وزارة التعليم العالي والبحث العلمي

BADJI MOKHTAR UNIVERSITY -ANNABA-
UNIVERSITE BADJI MOKHTAR -ANNABA-

جامعة باجي مختار
- عنابة -

Faculté : Sciences de l'Ingénieur -Année 2009-

Département : Informatique

MEMOIRE

Présentation en vue de l'obtention du diplôme de magister

Environnement collaboratif synchrone basé services web destiné pour la gestion des connaissances

Option

Texte Parole et Image

Par

Mohamed Saïd Mehdi MENDJEL

DIRECTEUR DE MEMOIRE: M. M. SELLAMI Professeur Univ. Annaba

DEVANT LES JURY

PRESIDENT: Mr.FARAH Nadir	**Maitre de conférences**	**Univ. Annaba**
EXAMINATEURS:		
Dr. SERIDI Hassina	**Maitre de conférences**	**Univ. Annaba**
Dr. KHADIR Tarek	**Maitre de conférences**	**Univ. Annaba**
Dr. KHOLLADI M.Kheireddine	**Maitre de conférences**	**Univ. Constantine**

Remerciements

Avant tout, je remercie Dieu le tout puissant de m'avoir aidé à mener à bout ce travail, et a le concrétiser.

Je remercie vivement Monsieur SELLAMI Mokhtar, professeur à l'U.B.M.A d'avoir accepter de diriger ce mémoire et pour ses conseils et orientations avisés.

Je remercie également Messieurs FARAH Nadir Dr à l'U.B.M.A, KHADIR Tarek Dr à l'U.B.M.A, KHOLLADI Med Kheireddine Dr à l'Université de Constantine et Mme SERIDI Hassina Dr à l'U.B.M.A de m'avoir fait l'honneur d'accepter de juger ce mémoire.

Merci infiniment

Résumé

Le développement fulgurant des technologies de l'information, des réseaux et des moyens de communication est à l'origine d'une expansion prodigieuse de l'internet. Nous traversons une nouvelle ère où la démocratisation de l'internet et le passage à la société de l'information conduisent à un processus de globalisation incitant à la mise en relation des personnes et à la constitution d'équipes virtuelles réparties partout dans le monde. C'est ainsi l'ère des plates-formes collaboratives construites dans des domaines différents (gestion des connaissances, enseignement à distance, Télémédecine, etc.), intégrant plusieurs outils de collaboration (synchrones et asynchrones) et offrant des possibilités d'interaction et d'échange d'informations entre des groupes de personnes situés dans des endroits différents.

L'arrivé de l'architecture orientée service et des technologies des services web a apporté de nouvelles solutions à ces plates-formes collaboratives , notamment au niveau de la réactivité et de l'évolutivité, en mettant en place des services ou des modules de collaboration évolutifs et recomposables au gré des besoins des utilisateurs, et au niveau de la flexibilité et de l'hétérogénéité, en faisant communiquer des services de natures différentes, qui peuvent devenir à leur tour des nouveaux services consommables par d'autres plates-formes.

Le but de ce mémoire est ainsi de proposer une infrastructure collaborative basée sur un ensemble d'outils de collaboration synchrones conçus sous forme de services et qui peuvent être consommés par n'importe quelle plate-forme offrant ce type de technologies.

Mots clés : Architecture Orientée Services, SOA, Services web, Plate-forme collaborative, Outil de collaboration synchrone, Outil de collaboration asynchrone.

Abstract

The rapid development of information technology, networks and communications rise to a prodigious expansion of the Internet, We are in a developing era where democratization of the Internet and the transition from information to society has led a process of globalization, encouraging the linking of people and the creation of virtual teams spread around the world. Thus the era of collaborative platforms built in different fields, such as knowledge management, distance learning, telemedicine, etc, that are Integrating several collaborative tools, which include synchronous and asynchronous, providing opportunities for interaction and exchange of information between groups of people located in different places.

The arrival of service-oriented architecture and Web services technologies has provided solutions to this newely found collaborative platform, especially in the reactivity and the scalability, by setting up services or collaborative evolutionary modules and redialing to the needs of users and concerning flexibility and heterogeneity by communicating with different types of services, which have become new services consumable by other platforms.

The purpose of this memorandum is to provide a collaborative infrastructure based on a set of synchronous collaboration tools designed in the form of services and which can be consumed by any other platform offering this type of technology.

Key words: Service oriented architecture, SOA, Web service, Collaborative platform, Synchronous tool for collaboration, Asynchronous tool for collaboration.

ملخص

التطور السريع في تكنولوجيا المعلومات ، الشبكات والإتصالات أدى إلي توسع هائل للإنترنت. إننا نمر بعهد جديد من الديمقراطية في لإنترنت ، و الانتقال إلى مجتمع المعلومات اللذان أديا إلى عملية عولمة التي قامت بتحريض الناس على إقامة علاقات وإنشاء فرق إفتراضية مع بعضهم البعض وفي كافة أنحاء العالم. وهكذا يأتي عهد بناء مناهج تعاونية في مختلف المجالات (إدارة المعرفة، التعلم والطب عن بعد..... إلخ). هذه المناهج التعاونية يمكن أن تدمج عدة أدوات تعاونية المتزامنة منها و الغير متزامنة، و هذا لإتاحة فرص التفاعل وتبادل المعلومات بين مجموعات من الناس عن بعد.

إن وصول تكنولوجيا الخدمات الموجهة وتكنولوجيا خدمات الانترنت وفر عدة حلول جديدة لهذه المناهج التعاونية، لاسيما من حيث سرعة الاستجابة وذلك من خلال إنشاء وحدات تعاونية قابلة للإعادة حسب احتياجات المستخدمين،أو من حيث مستوى المرونة وعدم التجانس بجعل كافة الخدمات تتعامل مع بعضها البعض رغم إختلاف طبيعتها، التي تصبح بدورها خدمات جديد قابلة للإستهلاك من طرف مناهج أخرى.

إذا الغرض من هذه المذكرة هو توفير بنية تعاونية تستند إلي مجموعة من أدوات التعاون المتزامن مصممة في شكل خدمات يمكن إستهلاكها من طرف أي منصة توفر هذا النوع من التكنولوجيا.

الكلمات المفتاحية: الهندسة الموجهة للخدمات، خدمات الشبكة العالمية، المنصة التعاونية، أداة التعاون المتزامن، أداة التعاون الغير متزامن.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Problématique abordée

Face aux nouveaux défis à relever, l'information, la connaissance et le savoir-faire constituent des ressources stratégiques et opérationnelles indispensables pour permettre aux diverses organisations d'assurer une croissance pérenne. Le développement et l'exploitation des connaissances requièrent alors de déployer des dispositifs organisationnels et techniques (réseaux informatiques et les technologies de l'information et de la connaissance) permettant à la fois leur mémorisation, leur partage, leur production collective et leur diffusion au plus près des acteurs concernés.

Autour de ces technologies, le management des " communautés de pratiques " prend une importance considérable, en définissant des formes nouvelles de coordination au sein ou entre les organisations, mais en offrant également des outils facilitant la communication et les échanges, tels que les services de collaboration synchrones. Nous pouvons citer à titre d'exemple :

- La conversation (chat) : c'est un outil de communication en temps réel sous forme de texte.

- Les tableaux blancs : Ces applications permettent l'échange d'informations visuelles comme l'écriture de texte et le tracé de schémas sur un tableau.

- La vidéoconférence : L'utilisation de la vidéoconférence s'inscrit dans l'éventail des moyens offerts par les technologies de l'information et de la communication (TIC). À ce titre, elle est marquée par une composante technique importante dont il faut tenir compte.

- Le bureau partagé : Le partage de bureau permet à chaque chercheur connecté à l'espace de travail virtuel de partager son bureau, ainsi que toutes les applications installées sur son ordinateur.

Faire interagir ces différentes applications en réseau a toujours été une affaire complexe, notamment si on souhaite standardiser certains aspects de cette interaction.

Pour résoudre ce problème on fait appel à un ensemble de protocoles qui permettent, au moins de faire communiquer (avec un protocole de haut niveau, pas juste des bits) des programmes tournant sur des machines différentes et écrits dans des langages de programmation différents. Ils le font en reprenant certains principes du Web (transport sur HTTP, formatage en XML) mais en mettant l'accent sur la communication entre applications, pas entre humains en se basant sur une architecture orientée services.

Notre premier objectif, dans ce mémoire, est ainsi de proposer une infrastructure collaborative intégrant et offrant des services de collaboration synchrones en se basant sur les nouvelles technologies orientées services.

Notre deuxième objectif est de mettre en œuvre cette infrastructure en proposant une approche qui se base sur l'intégration d'applications déjà existantes en utilisant des services web. De nombreux travaux se sont déjà focalisés sur cette problématique : les plates-formes collaboratives à base d'intégration. Notamment dans le domaine des applications reparties, où le spectre va de solutions à base d'intergiciels (tel que CORBA) à l'utilisation de nouvelles technologies des services web.

Plan du mémoire

Le mémoire est structuré en deux parties principales, plus cette introduction et une conclusion générale. La première partie, intitulée état de l'art comprend trois chapitres :

Dans le premier chapitre qui s'intitule apparition des technologies web, nous introduisons un petit historique sur l'évolution des différentes technologies amenant à la problématique générale. L'objectif de ce chapitre est de donner une idée générale sur ces nouvelles technologies qui ont été normalisées et standardisées dans le but de réaliser des applications simples, flexibles et hétérogènes.

Dans le deuxième chapitre qui s'intitule les plates-formes collaboratives, nous présentons un état de l'art de notre problématique de recherche. Nous introduisons des concepts fondamentaux sur la collaboration et la coopération, termes et classification afin de présenter le contexte de nos travaux. Dans ce contexte nous abordons la problématique liée au développement d'applications collaboratives, nous présentons les différents outils de collaboration ainsi que quelques plates-formes collaboratives que nous avons testé tout au long de notre travail, et nous terminons ce chapitre par une analyse des différentes approches de développement d'applications

collaboratives ainsi que leurs limitations pour pouvoir concevoir une plate-forme qui se base sur une approche d'intégration basée sur des frameworks.

Nous consacrons le chapitre 3 qui s'intitule « architecture orientée service » pour présenter l'apparition de plusieurs architectures jusqu'à l'arrivée de l'architecture orientée service qui joue le rôle principal dans notre proposition, nous expliquons aussi dans ce chapitre pourquoi on a choisi une telle architecture en démontrant que les possibilités offertes par cette dernière ne sont pas uniquement des promesses.

Nous débutons la deuxième partie qui concerne notre contribution, par un chapitre 4 qui s'intitule la technologie des services web, dans lequel on fait une présentation générale des services web, définition et description, les principales technologies de développement, etc, pour pouvoir proposer une architecture ou bien une infrastructure collaborative basée sur ce type de technologies.

Dans le chapitre 5 intitulé architecture proposée, nous présentons l'architecture de notre proposition qui se base sur une architecture 3 tiers et qui englobe plusieurs technologies :

- L'utilisation d'outils open source.
- L'utilisation d'une architecture orientée service.
- L'utilisation de la technologie des plates-formes riches RIA.
- la notion d'intégration d'applications.

Dans le chapitre 6 intitulé Expérimentation, et dans le but de valider notre approche pour offrir a une communauté d'utilisateurs un environnement collaboratif offrant des outils de collaboration synchrones dans un domaine bien précis, nous optons pour une intégration de deux plates-formes libres. La première plate-forme s'appelle Openmeetings qui se base sur une architecture orientées service, elle offre à l'utilisateur via une interface riche un ensemble de services synchrones (la vidéoconférence, le tableau blanc partagé, le chat et le bureau partagé).
La deuxième plate-forme s'appelle Compendium, c'est une plate-forme destinée pour la gestion des connaissances, et qui facilite la structuration et la modélisation collaborative des idées entre les utilisateurs.

