne *l'intégration graphique du résultat* de la visualisation de Reflet dans le LMS. Cette intégration est de type juxtaposition : ouverture dans une fenêtre à part ou dans un cadre à part, sans adaptation au "look and feel" du LMS. Par ailleurs, pour sa

visualisation Reflet utilise un arbre de la formation. En conséquence, cet arbre apparaît en double si le LMS en possède déjà un. Et actuellement, il faut conserver les deux car ils permettent chacun l'accès à des fonctionnalités différentes et nécessaires. La quatrième concerne *la synchronisation des données*. En effet, la base d'informations sur l'utilisateur évolue. Il faut que Reflet et le LMS soient en phase. De même, le modèle de la formation évolue au cours du temps dans le LMS. Il faut détecter et répercuter ces évolutions dans Reflet, à la main pour l'instant. Enfin, Reflet est aujourd'hui offert comme un service logiciel accessible sur les serveurs du LIUM. Ceci montre la tendance actuelle de partage et de réutilisation dans d'autres contextes, de prototypes d'EIAH développés dans les laboratoires.

3.4.3 Conclusion

Les différents éléments identifiés ici lors du passage vers un composant réutilisable sont les suivants : modélisation de la formation (récupération dans la plate-forme d'accueil, mise au standard IMS-LD), identification unique des utilisateurs, intégration plus forte avec l'interface utilisateur (même "look and feel"), synchronisation des données en cas d'évolution (ajout d'utilisateurs, ...). Ainsi, pour cette adaptation, les questions à se poser sont : identifier la valeur ajoutée du passage en mode composant (est-ce vraiment intéressant de passer de JSP/Servlets à EJB par exemple ?), identifier les informations et/ou événements produits et émis (pour définir les interfaces). Afin de faciliter l'intégration de Reflet, les auteurs pensent qu'il faut augmenter le nombre de paramètres de ce dernier afin de l'ouvrir.

3.5 Intégration de Formid à Baghera

Formid (Guéraud et Al, 2004) est un outil de simulation couplé à du suivi des activités pour les tuteurs. Baghera (Baghera, 2005) est un Système Multi Agents (SMA) utilisé dans des activités pédagogiques en géométrie.

3.5.1 Description

L'outil de simulation a été intégré au SMA sous forme d'un agent « compagnon élève ». L'intégration a été faite avec les objectifs suivants : éviter de toucher au code et ne pas ajouter de contraintes sur l'interface. La simulation fournit la liste des mécanismes et propriétés observables. Cette liste est nécessaire pour l'intégration (comme indiqué section 2.2).

3.5.2 Choix adoptés pour l'intégration

Dans la partie simulation, il a fallu adapter l'interface. Ce travail a nécessité un mois d'ingénieur à temps plein, en sachant que cette personne connaissait bien Formid et a dû s'adapter à Baghera. Nous pouvons donc constater que l'adaptation est un travail conséquent. Pourtant, Formid repose sur le modèle MVC qui est sensé séparer le modèle, les aspects contrôle et la visualisation simplifiant ainsi la réutilisation ou l'adaptation des interfaces notamment. Côté Baghera, il suffit d'ajouter un agent base de données.

3.5.3 Conclusion

Cette étude a l'avantage de montrer un exemple d'intégration d'un système multi-agents avec un système qui ne l'est pas. Le couplage obtenu est assez souple et prometteur. L'intégration de l'interface reste relativement laborieuse, même si l'on pouvait penser que cela aurait été facilité par le modèle MVC. Cependant l'intégration d'une interface qui ne serait pas MVC serait encore plus lourde. Aussi retenons que pour l'interface, être MVC est une condition nécessaire mais non suffisante pour être un bon candidat à l'intégration.

3.6 Discussion

Les exemples d'intégration présentent les nouveaux problèmes apparus lors d'essais d'intégration. Nous n'allons pas tous les reprendre car d'une part ce serait fastidieux et d'autre part la retro-ingénierie (reverse engineering) des solutions adoptées n'est pas aisée. Cependant essayons de dégager des grandes tendances

(Tableau 1).

cf.	Éléments intégrés	Patatoïde	Glu	Communication des données
3.1	des EIAH de géométrie	médiateur + gestionnaire d'activité	composant Corba et son IDL	RPC et événements
3.2	SSCI et Tulka whiteboard	deux middleware	classes java spécifiques	échanges de messages.
3.3	COW et 3D SPIN	COW	page web	via une URL
3.3	COW et LMS CVA	serveur d'application		sérialisation des objets
3.3	COW et LMS OU	COW + serveur d'application	deux versions du serveur d'application + service de nommage commun	via une interface WSDL
3.4	Reflet et LMS WebCT / Reflet et LMS Dokeos	Reflet + base de données		via une URL
3.5	Formid et Baguera	SMA	ajout d'un agent	celle de la plate-forme Multi-Agents

Tableau 1. Eléments de comparaison des études de cas

La dernière colonne présente les différentes solutions adoptées lors de la communication des données.

La plupart des études porte sur l'intégration de deux EIAH. Quid du passage à trois ? Est-il judicieux d'adopter une approche centralisée comme en 3.1 ? Ou bien faudrait-il favoriser les approches décentralisées comme par exemple en 3.5 (SMA) ? C'est aussi l'approche choisie dans l'architecture

orientée service SOA (Service Oriented Architecture) (He, 2003).

Pour l'intégration des EIAH, nous pensons que les problèmes à résoudre sont tels qu'il n'est pas possible de les résoudre une fois pour toute. Aussi les solutions ad hoc semblent le meilleur choix stratégique actuellement, dans le but d'expérimenter. Les expérimentations sont une condition sine qua non pour progresser sur ce point dans la communauté. Le domaine des EIAH ne possède pas encore des composants de type COTS (Composants sur étagères) permettant de rationaliser quelques problèmes d'intégration. En outre, il serait souhaitable d'identifier clairement les dimensions sur lesquelles porte l'intégration dans chaque étude de cas. En particulier identifier dans quelle dimension se situent les problèmes d'interopérabilité qui se sont posés et les limites des différentes technologies que nous rencontrons permettrait de tirer plus facilement des enseignements des différentes études.

4. Conclusion : vers des guides de bonnes pratiques

Cet article est une première tentative pour dégager des principes d'intégration d'EIAH. Nous sommes dans une démarche "design by reuse" en tentant de définir des bonnes pratiques pour que les prochains développements en EIAH tendent vers la démarche "design for reuse".

La diversité des études de cas que nous analysons a montré les difficultés que rencontrent les chercheurs en EIAH pour atteindre un degré d'intégration satisfaisant. Elle met aussi en évidence des besoins afin de trouver comment intégrer à moindre frais des applications existantes ou futures.

Au regard de ces études de cas, déterminer le mécanisme le plus approprié pour intégrer un ensemble d'EIAH semble une gageure. Afin d'y palier, nous avons défini les caractéristiques d'un dispositif d'intégration d'EIAH mais également les verrous scientifiques liés aux dimensions de l'intégration pour déterminer les points importants à prendre en compte. Ainsi, nous proposons d'utiliser les types d'intégration (juxtaposition, transmission, coopération, interopération, assemblage et composition) et les dimensions de celle-ci (indexation des composants, communication de données, commande de plusieurs composants, protocole de coopération pour les composants, gestion des interfaces et collecte des données) pour caractériser les difficultés que nous rencontrons dans l'intégration des EIAH.

Nous recommandons :

- d'utiliser les spécifications, les normes et les standards existants ainsi que les technologies leaders, quitte à en montrer les limites et tout en prenant la distance qui s'impose.

- d'annoncer le contexte (version de la machine virtuelle java, du navigateur, du serveur d'application, etc.) dans lequel a été développé et utilisé le composant.

- de concevoir des dispositifs et des composants qui aient les fonctions énoncées en 2.2

- d'utiliser des guides de bonnes pratiques existant

- de construire des interfaces plastiques

- de favoriser des contextes où les retours d'usages et les retours d'expérience puissent être mutualisés.

- de mettre en place des outils permettant de mesurer la qualité de cette "intégration"

Nous recommandons par ailleurs de bien décrire le composant (ou bien se décrire) afin d'afficher clairement les informations et événements entrants, produits, émis et observables. Ce n'est pas déterminant lorsque le nombre d'EIAH utilisé est faible. Cependant, cette description est primordiale dans un contexte de mutualisation. Pour cela, nous avons besoin d'une méthodologie de description et d'un

vocabulaire commun pour décrire les fonctionnalités et les propriétés des composants et des dispositifs d'intégration. Ce vocabulaire devrait utiliser une terminologie claire faisant référence à une ontologie des ressources pédagogiques (OURAL, 2005). Cette ontologie doit être négociée et approuvée par la communauté des chercheurs en EIAH. Par ailleurs, d'autres ontologies sont nécessaires pour permettre une description plus aisée des données qu'échangent les composants, *i.e.* les connaissances du domaine d'apprentissage, les connaissances d'interaction (ou traces), les connaissances d'enseignements (stratégies tutorielles), les modèles et profils d'apprenants, modèle de formation, etc.