Chapitre 1
Apparition des technologies web

1.1 Historique

Les plates-formes collaboratives, les services web et les technologies associées sont très présents dans l'informatique d'aujourd'hui. Pour cela, on va proposer un petit aperçu historique de l'évolution de ces différentes technologies amenant à la problématique générale qui les intègre. Cela ne représente qu'un choix restreint et d'autres théories peuvent s'appliquer.

1.2 Démocratisation de l'internet

L'un des facteurs majeurs, si ce n'est le principal facteur, de l'avènement des services web sur les ordinateurs de tous les jours est la démocratisation des réseaux et d'Internet en particulier. En effet, alors qu'avant 1995-1998, un ordinateur était principalement destiné à fonctionner de façon autonome. Aujourd'hui, un ordinateur sans accès au réseau des réseaux se trouve limité au niveau de ses fonctionnalités et de l'accès aux informations. Ce facteur est, de plus, présent au niveau des entreprises, mais également des particuliers, d'où une évolution et un changement radical dans l'utilisation de l'ordinateur.

1.2.1 Parc d'ordinateurs hétérogène

L'interconnexion des ordinateurs entre eux par l'intermédiaire d'un réseau (et par l'extension d'Internet) pose le problème de la gestion du parc hétérogène. En effet, l'hétérogénéité se traduit :

- Par la multitude des systèmes d'exploitation.
- Par les différentes versions majeures et mineures plus ou moins compatibles d'un même système et de ses services.
- Par les différentes architectures possibles.

L'interconnexion d'un tel parc présente de nombreux artifices permettant une communication plus ou moins aisée. C'est pour cela que les différentes technologies, abordées par la suite, tentent d'uniformiser les communications en utilisant des langages de communication standardisés et indépendants de la plate-forme source et/ou cible.

Le parc d'ordinateurs est très hétérogène au niveau de l'architecture et des systèmes, comme nous venons de le voir, mais également au niveau de la puissance de calcul.

Aujourd'hui, la puissance peut être très faible, comme dans le monde de l'embarqué (petite puissance de calcul, peu de mémoire, peu d'énergie, tel que nos PDA dont l'horloge du processeur est cadencée entre 100 et 300 Mhz). Mais cette puissance peut être énorme, comme pour certains serveurs ou encore des calculateurs permettant de réaliser des simulations météorologiques ou nucléaires.

Ces différences doivent cependant permettre à un ensemble d'ordinateurs de pouvoir communiquer à travers un réseau sans restreindre l'utilisation de l'un ou de l'autre.

1.2.2 Sous utilisation des ressources et répartition des calculs

L'évolution des capacités de calcul des ordinateurs est très liée au nombre et à la complexité des semi-conducteurs utilisés pour la fabrication de leurs processeurs.

La plupart des ordinateurs sont cependant sous-utilisés. L'exemple le plus flagrant est l'ordinateur de bureau qui reste allumé la nuit : à part quelques transferts réseaux liés aux scripts de sauvegarde et aux mises à jours logiciels, il n'a aucune information à traiter pendant ce laps de temps.

C'est ainsi que sont nés plusieurs projets tel que Decrypton [1], lancé par l'AFM (Association Française contre les Myopathies) [2], et IBM suite au Téléthon 2001, dont le but est de réaliser la première cartographie du protéome humain (c'est-à-dire la modélisation de l'ensemble des protéines produites par le génome humain) qui a mobilisé 75000 ordinateurs permettant de réaliser en 2 ans ce qu'un ordinateur, à lui tout seul, aurait mis 1170 années.

Tous ces calculs ne sont possibles que par l'entente mutuelle traduite par les communications entre des serveurs et des ordinateurs hétérogènes à travers un réseau non stable (Internet). Ces communications utilisent alors des protocoles (pour la communication) et des langages (pour le codage des données) se rapprochant des technologies proches des services web que nous présenterons dans la suite de ce mémoire.

Il est à noter que ces approches de gestion de l'interopérabilité ne sont pas récentes et ont toujours existé depuis l'apparition des premiers réseaux (ARPANET –Advanced Research Projects Agency Network–, projet lancé en 1967, première démonstration en 1972) [2]. Mais ces dernières années, cela est rendu encore plus présent par l'utilisation d'un plus grand nombre d'ordinateurs et l'automatisation obligatoire des différentes tâches de communications à travers

un réseau. Cette automatisation se réalise sans même savoir précisément les destinataires des données et par quels moyens elles vont être transmises.

1.3 L'Apparition de JAVA

Parmi les technologies nées pour apporter des solutions aux problèmes présentés précédemment, le langage Java joue un grand rôle. Java est une technologie composée d'un langage de programmation orienté objet et d'un environnement d'exécution, développé par Sun. La première version fut présentée au SunWorld en 1995. Plusieurs versions ont suivi et aujourd'hui, la dernière version stable est Java 1.6 alias Mustang (ou Java SE 6) Une version bêta est sortie le 15 février 2006, une autre bêta en juin 2006, une version "release candidate" en novembre 2006, et la version finale le 12 décembre 2006. Avec cette version, Sun remplace le nom J2SE par Java SE et supprime le « .0 » au numéro de version. Enfin, la version Java 7 vient de sortir en janvier 2009, son nom de code est Dolphin, une des nouveautés majeures de cette version est l'ajout des closures, il s'agit de définir d'une manière légère des fonctions capables de disposer du contexte d'un objet ou d'une autre fonction [3].

Nous allons présenter les principaux atouts de la plate-forme Java, à savoir les aspects portabilité et multi-plateforme, liés étroitement à l'utilisation d'un concept de JVM (Java Virtual Machine) et enfin, d'une API très riche et documentée.

1.3.1 Portabilité et multi-plateformes

La plateforme Java présente l'avantage d'être présente sur la majorité des systèmes (Windows, Linux, MacOS, Sun, ...), et également sous certaines formes sur les plates-formes embarquées. Cela est rendu possible par l'utilisation du concept de bytecode et d'une API correctement écrite et par l'utilisation d'une JVM.

1.3.1.1 Le Concept de bytecode

Le concept de bytecode consiste à compiler le code dans un langage intermédiaire relativement proche du langage machine mais assez éloigné pour permettre d'y ajouter un interpréteur. Cet interpréteur sera la clé de voûte du portage de la plateforme Java : le code Java, une fois compilé en bytecode est analysé par une application convertissant les instructions du

langage intermédiaire en langage machine. Le code ainsi compilé peut alors être exécuté, sans recompilation, sur une architecture différente.

Cela étend le concept du « write once, execute everywhere » au concept du « compile once, execute everywhere ».

1.3.1.2 API de base

La plateforme Java présente une API de base définissant les types de base. Seule cette partie est dépendante de l'architecture cible et n'est pas écrite en Java. C'est donc sur cette partie relativement compacte que se base la portabilité.

1.3.1.3 API très complète et documentée

Java est une plateforme de nouvelle génération (par rapport aux langages C/C++ et leurs bibliothèques). Il se devait de proposer directement à l'utilisateur tout un ensemble de classes et méthodes testées et approuvées permettant de programmer rapidement les différents types et concepts de base de l'algorithmique. C'est le pari que Sun a réalisé avec Java et sa JDK regroupant 212 classes réparties en 8 paquets dans la version 1.0 et 3562 classes regroupées dans 166 paquets dans la version 5.0 [4].

1.3.2 La machine virtuelle Java

La machine virtuelle Java (JVM) est l'interpréteur de bytecode produit par le compilateur. C'est la seule partie de la plateforme Java à porter sur le système cible de déploiement. Sun en fournit des versions pour les architectures les plus courantes comme Windows, Linux et Solaris, mais il existe également des JVM pour les systèmes d'exploitation tels que MacOS, Windows CE, PalmOS, etc.

Cette JVM transforme le bytecode en instruction machine, impliquant une optimisation de celle-ci pour éviter la lourdeur d'une interprétation de code. Pour cela, elle utilise des algorithmes tel que « Just In Time » (JIT) ou encore « HotSpot ». Une des améliorations de JIT consiste à ne transformer en code exécutable que les morceaux de blocs de bytecode qui seront réellement exécutés : seules les méthodes réellement exécutées du code sont interprétées. Par contre, HotSpot analyse le code exécuté pour optimiser les éléments clés du code : par exemple, la conversion en code machine des parties de codes souvent répétées sont conservées en mémoire.

Enfin, une courte présentation de Java ne serait pas complète sans parler de la présence d'un « garbage collector » ou « ramasse-miettes » permettant au système de conserver une référence vers les zones de mémoires dont le programme utilisateur n'est plus capable d'accéder (écrasement de référence suite à une affectation, etc...) pour ensuite les libérer pendant les moments d'inactivités. Ou encore de la présence de « Java Native Interface » (JNI) permettant aux, programmes s'exécutant dans la machine virtuelle, d'accéder à l'interface native du système, comme certains pilotes par exemple, pour des raisons de performances ou de non compatibilités avec les interfaces fournies par Java, au prix tout de même, d'une incompatibilité, entre plates-formes différentes, du code Java produit dans ce cas.

1.4 L'Apparition d'XML

Le langage XML (Extensible Markup Language) est un langage balisé, issu d'une recommandation W3C (WorldWideWeb Consortium), ayant pour but d'encoder tout type de données, indépendamment du code machine de celles-ci. Il a été développé dans le but de partager facilement des données entres différents systèmes et en particulier à travers un réseau type Internet.

1.4.1 XML : un descendant de SGML

SGML (Standard Generalized Markup Language) est un métalangage avec lequel il est possible de définir des langages balisés. Il est lui-même descendant de la recherche sur la théorie des langages et en particulier du langage GML (Generalized Markup Language) d'IBM. Il est devenu par la suite un standard ISO.

XML est un sous-ensemble simplifié du langage SGML (Standard Generalized Markup Language) et est utilisé dans de nombreuses technologies ou protocoles tels que (RDF, RSS, XHTML, Atom, MathML, etc...). Tous les langages dérivés présentent le même avantage : encoder de façon structurée l'information destinée à être partagée à travers un parc d'ordinateurs et de logiciels hétérogènes.

En plus d'avoir donné naissance à XML, SGML est également à la base du langage de balisage HTML, un des éléments clés d'Internet. Par contre, son petit frère, XHTML qui hérite les propriétés de XML, d'où les similitudes entre ces différents langages [5].

1.4.2 Codage des données et interopérabilité

XML permet d'écrire sous forme de fichier texte, dont le codage est exprimé en en-tête, tous les types définis eux-mêmes par un autre fichier XML. Ainsi, grâce à ces deux fichiers (l'un contenant les données, l'autre de description des types de données), n'importe quel logiciel pouvant accéder à des fichiers texte peut analyser les données.

Ce type de technologie est une solution, très utilisée aujourd'hui, pour permettre l'interopérabilité des logiciels à travers Internet. Il est vrai que la syntaxe XML est très verbeuse, ce qui la rend difficilement lisible pour un humain : mais c'est le prix à payer pour une compatibilité maximale et une description au mieux des données pour leur traduction informatique.

Son succès est grandissant, et de nouvelles technologies apparaissent comme AJAX par exemple, une technologie web basée sur la JavaScript, le XML et le XHTML, permettant de travailler les données du côté client pour en envoyer une version complète au serveur sans faire de requête entre chaque petite modification.

1.4.3 Utilisation des fichiers XML

Plusieurs méthodes permettent d'exploiter les données d'un fichier XML. Des méthodes permettent d'accéder à la donnée précise en parcourant le chemin hiérarchique lié au balisage du fichier : ce sont les technologies telles que le langage XPath. Il permet d'exploiter le fichier sans devoir construire une représentation complète en mémoire du fichier. D'autres technologies consistent à construire un arbre représentant le fichier XML dans son entier, les nœuds de cet arbre étant les différentes balises XML. Cet arbre s'appelle l'arbre DOM (Document Object Model). Une telle utilisation est difficilement concevable pour accéder aux quantités de données stockées dans une base de données XML, mais est totalement viable pour analyser un fichier de description simple de service web, par exemple.

1.5 Conclusion

De part l'évolution de la puissance des ordinateurs et de leur mise en réseau, mais également de leur hétérogénéité, les technologies permettant l'interopérabilité des calculs et des échanges de données dans de telles circonstances ont pris une énorme importance, De plus, ces

technologies ont été appuyées (ou créées) par les grands constructeurs tel Sun, IBM, Microsoft. De même, elles ont été normalisées ou standardisées, étapes indispensables aujourd'hui (en passant par un organisme tel que le W3C). Tout ceci a poussé l'adoption par un grand nombre d'industriels et chercheurs de ces nouvelles technologies.

Avant de parler de ces différentes technologies et la façon de les utiliser pour concevoir une plate-forme collaborative à base d'outils de communication synchrones et asynchrones, dans le chapitre qui suit nous allons expliquer quelques concepts généraux sur l'apparition du domaine des TIC (technologie de l'information et de la communication) et l'évolution des plates-formes collaboratives.

Chapitre 2
Les plates-formes collaboratives

2.1 Introduction

Le Groupware, ou logiciel de travail en groupe, ou logiciel collaboratif, désigne une catégorie de logiciels conçus pour une utilisation en réseau par plusieurs individus faisant partie d'une même équipe de travail ou amenés à travailler ensemble.

Cependant un grand nombre de ces outils a été conçu dans le but évident de faciliter la tache aux personnes qui travaillent en réseau, mais il est extrêmement rare que les développeurs de logiciels ou d'outils de communication consultent des utilisateurs pour s'assurer de répondre à l'ensemble des besoins de toutes les clientèles, c'est la raison qui nous incite à développer l'outil de travail qui répond à nos propres besoins.

2.2 Quelques concepts fondamentaux

Malgré la diversité des applications, nous arrivons à identifier certains concepts fondamentaux que les groupewares doivent s'efforcer de prendre en compte afin que les tâches collaboratives et coopératives puissent se dérouler correctement. Bien évidemment, pour qu'un groupeware soit accepté, il faut qu'il offre à ses utilisateurs de bonnes conditions de travail en groupe.

2.2.1 La collaboration et la coopération

Les termes "coopération" et "collaboration", et les concepts qu'ils représentent, sont important dans notre travail. Néanmoins, leur distinction devient rapidement difficile. Dans la littérature ces deux termes sont très souvent utilisés comme synonymes.

Les dictionnaires ne nous aident pas vraiment dans cette tâche : une des définitions pour "collaborer" étant "coopérer".

Dans les travaux d'Ellis [6], nous trouvons cependant une distinction entre ces deux termes. Son "modèle des trois couches" cherche à caractériser la coopération en trois niveaux distincts, selon l'intensité des relations établies entre les individus et les tâches réalisées : la communication, la coordination, et la collaboration. La communication représente le niveau de coopération le plus lâche, la coordination un niveau intermédiaire, et la collaboration représente le niveau de coopération où les tâches sont les plus imbriquées et fortement liées.

Dans “le modèle du trèfle fonctionnel” [7] nous trouvons une classification plutôt fonctionnelle qui n’insiste pas sur une distinction entre coopération et collaboration. La notion de coopération est caractérisée par trois axes ou fonctions principales: la communication, la coordination, et la production.

La communication correspond à l’échange direct d’informations entre les membres d’un groupe. La coordination consiste à définir les règles d’interaction pour contrôler la contribution des membres du groupe qui participent à un travail commun. La production est caractérisée par le partage d’un espace où les membres peuvent stocker, traiter et partager un ou plusieurs objets communs. Dans [7], le lecteur trouvera plus de détails concernant les fonctionnalités relatives à chacun de ces trois axes.

2.2.2 La télé-présence

La télé-présence, ou notion de présence, représente la conscience (en anglais awareness) commune dans un travail de groupe qui permet de définir le contexte dans lequel le travail se réalise. Cette conscience de groupe correspond à la compréhension que chacun a sur :

- avec qui il travaille,
- l’activité de chacun,
- La manière dont les actions de chacun interagissent.

Dans un collecticiel, à la différence d’une application mono-utilisateur, plusieurs personnes peuvent agir en même temps sur les mêmes artefacts de travail. Dans un tel environnement partagé, les acteurs n’ont pas spontanément connaissance des actions des autres. Ainsi la conscience des activités individuelles et des activités du groupe s’avère une information critique pour rendre possible le succès de la collaboration. Elle peut présenter des avantages tels que : la réduction de l’effort de coordination des actions ; l’anticipation des actions des autres; ou même l’aide à des personnes pour s’intégrer aux activités.

Selon Dix [8], la notion de télé-présence se décline sous trois formes :

- Conscience de la présence des membres du groupe et de leur disponibilité dans le travail coopératif ;
- Conscience des actions réalisées par les membres du groupe ;
- Conscience des effets consécutifs à ces actions.

Ainsi, la télé-présence, sous ses différentes formes, permet de contrôler les comportements de chaque participant. En effet, pour pouvoir travailler ensemble il est indispensable d'une part d'être conscient de la présence des autres participants et d'autre part d'être au courant de leur travail.

2.2.3 La session et les artéfacts de travail

L'application collaborative doit fournir des opérateurs permettant de gérer une session, tels que créer et détruire une session ; joindre et quitter une session ; et sélectionner ou modifier les modes de coopération, ainsi que les modes de manipulation des artéfacts de travail.

La manipulation d'artéfacts évoque la manipulation d'objets physiques que l'on déplace ou modifie sur un bureau réel, comme par exemple déplacer un livre, écrire sur une feuille, utiliser son téléphone, etc. Ce concept vise ainsi à déterminer comment les collecticiels peuvent mettre en œuvre le partage d'artefacts de travail (des objets virtuels, des documents, des outils, etc.). Le concept de "bureau virtuel" ou d' "espace partagé" correspond à une des notions les plus répandues dans l'organisation d'artefacts de travail tels que des documents et des outils partagés nécessaires aux tâches à être exécutées par le groupe

2.3 La classification des applications collaboratives

Il est difficile de définir une classification (ou taxonomie) unifiée des applications de travail collaboratif et coopératif. En plus de la grande diversité de ce domaine, chaque classification peut privilégier un aspect particulier. Nous avons ainsi choisi de présenter deux classifications considérées comme incontournables dans la littérature du domaine : l'une concerne des caractéristiques temporelles et spatiales, et l'autre la catégorie fonctionnelle de l'application.

2.3.1 La classification espace-temps

L'espace et le temps constituent les dimensions les plus courantes dans les classifications d'applications collaboratives [6].

La première dimension, l'espace, concerne la distance géographique séparant les utilisateurs de l'application. Par exemple, les membres d'une réunion peuvent se trouver dans une même pièce ou être carrément situés dans des lieux distants (des bâtiments, des villes, ou bien même des pays différents).La dimension temporelle caractérise plutôt le type d'interaction entre utilisateurs.

Les membres du groupe peuvent interagir en même temps et en direct (les actions d'un participant sont immédiatement transmises aux autres), ce qui définit une collaboration synchrone. Dans ce cas, des problèmes de synchronisation et de gestion de la concurrence se posent. Il faut en effet résoudre les problèmes de conflit induits par exemple lors de modifications simultanées et contradictoires réalisées sur une même donnée. Si une gestion de la concurrence n'est pas mise en place, il est fort probable que des conflits et des incohérences se produiront dans le déroulement de l'application. Les problèmes de contrôle de concurrence et de synchronisation ont été beaucoup étudiés : dans le domaine des systèmes répartis conventionnels et dans le domaine du TCAO (où des problèmes d'ordre humain s'ajoutent aux problèmes techniques).

Outre la collaboration synchrone, les membres d'un groupe peuvent également agir à des instants différents, c'est-à-dire, les actions sont espacées dans le temps. Il s'agit alors d'une collaboration asynchrone. Dans ce cas, il est important que l'état de l'activité soit conservé en permanence afin que les membres du groupe soient capables d'observer l'historique des actions qui ont été effectuées avant leur arrivée.

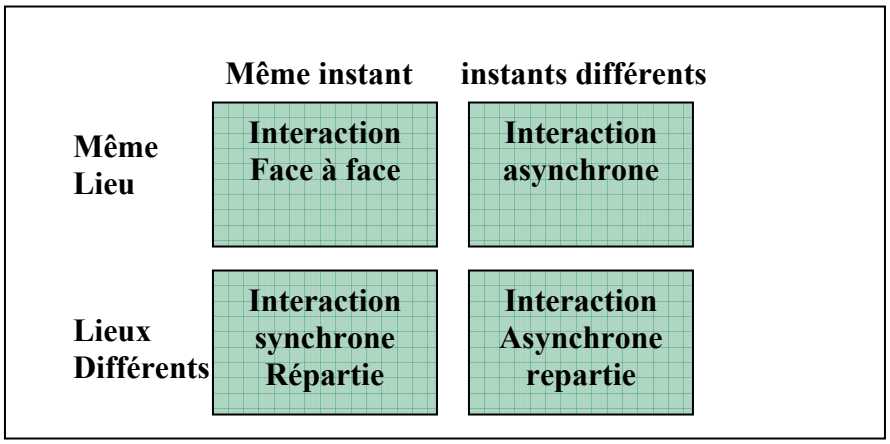

Figure 2.1 : La matrice espace-temps [9]

La Figure 2.1 illustre la “matrice espace-temps”, ou encore matrice de Johansen [9]. De nombreuses critiques ont été suscitées vis-à-vis de cette classification traditionnelle (de plus en plus les applications tendent à couvrir plusieurs quadrants) et plusieurs propositions ont été développées pour affiner cette classification.

2.3.2 Les catégories fonctionnelles d’applications

Beaucoup d’auteurs choisissent d’élaborer une classification par domaines d’application :

- **Le Courrier électronique (courriel) :** Cette catégorie de collecticiels représente de loin le moyen de communication asynchrone le plus répandu et le plus utilisé de nos jours. le courriel désigne le service de transfert de messages envoyés via Internet vers la boîte aux lettres électronique des destinataires choisis par l’émetteur. Une adaptation de ce service de messagerie a connu un grand succès dans le monde de la téléphonie mobile, appelé SMS (en l’anglais Short Messaging Services).
- **La Messagerie instantanée et le forum de discussion :** Comme le courrier électronique, ces deux types de systèmes sont dédiés à la communication par échange de messages. Dans le premier cas (également appelé Chat), contrairement au courrier électronique, la communication est conçue pour être instantanée (synchrone). Le deuxième type de système représente des lieux (normalement des sites Web) où les individus peuvent partager leurs connaissances/expériences par des échanges de messages de manière asynchrone. La plupart des forums sont organisés en fils de discussion où un message ou un thème initial détermine un premier fil de discussion. L’ensemble des discussions est généralement visible par les participants, et éventuellement par tous les internautes.
- **Gestion de flux de processus :** Connue en anglais sous le nom de workflow management, cette catégorie représente des systèmes dédiés à la gestion de processus (industriels, commerciaux, administratifs, etc.) et à la coordination des différents intervenants au cours d’un processus. Également dénommé gestionnaire de procédés, il est souvent défini comme un outil qui prend en charge les documents en cours d’élaboration liés à des procédures et au routage des données. Ces systèmes sont très souvent utilisés par des entreprises dans le but de coordonner les taches exécutées par différents secteurs.