Plusieurs projets de recherche en cours ont pour but de concevoir des outils et des modèles pour permettre la mutualisation des résultats de la recherche en EIAH, et en particulier les productions logicielles. Au niveau européen, le réseau NOE (Network Of Excellence) KALEIDOSCOPE (KALEIDOSCOPE, 2005) se propose de définir une plate-forme virtuelle dans le but d'obtenir une meilleur interopérabilité entre les produits de la recherche, s'appuyant sur une approche basée sur le composant. Au niveau national, l'action spécifique plateforme (ASPF, 2005) cherche aussi à outiller la communauté des chercheurs en EIAH pour une meilleure réutilisation des résultats de recherche. Nous pensons, en accord avec (ASPF, 2005) que *"l'architecture logicielle de la plate-forme doit encourager une approche de conception et de réalisation des EIAH de type "Model-Driven", car c'est un aspect qui nous paraît fondamental pour l'avenir de la conception des EIAH et de leur interopérabilité."*

BIBLIOGRAPHIE

AIKEN R. M., BESSAGNET M.-N., ISRAEL J. (2005). Interaction and Collaboration using an Intelligent Collaborative Learning Environment, Revue Education and Information Technologies, Vol 10, N° ½, Avril 2005, pp 65-80, ISSN 1360-2357.

AS Plateforme pour la recherche en EIAH. Contributions de l'Action Spécifique " Conception d'une Plateforme pour la recherche en EIAH " à l'ingénierie des Environnements Informatiques pour l'Apprentissage Humain. Revue STICEF, ce numéro.

BARBIER F., CAUVET C., OUSSALAH M., RIEU D., BENNASRI S., SOUVEYET C. (2002), Composants dans l'ingénierie des systèmes d'information : concepts clés et techniques de réutilisation http://sis.univ-tln.fr/gdri3/fichiers/assises2002/papers/05-ComposantsDansLIngenierieDesSystemesDInformation.pdf

BRASSARD C., DAELE A. (2003). *Un outil réflexif pour concevoir un scénario pédagogique intégrant les TIC*. In Desmoulins C., Marquet P., Bouhineau D. (Eds) Actes de la conférence EIAH 2003, Strasbourg.

BESSAGNET M.-N., AIKEN R. M., ISRAEL J. (2005). Helping to Network (Groups of) Learners Using a Collaborative Learning System, Conférence WCCE 2005 4-7 July 2005

CALVARY G., COUTAZ J. THEVENIN, D. LIMBOURG, Q., BOUILLON, L., VANDERDONCKT J. A Unifying Reference Framework for Multi- Target User Interfaces, in Interacting With Computers, VOL. 15/3, pp. 289-308, 2003

DESPRéS C., COFFINET T. (2004), *Reflet, un miroir sur la formation*, In: Actes de TICE 2004, 20-22 octobre 2004, Compiègne (France), p. 19-24.

Actes de la conférence EIAH'2005.

GUéRAUD V., ADAM J.-M., PERNIN J.-P., CALVARY G., DAVID J.-P. (2004). L'exploitation d'Objets Pédagogiques Interactifs à distance : le projet FORMID. Revue STICEF, Volume 11, . mis en ligne le 25/05/2004, http://sticef.org

GRANDBASTIEN, M., BOUYT, B., CLAëS, C., BOURDA, Y., DUVAL, E. (2002). *Les ressources*

numériques : Enjeux de la normalisation, In TICE'2002, Villeurbanne.

HE, H. (2003). What is Service-Oriented Architecture? September 30, 2003. http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html

JACQUINOT-DELAUNAY G., MONNOYER L., *Le dispositif. Entre usage et concept*, http://www.wolton.cnrs.fr/hermes/b_25fr_presentation.htm

PERNIN J.-P., LEJEUNE A. (2004). Modèle pour la réutilisation de scénarios d'apprentissage, I*NRP et CLIPS-IMAG Grenoble. In* ISDM n°18 – 2004. Colloque TICE MEDITERRANEE 26 - 27 novembre.

REBAï, I., LABAT, J.-M. (2004). *Des métadonnées pour la description des composants logiciels pédagogiques*, Technologies de l'Information et de la Connaissance dans l'Enseignement Supérieur et l'Industrie. Compiègne : Université de Technologie de Compiègne. p80-87.