- **Système d'aide à la décision :** Les GDSS (en l'anglais Group Decision Support Systems) sont conçus pour faciliter la prise de décisions en implémentant une sorte de salle de réunion électronique apportant de nombreux outils (par exemple votes, annotation d'idées, etc.)
- **Outil de partage d'application :** Ce type de logiciel permet à plusieurs utilisateurs (travaillant sur des ordinateurs différents) d'utiliser simultanément une application hébergée sur un seul ordinateur (normalement représenté par un serveur).
- **Editeur partagé :** Ces systèmes d'édition conjointe (en l'anglais shared editing) permettent à un groupe d'utilisateurs d'éditer collectivement un document partagé. Ils peuvent être utilisés de façon synchrone ou asynchrone, et ils offrent en général des mécanismes de gestion de versions. Mais la principale complexité de ces systèmes concerne la gestion des tâches concurrentes, lorsque des utilisateurs modifient un même document simultanément.
- **Audioconférence :** Les outils d'audioconférence permettent aux utilisateurs de parler à plusieurs. Si d'un coté la qualité de transmission joue un rôle important pour la compréhension de la communication, les flux audio ne consomment pas énormément de bande passante. La communication audio est l'un des moyens de communication le plus riche au niveau signifiant, mais l'utilisation d'un système d'audioconférence à plusieurs comme seul moyen de communication peut entraîner des difficultés dans l'identification des interlocuteurs.
- **La Vidéoconférence :** Ces systèmes permettent aux utilisateurs distants de se réunir et communiquer par l'intermédiaire d'un support audio et vidéo en même temps. Si la vidéo peut donner une sensation de présence plus forte que l'audioconférence seule, une vidéoconférence nécessite de disposer d'une bande passante capable de diffuser et recevoir des données audio et vidéo avec une qualité acceptable.
- **Agenda partagé :** Également connu comme calendriers partagés (en l'anglais group calendars), ces systèmes permettent de résoudre des problèmes concernant la planification de tâches. Normalement ces systèmes incluent des fonctionnalités telles que la détection d'incompatibilités dans la planification d'une tâche ou la détermination de plages horaires communes aux membres d'un groupe. Cela permet, par exemple, à l'organisateur d'une réunion d'avoir une vision claire de la disponibilité des acteurs d'un projet.

- **La plate-forme collaborative :** Cette catégorie spéciale de collecticiels représente en fait des applications qui rassemblent dans un même environnement intégré différents outils associés aux catégories précédentes. Par ailleurs, pour Schmidt et Rodden [10], le caractère dynamique du travail coopératif et ses articulations sont tels que les efforts devraient se tourner vers la mise au point de ces plates-formes fournissant un large éventail de support pour le travail collaboratif : support de partage et d'échange d'information (partage d'application, éditeur partagé, etc.) ; support pour la communication (messagerie instantanée, vidéoconférence, etc.) ; support pour la prise de décisions, etc.

2.4 Les outils de collaboration asynchrones

Ces outils sont utilisés dans le web mais à des moments différents dans le temps. Ils permettent une communication en continu sans tenir compte des obstacles liés à l'espace et au temps. (Exemple : La messagerie électronique, le forum de discussion, les listes de diffusion, les journaux Web (type blogue, wiki, portails et les outils de transfert de fichiers).

2.4.1 Messagerie ou courrier électronique

Un outil de messagerie ou de courrier électronique est un outil de collaboration en temps différé (asynchrone) qui permet l'envoi de messages (plus rapidement qu'une lettre à la poste) et de joindre des fichiers à ces envois. Déjà plusieurs connaissent des logiciels de messagerie comme : Outlook Express ou la messagerie de Netscape.

D'autres messageries existent sur Internet et sont la plupart du temps gratuites telque Hotmail ou Yahoo. Ce sont des boîtes de messagerie disponibles sur le Web sur des annuaires ou des portails visant à fidéliser des internautes. Chacun peut donc disposer d'une adresse de courrier électronique et peut être rejoint aisément sur Internet. De plus en plus, certains nouveaux outils (notamment les plateformes de formation et les messageries instantanées) ajoutent une messagerie à l'ensemble des fonctionnalités qu'ils mettent à la disposition de leurs utilisateurs. L'utilisation d'une messagerie ou du courrier électronique facilite la distribution des tâches entre les utilisateurs. En outre, ce type d'outil permet d'offrir un soutien particulier à certains participants afin d'accroître leur motivation et leur intérêt.

2.4.2 Forum de discussion

Le forum de discussion désigne un outil collaboratif qui permet d'échanger entre plusieurs usagers ou groupes d'usagers, des messages en mode texte, en temps différé, à n'importe quel moment et à partir de n'importe quel endroit sur Terre (en autant qu'il soit possible d'avoir accès à Internet). Un utilisateur d'un forum peut entrer en communication avec le serveur qui gère le forum et participer à des discussions de groupe sur un thème particulier.

Le principe est simple, un utilisateur écrit un message et d'autres personnes, participant au même forum, vont le lire, répondent ou rédigent de nouveaux messages. Tous les messages apparaissent à l'écran et se présentent selon l'ordre chronologique (du plus récent au plus ancien). Le message d'origine est affiché en premier puis les autres qui ont été rédigés par la suite apparaissent en réponse à ce message, constituant ce qu'on appelle un fil de discussion.

2.4.3 Les outils d'édition

Une gamme d'outils est apparue sur Internet pour permettre à des utilisateurs de rédiger et d'envoyer directement sur une page Web leur texte sans avoir à passer par un long apprentissage des rudiments du code « HTML ». Ces outils permettent donc à chacun de laisser sa marque sur Internet et ainsi d'en démocratiser l'accès. Parmi ces types d'outils :

- **Le blogue :** Un blogue est un outil d'édition qui prend la forme d'un journal de bord. C'est un outil qui permet de rédiger de courts textes placés par ordre chronologique et par thème abordé. L'auteur du texte doit vivre avec les conséquences de ses écrits, alors il faut faire attention à ce qu'on écrit. Certains blogues autorisent la publication d'images pour agrémenter la lecture des textes.
- **Le Wiki :** Un « wiki » est un autre outil d'édition qui permet à plusieurs utilisateurs de rédiger un texte en commun et d'y apporter toutes les modifications voulues. Contrairement aux blogues, qui prennent souvent la forme de journaux avec la date et l'heure de parution de l'article, les « wiki » sont des textes Web qui peuvent prendre différentes formes (on peut décider par exemple d'écrire un roman en groupe avec un « wiki »). Certains « wiki » autorisent la publication d'images. Toutefois, il faut une certaine connaissance de balises simples de mise en page pour l'édition de texte afin de

parvenir à une belle présentation de son texte. Il existerait donc un langage « wiki » un peu astreignant à apprendre.

- **Le portail :** Un portail est un site Web souvent rédigé de manière dynamique et offrant des services destinés à attirer et à fidéliser les internautes au point de devenir leur principale porte d'entrée dans le Web. Un portail propose, non seulement des articles à la communauté qu'il cherche à rejoindre mais, également une série de liens donnant accès à des sites, un moteur de recherche et d'autres services (notamment un courriel gratuit, un forum de discussion, un espace de clavardage « chat », etc.).

2.5 Les outils de collaboration synchrones

Se sont des outils de collaboration qui s'utilisent en temps réel entre des personnes situées à des endroits différents. La communication à distance entre les interlocuteurs se fait donc un peu comme au téléphone c'est-à-dire en temps réel. Le but de ce type de communication est d'échanger des informations le plus rapidement possible pour décider de la suite des événements ou encore de réunir différentes personnes pour les informer en même temps de certaines informations à consigner (Exemple : réunions, cours). Parmi ces types d'outils de collaboration synchrones, on peut citer : la vidéoconférence, la téléphonie par Internet, la séance de clavardage, l'utilisation de tableau partagé ou (« whiteboard »).

2.5.1 La vidéoconférence ou la visioconférence

Les deux technologies ont la caractéristique commune de transmettre à distance la voix et l'image et de permettre à des personnes situées dans des lieux distants de se voir et se parler comme si elles étaient en présence les unes des autres. En dehors de cette particularité qui les rapproche, vidéoconférence et visioconférence diffèrent de bien des façons. L'explication de ces différences réside dans l'histoire de leur évolution.

Dans les années 70, apparaît la vidéoconférence qui achemine les données par des canaux satellitaires, et notamment le satellite canadien Anik lancé en 1972 [11]. La grande percée technologique de cette époque est que la vidéoconférence est bidirectionnelle. Les données audio et visuelles circulent dans les deux sens alors que la télévision ne diffuse l'information que dans le sens émetteur récepteur.

Dans les années 80, la vidéoconférence devient multidirectionnelle, grâce à des ponts servant de relais. Plusieurs sites se branchent sur un pont qui retransmet les données; ainsi, des participants en Afrique peuvent-ils se joindre à d'autres en Amérique du Nord et en Europe.

La visioconférence naît plus tard au milieu des années 90. Elle emprunte les mêmes canaux numériques que le réseau Internet. Alors que la vidéoconférence demande des salles aménagées particulièrement pour cette fonction, la visioconférence ne demande chez l'usager qu'un micro-ordinateur personnel équipé d'une caméra, d'un casque d'écoute, d'un micro, et un accès Internet.

Autour de 1995, MicroSoft et Cornell University mettent chacun au point un logiciel de visioconférence appelé respectivement NetMeeting et CuSeeMe [12] [13]. Ces deux logiciels sont offerts gratuitement aux internautes, sans aucun soutien à l'usager. Leur performance est faible car elle est liée à la capacité des canaux par lesquels sont transmises les données, et le réseau Internet de cette époque est encore peu performant. La bande passante est insuffisante pour que les données soient transportées de façon uniforme. Par exemple, la voix par NetMeeting, n'est possible qu'à deux utilisateurs. Lorsqu'un troisième se branche au système, la communication orale disparaît, et c'est par clavardage que les utilisateurs poursuivent leur conversation. NetMeeting présente cependant déjà des possibilités de partage de sites Web, d'un tableau blanc et d'applications logicielles. Ces possibilités de partage, malgré leur peu de fiabilité, laissent entrevoir les possibilités futures lorsque les problèmes de capacité des réseaux seront résolus. Depuis, la technologie évolue constamment, permettant des utilisations de plus en plus diversifiées.

Voyons concrètement ce dont il s'agit.

2.5.1.1 Les utilisations

À ses débuts, la vidéoconférence est utilisée dans les entreprises et les universités pour des rencontres de groupe. Les entreprises y voient l'avantage de tenir des réunions de travail virtuelles alors que les participants à ces réunions ne se trouvent pas physiquement ensemble au moment de la rencontre. Les coûts reliés à la vidéoconférence, opérée souvent par des compagnies de télécommunication bien qu'importants, sont cependant souvent moindres que ceux reliés aux déplacements des personnes. Les universités utilisent aussi cette technologie pour

permettre l'accès à des ressources qualifiées de « rares ». On ne trouve pas de professeurs spécialisés dans toutes les régions du pays, et c'est également pour contrer cette même rareté de ressources que des systèmes de vidéoconférence sont installés dans plusieurs écoles dans plusieurs pays. On parle alors de formation à distance selon un modèle qui rassemble les apprenants dans des salles équipées de systèmes de vidéoconférence et reliées par des infrastructures de télécommunication dédiées à la transmission des données.

2.5.1.2 Quelques logiciels de visioconférence

iVisit : iVisit est un logiciel de vidéoconférence de type P2P (poste à poste) distribué gratuitement depuis 1997. Il permet de communiquer de façon interactive avec des interlocuteurs en utilisant l'image, le son et l'écrit. Ce logiciel fonctionne aussi bien sur Macintosh que sur PC (Win95/98/2000/ME/XP/NT) et ne requiert aucun serveur. Celui ci, s'il est utilisé, joue le rôle d'un annuaire des interlocuteurs en ligne. iVisit permet également de participer à des conférences protégées par mot de passe [14].

INRIA Videoconferencing System(IVS) : IVS fut l'un des tous premiers logiciels de téléconférence sur l'Internet. Ce logiciel fut développé par Thierry Turletti [15] de l'équipe RODEO à l'INRIA Sophia-Antipolis. Une des originalités de son approche était de mettre en œuvre un codec H261 logiciel [16]. Mais malheureusement son architecture logicielle est très complexe et peu propice à des extensions.

Cégeps en réseau : Le projet « Cégeps en réseau » [17], est un projet d'innovation techno-pédagogique et organisationnelle qui vise à répondre aux défis posés par l'enseignement à de petites cohortes d'étudiants dans des programmes de formation technique de niveau collégial. Ce projet recourt aux TIC (technologies d'information et de communication) pour établir différentes formes de collaboration entre établissements disposant d'un même programme de formation technique, touchés par des problématiques liées aux petites cohortes. Pour que le projet réussisse, la collaboration doit s'établir à plusieurs niveaux : entre enseignants, entre répondants TIC, entre cadres, entre établissements d'enseignement, et aussi, entre étudiants.

2.5.2 Fonctionnalités de communication écrites (clavardage ou chat)

Le clavardage (ou « chat ») est un outil de communication synchrone, ou en temps réel, qui permet une interaction en simultané entre les usagers sur Internet. Il y est possible de visualiser en direct les messages transmis par d'autres personnes et d'y répondre tout de suite. Ce type de rencontre virtuelle convient bien aux échanges nécessitant une rétroaction immédiate.

En terme d'efficacité, ce type d'outil de communication est plus adéquat avec des groupes restreints d'utilisateurs, car à plusieurs il peut devenir une vraie Tour de Babel.

Dans certains cas, il est possible de garder en mémoire ou de sauvegarder le contenu des messages pour pouvoir les consulter à nouveau par la suite. Ce type d'outil est souvent offert en convergence et ne représente souvent qu'une partie d'un ensemble plus large (ex. : plate-forme de formation, logiciel de vidéoconférence ou messagerie instantanée).

Dans une optique d'utilisation pour l'apprentissage collaboratif à distance, ce type d'outil permet à des participants de plusieurs ateliers disséminés sur un territoire de se rencontrer pour faire connaissance. Il permet des interventions plus informelles et plus spontanées qu'avec les outils asynchrones comme les forums de discussion. Cependant, il peut s'avérer frustrant au plan de la communication écrite, car pour l'utiliser il faut pouvoir dactylographier assez rapidement. En outre, il n'est pas toujours possible de pouvoir gérer les moments de rencontre (décalage horaire, retard dans la connexion, problèmes de connexion, etc..).

2.5.3 Le Tableau blanc partagé

Le système de tableau blanc partagé, comme son nom l'indique, met à disposition une surface de dessin accessible par plusieurs utilisateurs. Il permet ainsi à des utilisateurs de travailler d'une manière synchrone sur des documents 2D (voir Figure 3).

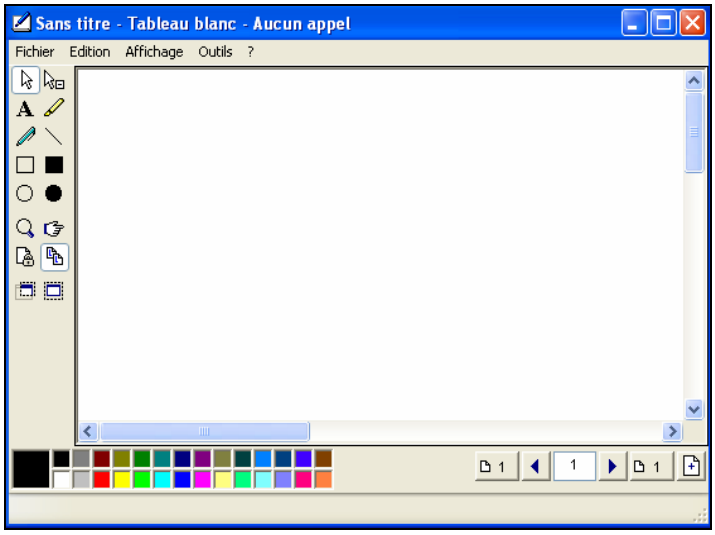

Figure 2.2 : Le Tableau blanc

2.6 Les plates-formes collaboratives libres (open source)

Concevoir une plate-forme collaborative qui offre des solutions de collaboration et de coopération à un nombre considérable de personnes était et restera une tache très difficile, cette complexité due à plusieurs raisons :

- La complexité du code.
- Le manque de temps.
- Développer une plate-forme complète cela oblige le développeur à connaître un maximum de technologies et à maitriser un maximum de techniques de développement.

Pour remédier à ces problèmes, on va se baser sur ce qu'on appelle le logiciel libre (en anglais open source), c'est-à-dire au lieu de développer une plate-forme des la première ligne de code, on va essayer de trouver une méthodologie ou une technique pour fusionner deux codes sources existant et qui répondent à nos besoins.

A présent on va définir la notion de logiciel libre avec son modèle de développement, puis on présentera un ensemble de plates-formes collaboratives et open sources que nous avons expérimenté et qui offrent des outils de collaborations qui peuvent être exploité dans n'importe quel domaine (éducatif, entreprise, gestion des connaissances, etc.).

2.6.1 La définition du logiciel libre

La notion de « logiciel libre » a émergé au cours des années 1980. Cette expression se réfère plus aux droits d'utilisation qu'aux prix des logiciels. En effet, « libre » ne signifie pas «gratuit».

Les deux caractéristiques principales d'un logiciel libre sont, d'une part, son code source ouvert et, d'autre part, la licence qui le régit.

En général, un logiciel est dit libre si les quatre libertés suivantes sont assurées :

- La liberté d'exécuter le programme, quelle que soit l'utilisation.
- La liberté d'étudier le fonctionnement du programme et de l'adapter aux besoins de l'utilisateur.
- La liberté de distribuer des copies du logiciel.
- La liberté d'améliorer le logiciel et d'en faire profiter la communauté (en premier distribuant).

Ces différentes libertés sont garanties par la licence qui régit chaque logiciel. Il existe plusieurs licences dans le monde du logiciel libre; elles spécifient le cadre d'utilisation des logiciels ainsi que les différentes permissions (étude, modification et redistribution) de leurs utilisateurs. La plus connue des licences est la licence GPL [42] qui compte de multiples dérivées. Parmi les licences importantes on compte également la licence BSD [42] et la licence Apache (voir glossaire).

Pour étudier et modifier un logiciel, il est nécessaire de disposer du code source du logiciel: on dit alors que le logiciel qui permet cette pratique est un logiciel libre.

2.6.2 Modèle de développement du logiciel libre

Le modèle de développement des logiciels libres est essentiellement collaboratif. On parle d'ailleurs beaucoup de « communauté » du logiciel libre. Un projet de logiciel libre rassemble généralement plusieurs développeurs qui peuvent éventuellement être distants les uns des autres.

Le logiciel libre n'est pas uniquement le fruit des développeurs isolés et en marge du monde informatique, au contraire, de nombreuses sociétés, reconnues dans le domaine, investissent pour

développer du logiciel libre. C'est le cas par exemple au sein de Red Hat, Mandrake [18], et dans une certaine mesure, dans des compagnies reconnues, telles qu'Oracle ou Sun.

En effet, la communauté open source peut compter sur une collaboration avec des compagnies qui développent du logiciel propriétaire tout en utilisant des solutions open source. Ainsi, Oracle utilise le serveur Apache et fournit du support pour ses clients sur de tels logiciels libres. Donc dès qu'un problème est détecté et résolu, le produit libre se trouve amélioré et la communauté du logiciel libre profite de cette expérience. Chacun peut ainsi librement apporter sa contribution au développement de ce type de logiciels.

2.6.3 Quelques plates-formes collaboratives open source

Durant notre travail d'analyse des besoins et de l'existant nous avons été obligé d'expérimenter et d'installer un certain nombre de plates-formes, pour essayer de donner une nouvelle approche basée sur leurs différentes architectures.

Parmi ces plates-formes on peut citer :

2.6.3.1 Moodle

Qui est une plateforme open source d'enseignement à distance développée en PHP et qui permet aux enseignants de créer et de gérer très facilement un cours sur Internet, sans nécessiter de compétences informatiques particulières.

Elle offre entre autres des outils de communication (forums, chat), des instruments d'évaluation (exercices, sondages, travaux) et la possibilité de déposer des documents (pages web, fichiers PDF, présentations Powerpoint, séquences vidéo, etc.).

Cette plate-forme peut aussi offrir un outil de vidéoconférence, cela se fait par l'intégration d'un autre outil open source qui s'appelle Dimdim (voir figure suivante) [19].

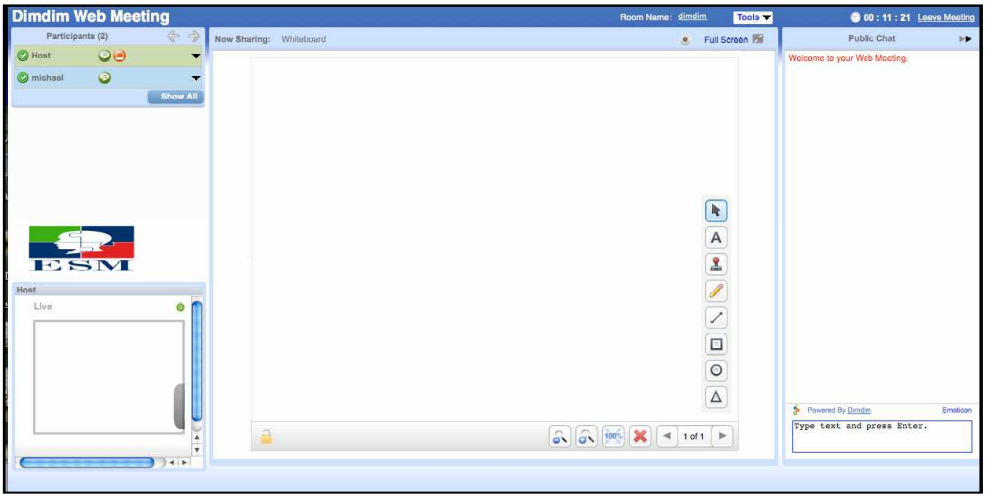

Figure 2.3 : L'intégration de Dimdim dans moodle

2.6.3.2 Sakai

Sakai est un environnement numérique d'apprentissage collaboratif et en open source développé en java par quatre universités américaines (University of Michigan, Indiana University, Massachussets Institute of Technology et Stanford University) [20].

L'implémentation de cette plate-forme est basée sur un ensemble de frameworks tels que Jakarta, Hibernate et Spring tout ça pour former une architecture basée services qui offre des possibilités de communications entre plusieurs Services Web.

Voici l'interface générale de la plate-forme :

Figure 2.4 : L'interface de la plate-forme Sakai

Cette plate-forme offre un ensemble d'outils de collaboration tel que les salles de discussion, la vidéoconférence et le tableau blanc partagé par l'intégration d'un autre outil open source qui s'appelle agora, un forum de discussion, wiki, envoyer des emails et d'autres outils pour la gestion des cours, l'organisation des examens, etc.