ROSSELLE, M. (2001). *Conception d'un dispositif d'expérimentation de logiciels éducatifs – Application en géométrie*. Thèse de doctorat, Université Henri Poincaré, Nancy I.

ROSSELLE M., GRANDBASTIEN M. (2004). Towards interoperable ILEs: results from a case study in the geometry domain, Conférence CALIE'2004, Grenoble. 8p.

STAFFORD J.A., WALLNAU K., Component composition and Integration, In I Crnkovic and M. Larsson (eds) Building Reliable Component-Based Software Systems, pp179-191, Artech House Inc.

VANTROYS, T., PETER Y. (2003). *COW, a Flexible Platform for the Enactment of Learning Scenarios*. In: J. Favela and D. Decouchant (eds.): Groupware: Design, Implementation, and Use - 9th International Workshop, CRIWG 2003. LNCS 2806, Springer-Verlag, pp 168-182.

WICKS M. (2004), Tool Integration in Software Engineering: The State of the Art in 2004. http://www.macs.hw.ac.uk:8080/techreps/docs/files/HW-MACS-TR-0021.pdf. 26p.

Références à des sites Internet

Site public de l'action spécifique plate-forme, http://www.univ-lille1.fr/aspf. Visite le 29/07/05

http://www-baghera.imag.fr/. Visite le 05/04/05.

Calvary G., Coutaz J. (2002). Plasticité des interfaces : une nécessité !. Assises GDR I3 Décembre 2002. http://sis.univ-tln.fr/gdri3/fichiers/assises2002/papers/14-PlasticiteDesInterfaces.pdf. Visite le 05/04/05

Campus Virtuel d'Archimed, http://www.archimed.fr/. Visite le 29/07/05.

IMS Learning Design, http://www.imsglobal.org/learningdesign/. Visite le 25/07/2005

KALEIDOSCOPE. http://www.noe-kaleidoscope.org/. Visite le 05/04/2005.

LTSC, Learning Technology Standards Committee. http://ltsc.ieee.org/. Visite le 07/04/2005.

Ontologies pour l'Utilisation de Ressources de Formation et d'Annotations sémantiques en Ligne, études et propositions à partir de cas d'utilisation
http://www.dr4.cnrs.fr/tcan/tcan/activites/2003/grandbastien.mov. Visite le 04/04/05.

http://www.openuss.org. Visite le 04/04/25.

http://giotto.mathematik.uni-tuebingen.de/~mibe/tulkawhiteboard/. Visite 04/04/2005.

A propos des auteurs

Marilyne Rosselle est Maître de Conférences à l'IUP MIAGE de l'Université Picardie Jules Verne à Amiens et membre du laboratoire SaSo (Savoirs et Socialisation). Elle a obtenu son doctorat en Informatique à l'Université Henri Poincaré (Nancy) en 2001. Ses recherches portent sur l'étude de l'interopérabilité et les usages des EIAH.

Courriel : marilyne.rosselle@u-picardie.fr

Marie-Noelle Bessagnet est Maître de Conférences à l'Université de Pau et des Pays de l'Adour et membre du LIUPPA(Laboratoire d'Informatique de l'UPPA) dans l'équipe IDEE (Interaction, Document Electronique, Education). Elle a obtenu son doctorat en Informatique à l'université Paul Sabatier (Toulouse) en 1991 et un Master d'Administration des Entreprises en 1994. Ses recherches actuelles portent sur les environnements de développement d'EIAH, sur l'intégration des TICE dans les systèmes d'information des enseignants et des apprenants, sur l'apprentissage collaboratif.

Courriel : marie-noelle.bessagnet@univ-pau.fr

Toile : http://mn.bessag.net

Thibault Carron est Maître de Conférences à l'Université de Savoie et membre du laboratoire SysCom (Systèmes Communicants). Il a obtenu son doctorat en Informatique à l'école Nationale Supérieure des Mines de Saint-Etienne en 2001. Ses recherches actuelles portent sur l'étude de l'observation dans les collecticiels.

Courriel : Thibault.Carron@univ-savoie.fr

Référence de l´article :

Marilyne Rosselle, Marie-Noelle Bessagnet, Thibault Carron , Comment intégrer des logiciels issus de la recherche en EIAH ?, *Revue STICEF*, Volume 12, 2005, ISSN : 1764-7223, mis en ligne le 17/02/2006, http://sticef.org