2.6.3.3 Open-Xchange

Open-Xchange [21] est un environnement de collaboration open source développé en java permettant l'enregistrement de contacts, de rendez-vous, de taches, de signets, de documents, etc. Cet environnement peut être utilisé par un navigateur web ou par de multiples clients lourds Voici l'interface générale de la plate-forme :

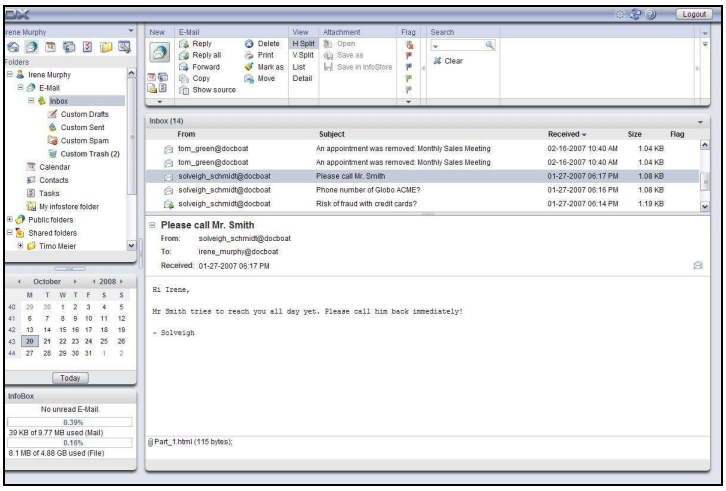

Figure 2.5 : L'interface d'Open-Xchange

Cette interface est décomposée en plusieurs modules, on va citer quelques uns :

Un portail : qui permet d'avoir un aperçu de tous les éléments concernant l'utilisateur, y compris les emails, calendrier et contacts et un accès direct à toutes les informations personnelles comme les tâches, les réunions et les projets en cours.

Un Calendrier : permet de coordonner les rendez-vous personnels et de groupes avec les informations de disponibilité de chaque utilisateur et chaque ressource.

Tâches : pour la gestion et l'administration des tâches individuelles.

Projets : est conçu pour organiser les équipes qui travaillent sur des projets en planifiant les réunions, les tâches, et en gérant les documents rattachés au projet.

Gestion de document : qui inclut des fonctionnalités de gestion de version, de verrouillage pendant l'édition des documents en cas d'utilisation des applications MSOffice et MS Internet Explorer.

Base de connaissances : un module pour la gestion des connaissances entre les employés: l'utilisateur peut construire une base de connaissances centralisée, gérer les signets et discuter dans les forums.

On a aussi testé plusieurs autres logiciels qui se basent d'une manière particulière sur les outils de collaborations synchrones, parmi ces plates-formes on peut citer : conferenceXP développée en c# qui offre un outil de vidéoconférence, un tableau blanc partagé et un bureau partagé, Openmeetings qui offre un maximum d'outils de collaboration synchrones. Dans les chapitres suivants on expliquera en détail l'architecture de cette plate-forme qui joue un rôle très important dans l'approche qu'on va proposer.

2.7 Différentes approches de développement

Différentes approches ont été utilisées dans le développement d'applications collaboratives. Nous allons décrire ces approches tout en identifiant comment elles peuvent influencer l'acceptation des applications (développées selon ces approches) au sein de véritables groupes de travail.

2.7.1 Le développement from scratch

Le développement from scratch concerne la conception complète d'une nouvelle Plate-forme collaborative "à partir de rien", c'est-à-dire, sans faire appel à des composants ou des

sous-systèmes implémentant des fonctionnalités générales ou spécifiques. Le principal avantage de cette approche est que le développeur est libre de prendre ses propres décisions concernant la spécification, la modélisation et l'implémentation de l'application.

Puisque ces systèmes représentent souvent des solutions propriétaires ne s'appuyant pas sur des modèles de conception ouverts, ils risquent fort de présenter les inconvénients classiques des applications fermées : non-standardisation, manque de flexibilité et d'interopérabilité. En outre, le fait de développer tout un système sans faire appel à des solutions pré-existantes peut conduire à "réinventer" la roue, sans parler de l'augmentation des coûts et de la fiabilité, probablement inférieure, du système. Finalement, puisque toutes les fonctionnalités collaboratives ont été statiquement imbriquées dans l'application, cela peut représenter une vraie surcharge architecturale étant donné que certaines fonctionnalités peuvent même ne pas être utilisées.

Il s'agit d'une approche en général utilisée pour le développement d'applications qui ne présentent pas trop de complexité vis-à-vis du nombre de fonctionnalités collaboratives offertes (par exemple CoLab [22]).

2.7.2 Développement s'appuyant sur des boîtes à outils

Une boîte à outils (en anglais toolkit) est une bibliothèque de composants logiciels en charge de gérer des structures de données et de réaliser des opérations communes pour le développement d'applications. Ces composants sont en général accessibles par un programme via des appels procéduraux. L'objectif d'une boîte à outils est ainsi de fournir un cadre de programmation adéquat pour implémenter certaines parties d'une l'application tout en réduisant l'effort de programmation.

L'approche de développement s'appuyant sur des boîtes à outils, comparée à l'approche de développement from scratch, présente l'avantage de réduire le coût de développement et de faciliter le prototypage rapide de nouveaux collecticiels. Par ailleurs, l'utilisation d'une boîte à outils permet de concevoir des collecticiels variés et adaptés aux besoins des utilisateurs, tout en gardant un environnement homogène, ce qui peut faciliter leur utilisation.

2.7.3 Le développement s'appuyant sur des frameworks et des plates-formes

De nombreux projets de recherche se fixent comme objectif de construire des frameworks ou des plates-formes pour le développement d'applications collaboratives. Il s'agit en général de solutions plus génériques que les boîtes à outils qui cherchent notamment à supporter des propriétés telles que la flexibilité et l'extensibilité.

Les frameworks, couramment appelés en français "architectures cadre" ou "cadre d'application", fournissent une structure réutilisable qui spécifie les fondements de l'application. Normalement, cette structure se présente sous la forme d'une architecture générique ou conceptuelle. Les frameworks peuvent également être associés à la notion de pattern (ou "patron") qui décrivent des solutions reconnues pour une classe de problèmes.

Une plate-forme offre en général un support générique d'exécution et de développement de collecticiels en définissant une architecture et des briques logicielles implémentant des mécanismes de contrôle pour la collaboration. Certaines plates-formes sont développées en s'appuyant sur des Framework (par exemple Sakai [20]), Néanmoins, une distinction consensuelle entre Framework et plates-formes s'avère difficile, surtout quand nous trouvons le terme Framework désignant des environnements fournissant un ensemble de bibliothèques ou briques logicielles pour permettre le développement d'applications (par exemple [.NET]). Nous allons ainsi employer ces termes selon les critères choisis par chaque auteur des travaux cités par la suite.

Différents auteurs considèrent que les collecticiels de demain seront davantage des assemblages de composants, Ils partent du principe que la plupart des fonctionnalités requises existeront déjà sous forme de composants développés par des tiers. En s'appuyant sur ce principe d'assemblages de composants, on va proposer dans les chapitres qui suivent une infrastructure collaborative extensible basée sur les services web intégrant des outils de collaborations synchrones avec un outil de gestion des connaissances. Les systèmes peuvent définir des environnements collaboratifs complètement intégrés.

2.8 Conclusion

Dans ce chapitre nous avons présenté les concepts fondamentaux liés à la collaboration et aux plates-formes collaboratives, ainsi qu'un état de l'art sur notre problématique de recherche qui se lie au développement de telles applications. Nous avons présenté un ensemble d'outils de collaboration (synchrones et asynchrones) ainsi qu'un ensemble de plates-formes collaboratives libres pour pouvoir situer notre approche de développement.

Dans le chapitre suivant nous allons présenter une nouvelle architecture logiciel qui se base sur les services cette architecture s'appelle architecture orientée services.

Chapitre 3
L’architecture orientée services

3.1 Introduction

Au cours des quarante dernières années, les systèmes informatiques se sont développés de manière exponentielle, sous forme d'architectures logicielles de plus en plus complexes et difficiles à gérer. Les architectures traditionnelles ont atteint leurs limites en termes de capacité, alors que les besoins traditionnels des organisations informatiques demeurent. Les départements informatiques doivent toujours répondre rapidement aux nouvelles exigences des entreprises, tout en réduisant constamment le coût informatique de l'entreprise et en absorbant et en intégrant de manière transparente de nouveaux partenaires et clients commerciaux. Le secteur d'activité des logiciels a connu de nombreuses architectures conçues pour permettre un traitement entièrement distribué, des langages de programmation destinés à n'importe quelle plate-forme et une myriade de produits de connectivité conçus pour permettre une intégration plus rapide et plus efficace des applications, et pour réduire considérablement les temps de mise en œuvre. Toutefois, une solution globale ne semble être encore qu'une utopie.

Pour remédier à ces problèmes, une architecture orientée services s'est présentée comme l'étape suivante de l'évolution qui permet d'aider les organisations informatiques à relever des défis toujours plus complexes. Cependant, Une question se pose : les architectures orientées services sont-elles réelles ? Si elles peuvent être conçues et décrites, peuvent-elles être mises en œuvre ?

En suivant ce concept mais dans un autre cadre c'est-à-dire dans le but de concevoir une infrastructure collaborative intégrant des outils de collaboration existants sous forme de services, on a choisi d'utiliser ce type d'architecture.
Dans ce chapitre on va essayer de donner une vue globale sur l'apparition des différentes architectures jusqu'à l'arrivé de la SOA, on expliquera pourquoi on a choisi une telle architecture en démontrant que les possibilités offertes par cette dernière ne sont pas uniquement des promesses.

3.2 L'évolution de l'architecture

L'évolution de l'architecture est la conséquence de principaux facteurs tels que :

- La prise de conscience que la méthode précédente était imparfaite (souvent due à une évolution des besoins de l'utilisateur également) ;
- L'évolution de la capacité des machines (tant en mémoire qu'en puissance de calcul) se traduisant par l'évolution de leurs fonctionnalités ;
- Mais également la complexité des systèmes informatiques et leurs coûts.

Nous allons étudier les trois évolutions majeures de l'architecture de programmation : la programmation modulaire, la programmation orientée objet et enfin la programmation orientée composant. La section suivante présentera l'étape suivante : l'architecture orientée service et les technologies des services web.

3.2.1 Architecture basée sur la programmation modulaire

Les principales directives de la programmation modulaire sont :

- Supprimer la programmation reposant sur les instructions « goto » permettant de faire des sauts en avant ou en arrière dans le code. Ceci en privilégiant l'utilisation de boucles tels que while, repeat ... until, for, etc...
- Découper le code en module cohérents (s'apparentant généralement à des fichiers) : un module traite un type de donnée abstrait comme une liste, ou un type « concret » comme les informations caractérisant une personne par exemple.
- Éviter les variables globales que tout le code peut accéder à n'importe quel moment et modifier sans aucune vérification.
- Utiliser des fonctions ou des procédures pour structurer le code : une fonction ou procédure pour un traitement bien précis sur un type de donnée bien précis. De plus, l'utilisation des fonctions permet de « factoriser » le code.
- Utiliser la compilation séparée : un module est compilable de façon autonome, et l'ensemble des fichiers compilés seront liés ensemble pour devenir un code exécutable.

- Même si les réseaux n'étaient pas aussi utilisés qu'aujourd'hui, le concept d'utilisation et d'invocation à distance, dans le cadre d'un système réparti était déjà présent. Les programmeurs utilisent, dans le cadre de la programmation modulaire, les mécanismes Remote Procedure Call (RPC).

Le mécanisme RPC permet de développer une application basée sur le modèle client-serveur. Un appel à distance est effectué par le client en envoyant un message à un système distant (serveur), pour exécuter une procédure donnée en utilisant les arguments fournis. Le résultat (calculé par le serveur) est ensuite envoyé au client.
De nombreuses implémentations sont nées de cette norme, et beaucoup sont incompatibles entre elles. Mais le principe était là : le client et le serveur pouvaient utiliser un encodage des données différent, un proxy réalise une conversion juste après l'appel de la fonction par le client, juste avant l'envoi effectif au serveur ou lors de la réception du message de réponse.

3.2.2 Architecture basée sur la programmation orientée objet

Parallèlement à l'arrivée de la programmation structurée, la programmation objet est arrivée au début des années 80. La programmation objet reprenait le concept du regroupement des données et des fonctions (appelées méthodes en terminologie objet), pour obtenir un seul module: une classe, qui, une fois instanciée, devient un objet.

Deux avantages principaux à cette architecture :

- Il existe une notion de regroupement du code et des données dans un seul et même objet, et ce, en forçant, si nécessaire leur non accessibilité depuis l'extérieur de cet objet : cela permet, par exemple, d'empêcher la modification d'une variable par du code n'appartenant pas à l'objet. Ce principe renforce le principe de séparation modulaire, de compilation séparée et de portée des variables dans les procédures et fonctions issue de la programmation modulaire.
- Il existe un système d'héritage permettant de raffiner le comportement d'une classe : par exemple, une personne peut être un étudiant, un professeur ou un stagiaire. Cet héritage fournit des mécanismes tels que le polymorphisme et l'encapsulation.

SmallTalk [23] fait parti des langages objets cités, le plus souvent, comme étant le plus fidèle du principe objet. Mais suivant les langages, des notions différentes peuvent apparaître. Par exemple, le C++ utilise le concept d'héritage multiple alors que le Java ne permet que l'héritage simple.

Une classe fournit donc un ensemble complet de méthodes permettant de traiter des données. Le programmeur peut alors :

- Instancier un nouvel objet (qui fera appel aux constructeurs de la classe pour l'initialiser),
- Accéder aux méthodes publiques (ou privées à l'intérieur d'un objet) pour obtenir ou modifier les propriétés de l'objet
- Utiliser un mécanisme de destruction de l'objet (dont le comportement varie fortement d'un langage objet à un autre).

Au niveau des mécanismes de répartition, on parle alors d'objets répartis. Ce sont des objets de « base » qui évoluent dans une architecture répartie, bien souvent à base de bus (Object Request Broker ou ORB dans le cadre de CORBA et Java RMI dans le cas de Java). Ce bus est une partie logiciel faisant partie d'un intergiciel (ou middleware), visant à s'affranchir des problèmes liés à la répartition, en gérant l'interopérabilité ou encore la portabilité des objets.

Pour permettre la portabilité du composant, la description de la partie dépendante de la plateforme est faite dans un langage portable, et un générateur de code produit du code spécifique pour une plateforme cible : ce code servira de passerelle entre les clients et l'application métier (mécanisme RMI de Java par exemple).

3.2.3 Architecture orientée composant

L'architecture orienté objet et ses objectifs initiaux ne permettent pas d'utiliser la puissance et les nouvelles méthodes issues de l'évolution des architectures (grilles de calcul, l'utilisation du pair à pair, architecture 3-tiers ou même n-tiers) tendant toujours vers plus d'uniformisation tout en augmentant la complexité des modèles, et ce dans le but de séparer la logique métier et la logique système. Une nouvelle architecture est apparue progressivement répondant à ces besoins : l'architecture orientée composants.

Cette nouvelle architecture tend à proposer des solutions aux problèmes courant rencontrés dans les systèmes réparties qui sont :

- La possibilité d'utiliser à large échelle les solutions proposées, par exemple à travers Internet,
- La gestion de l'hétérogénéité des systèmes,
- La prise en compte des communications asynchrones et la gestion du contrôle concurrent,
- La prise en compte des pannes partielles concernant le logiciel, le réseau (surcharge, coupure temporaire, etc.).
- Et enfin, un élément indispensable : la gestion de la sécurité [24].

La prise en compte de ces problèmes aboutit à des solutions basées sur les bus à objets vu précédemment. Elles sont généralement englobées dans un Framework, c'est-à-dire une boîte à outils permettant de « tout » faire. Ces frameworks peuvent être à la fois visuels (par exemple, un IDE de type RAD –développement rapide d'applications–) ou abstraits gérant le déploiement à travers un réseau des différents composants, pouvant même gérer, dans certains cas, les différentes versions de ceux-ci.

3.3 Les problèmes liés à l'intégration d'applications

Depuis que CORBA a pensé que l'intégration d'applications sur des plates-formes hétérogènes était possible, beaucoup de problèmes sont apparus et deviennent de plus en plus complexes chaque année.

Parmi ces problèmes on peut citer :

3.3.1 La complexité

Aujourd'hui, certains aspects informatiques ont changé, les environnements sont plus complexes, l'accès à faible coût et omniprésent à Internet a créé de nouveaux modèles de développement surtout pour les entreprises qui souhaitent fusionner des applications entières qui répondent à leurs besoins.Dans un environnement d'une telle complexité, les solutions point à point ne font qu'exacerber les problèmes et ne relèvent jamais véritablement les défis. Donc il faut développer des systèmes qui incorporent le caractère hétérogène des organisations, en tant que composant essentiel de l'environnement informatique, de sorte que ces systèmes puissent

prendre en charge une gamme diverse et sans limite de matériel, de systèmes d'exploitation, de logiciels intermédiaires, de langages et de stockage de données.

3.3.2 Une programmation redondante et ne pouvant être réutilisée

Comme dans beaucoup de sociétés, l'éventail des applications a peut-être augmenté en raison de fusions ou d'acquisitions d'autres entreprises. Le résultat est de faire face à des applications redondantes ou à des applications dont les fonctions ne peuvent être facilement réutilisées. Il se peut que chaque unité de l'organisation ait travaillé indépendamment des autres, ce qui freine l'effort de coordination visant à créer un parc ou des services informatiques fonctionnels et réutilisables. Cette redondance augmente les coûts et le temps de mise sur le marché pour le déploiement de nouveaux produits et services, car les modifications doivent être effectuées pour chaque application ou système concerné. Le fait de ne pouvoir réutiliser certaines fonctions requiert, en fin de compte, plus de ressources, et souvent plus de temps, pour livrer de nouvelles applications.

3.3.3 Des interfaces multiples

Il faut penser au problème d'intégration n**.**(n-1).Les développeurs d'entreprises par exemple doivent faire face à des problèmes d'intégration, quelle que soit leur nature, en raison d'une fusion, d'un nouveau partenariat commercial ou simplement d'un besoin d'interconnexion entre des systèmes existants. Si n systèmes d'applications doivent être directement interconnectés, le processus va créer n**.**(n-1) connexions ou interfaces. Dans la figure suivante, chaque pointe de flèche représente une interface.

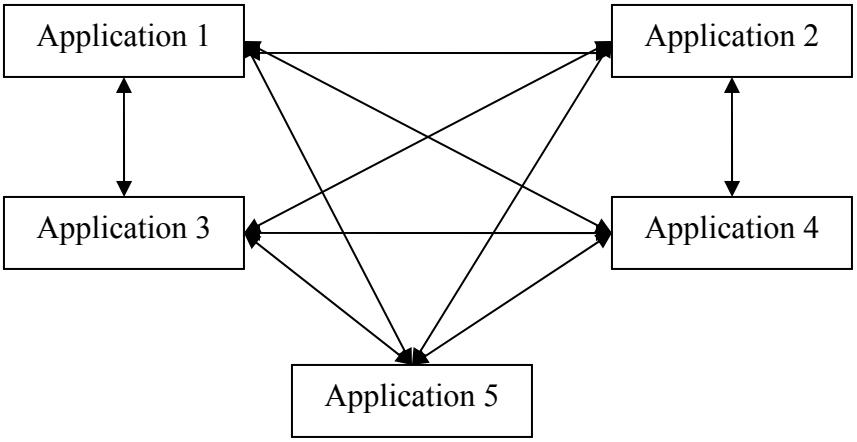

Figure 3.1 : Le problème d'intégration

Par conséquent, si on a besoin d'intégrer un autre système d'applications A (n+1), on doit créer, documenter, tester et maintenir 2n interfaces nouvelles. Dans la figure précédente, l'ensemble de cinq applications requiert 20 interfaces directes. L'ajout d'une sixième application supposerait dix nouvelles interfaces. Et pour augmenter la complexité de l'opération, on doit également modifier le code de chaque application.

3.4 Les exigences de la programmation d'aujourd'hui

Au vu des problèmes débattus précédemment, il devrait être évident qu'il est important de développer une architecture qui satisfasse un ensemble d'exigences. Ces exigences devraient inclure les éléments suivants :

3.4.1 L'exploitation du parc informatique existant

Il s'agit de la condition la plus importante. Il est rare que des systèmes existants puissent être éliminés. Ils contiennent certainement des données de grande importance. L'objectif est de constituer une nouvelle architecture dont le rendement corresponde à ce qu'on souhaite.

3.4.2 La prise en charge de tous les types d'intégrations requis

Cela comprend l'interaction avec l'utilisateur (pour fournir un fonctionnement unique et interactif), la connectivité des applications (pour créer un modèle de communication qui sous-

tend toute l'architecture), l'intégration des processus (pour organiser les applications et les services), l'intégration des informations (pour rassembler et déplacer les données) et la possibilité d'effectuer de futures intégrations (pour créer et déployer de nouveaux services et applications).

3.4.3 La possibilité de mises en œuvre incrémentielles et de migration du parc

Si cette exigence est satisfaite, l'un des aspects essentiels du développement de l'architecture pourra être mis en place : la possibilité de produire un retour sur investissement différentiel. D'innombrables projets d'intégration ont échoué en raison de leur complexité, de leur coût et de temps de mise en œuvre impossibles à réaliser.

3.4.4 La constitution dans un cadre de composants standards

On doit inclure un environnement de développement créé autour d'un cadre de composants standards pour promouvoir une meilleure réutilisation des modules et des systèmes, permettre au parc informatique existant de migrer vers le cadre en question et permettre une mise en œuvre opportune des nouvelles technologies.

3.4.5 La possibilité de mise en œuvre de nouveaux modèles informatiques

Des exemples spécifiques incluent les nouveaux modèles clients de portails, le calcul distribué et l'informatique à la demande.

L'architecture qui répond à ces exigences s'appelle « Architecture Orientée Services », elle permet de concevoir des systèmes logiciels fournissant des services à d'autres applications à l'aide d'interfaces publiées et découvrables, et où les services peuvent être sollicités via un réseau.

Ce nouveau model d'architecture est apparu avec l'apparition de ce qu'on appelle les services web, car son principal objectif était de publier (sur internet, un extranet ou encore un intranet) un ensemble d'offre (existant ou non) réalisant un service bien précis (par exemple un service pour crypter des données, un autre pour authentifier une personne, etc.…).Une apparition dans ce cadre appelle ces différents services quand elle en a besoin et peut être elle-même déployée comme un nouveau service.

3.5 Une architecture orientée services : au-delà des services Web

L'évènement des services Web a précipité le changement fondamental dans le développement, le déploiement et la gestion des infrastructures informatiques. La réussite de nombreux projets de services Web a montré que la technologie permettant de mettre en œuvre une véritable architecture orientée services existe. Elle permet de prendre du recul pour examiner l'architecture des applications. D'un point de vue de l'entreprise, le problème n'est plus uniquement un problème technologique. Il s'agit de développer une architecture et un cadre d'applications au sein desquels on pourra définir des problèmes et mettre en œuvre des solutions qu'on pourra réutiliser de manière cohérente.

Toutefois, il est important d'abord de comprendre qu'une architecture orientée services ne se limite pas aux services Web. Les services Web constituent un ensemble de technologies, y compris les technologies XML, SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) et UDDI (Universal Description, Discover and Integration), qui nous permet de créer des solutions de programmation pour des problèmes de messagerie et d'intégration d'applications spécifiques. Ces technologies sont suffisantes et ont déjà démontré qu'on peut dès à présent mettre en œuvre une architecture orientée services.

3.5.1 Les éléments constitutifs d'une architecture orientée services

Une architecture orientée services porte bien son nom car il s'agit en effet d'une architecture. Ce n'est pas uniquement un ensemble spécifique de technologies, telles que les services Web. Elle transcende ces technologies et, en théorie, en est complètement indépendante. Au sein d'un environnement d'entreprise, une définition purement architecturale d'une architecture orientée services pourrait être la suivante : « Architecture d'applications au sein de laquelle toutes les fonctions sont définies en tant que services indépendants, comprenant des interfaces bien définies qui peuvent être sollicitées dans des séquences définies ».

Donc un système basé sur ce type d'architecture doit effectuer et gérer la sollicitation du service et non l'application appelante. Cette fonction permet la réalisation de deux caractéristiques essentielles :
Tout d'abord, l'indépendance véritable des services et deuxièmement, la possibilité de gérer ces derniers.

La gestion comprend plusieurs fonctions :

- La sécurité, l'autorisation des requêtes, le cryptage et le décryptage des données, selon les instructions données, et la validation des informations.
- Le déploiement permet au service d'être déplacé au sein du réseau pour maximiser les performances et supprimer la redondance afin que la disponibilité des applications soit maximale.
- La maintenance permet de gérer de nouvelles versions du service.

3.5.2 La nature d'un service

Un service au sein d'un environnement d'entreprise constitue en général une fonction simple, une transaction plus complexe ou un service système. D'un point de vue de l'application, ces fonctions sont des fonctions non système atomiques. Les transactions peuvent sembler n'être que de simples fonctions pour l'application appelante, mais elles peuvent être mises en œuvre en tant que fonctions composites masquées par leur contexte transactionnel propre. Elles impliquent parfois de multiples fonctions de niveau inférieur transparentes pour l'appelant. Les fonctions systèmes sont des fonctions généralisées qui peuvent être extraites d'une plateforme spécifique, telle que Microsoft Windows ou Linux.

Ceci n'est pas une simple distinction artificielle entre les différents services. D'un point de vue de l'application, tous les services sont atomiques, mais le fait qu'il s'agisse de services d'entreprise ou système n'a aucune importance. Cette distinction ne sert qu'à introduire le concept important de granularité. La décomposition des applications d'entreprise en services ne constitue pas uniquement un processus abstrait, elle implique également des opérations très pratiques. Les services peuvent être composés de fonctions de niveau inférieur (granularité fine) ou de niveau supérieur complexe (granularité grossière). Il existe de véritables compromis de performance, de flexibilité, de possibilités de maintenance et de réutilisation basés sur les

définitions de ces fonctions. Le niveau de granularité indique la richesse fonctionnelle d'un service. Par exemple, plus la granularité d'un service est grossière, plus la fonction offerte par le service est riche.

Les services sont en général composés de fonctions commerciales à granularité grossière, telles que le service « ouvrir un compte », car ces opérations peuvent supposer l'exécution de multiples opérations à granularité plus fine, telles que le service « vérifier l'identité du client » et le service « créer le compte client ». Ce processus de définition de services est en général effectué dans un cadre plus large, le cadre d'applications. C'est ce cadre qui doit être mis en place. Il s'agit de développer un cadre d'applications de composants dans lequel les services sont définis en tant qu'ensemble de composants réutilisables pour créer de nouvelles applications ou intégrer un parc logiciel existant.

3.5.3 L'intégration d'applications

Si nous revenons à présent au premier exemple d'intégration mentionné (figure 3.1), on peut l'améliorer par un schéma qui minimise le nombre d'interfaces requises, tel que celui présenté à la figure suivante

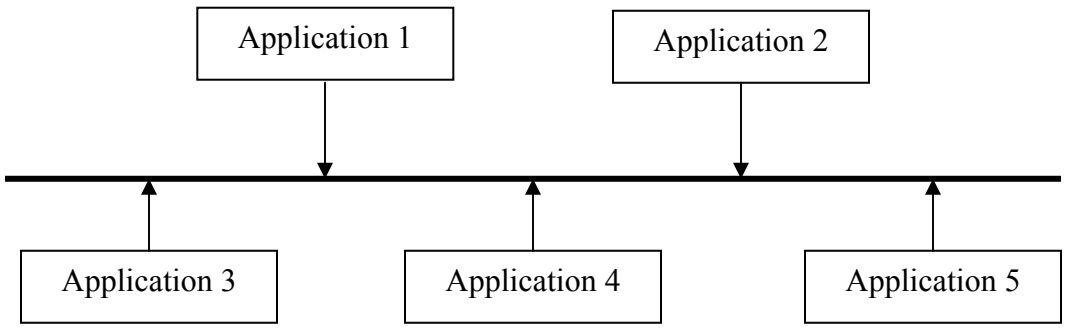

Figure 3.2 : L'intégration d'applications

La figure précédente peut paraître trop simpliste, mais elle illustre bien le point de départ de l'intégration des applications.

En suivant ce cadre d'intégration, on peut ajouter le concept architectural de bus de services, représenté dans la figure précédente par la ligne épaisse au centre, et un gestionnaire de services ou de flux pour connecter les services et fournir un chemin aux requêtes de services. Le gestionnaire de flux traite une séquence d'exécution définie, ou flux de services, qui sollicite les services requis dans la séquence adéquate pour produire le résultat final (voir figure suivante).

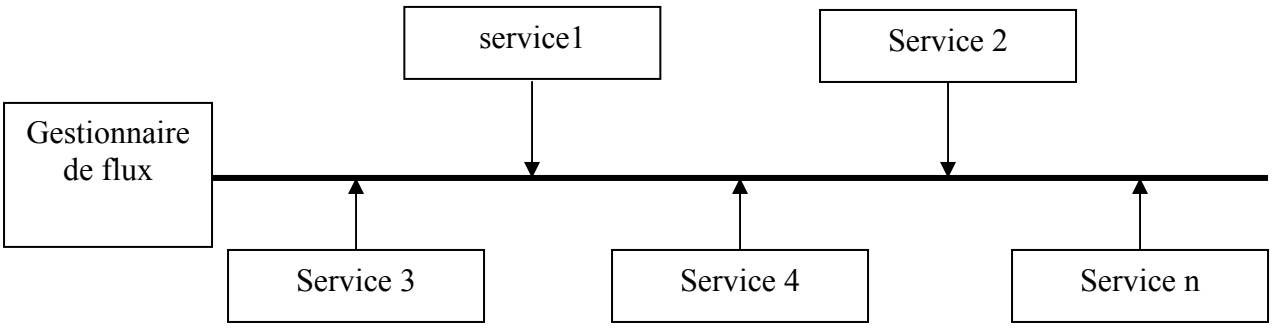

Figure 3.3 : L'intégration des services

3.5.4 Les exigences d'intégration au sein de l'architecture

Jusqu'à présent, le débat sur l'intégration s'est limité à l'intégration d'applications via des services de composants, mais l'intégration est un sujet beaucoup plus vaste. Lorsqu'on évalue nos exigences en ce qui concerne une architecture, on doit prendre en compte plusieurs types d'intégrations. On doit penser non seulement à l'intégration d'applications, mais également à celle de l'interface de l'utilisateur final, à la connectivité des applications, à l'intégration des processus, à l'intégration des informations et au modèle de développement prêt à l'intégration.

3.5.4.1 L'intégration au niveau de l'interface

L'intégration au niveau de l'interface de l'utilisateur final correspond à la manière dont l'ensemble complet d'applications et de services auquel un utilisateur donné accède est intégré afin de fournir une interface utilisable, efficace et cohérente. Ce sujet est en évolution constante et les nouveaux développements, à court terme, traiteront essentiellement des avancées dans l'utilisation des serveurs de portails. Alors que les portlets peuvent déjà solliciter des composants de services locaux à l'aide de services Web, de nouvelles technologies, telles que les services

Web pour portlets distants, activeront des fournisseurs de contenu et d'applications pour créer des services interactifs prêts à être utilisés avec des portails via Internet et offriront donc de nombreuses possibilités nouvelles d'intégration.

3.5.4.2 La connectivité d'applications

La connectivité d'applications est un type d'intégration relatif à tous les types de connectivités devant être pris en charge par l'architecture. À un certain niveau, cette intégration traite des communications synchrones et asynchrones, du routage, des transformations, de la distribution rapide des données, et des convertisseurs de passerelles et de protocoles. À un autre niveau, elle s'occupe également de la virtualisation des données en entrée et des résultats, ou des sources et des récepteurs, tels que les programmes de gestion de canaux et de protocoles. Le problème réside dans la manière fondamentale dont les données entrent, sortent et se déplacent dans le cadre qui met en œuvre l'architecture.

3.5.4.3 L'intégration des informations

L'intégration des informations est un processus qui fournit un accès cohérent à toutes les données pour toutes les applications, quel que soit la manière dont ces données doivent être envoyées, leur format, leur source ou leur emplacement. Lors de la mise en œuvre, cette exigence peut supposer la simple utilisation d'un logiciel adaptateur et d'un moteur de transformation. Toutefois, en général, le processus est plus complexe. Le concept clé est souvent la virtualisation des données qui peut inclure le développement d'un bus de données à partir duquel toutes les applications peuvent envoyer des requêtes de données via des services ou des interfaces standards. Les données peuvent ainsi être envoyées à l'application qu'elles proviennent d'une feuille de calcul, d'un fichier natif, d'une base de données SQL (Structure Query Language), d'un autre type de base de données ou d'un stockage de données en mémoire.

3.6 Les avantages du déploiement d'une architecture orientée services

Une architecture orientée services peut souvent évoluer à partir de systèmes existants et ne requiert donc pas de réécriture système complète.

Les organisations qui concentrent leurs efforts de développement sur la création de services, en utilisant des technologies existantes associées à une approche basée sur les composants vis-à-vis du développement logiciel, profiteront de nombreux avantages.

3.6.1 L'exploitation du parc informatique existant

Cet avantage est le premier et le plus important de toutes les exigences mentionnées précédemment. On peut constituer un service en ajoutant des composants existants à l'aide d'un cadre d'architecture orientée services adapté et mis à la disposition de la plate-forme ou l'application sur laquelle on travaille. L'utilisation de ce nouveau service requiert uniquement la connaissance de l'interface et du nom du service, la complexité du flux de données au sein des composants du service, est transparente pour les appelants. L'anonymat des composants permet aux organisations de tirer profit des systèmes actuels, de créer des services à partir d'un ensemble de composants installés sur différents ordinateurs exécutant différents systèmes d'exploitation et développés dans des langages de programmation différents. Les systèmes existants peuvent être encapsulés et leur accès est possible grâce aux interfaces des services Web. Plus important encore, ils peuvent être transformés et prendre ainsi de la valeur à mesure que leur fonctionnalité est transformée en services.

3.6.2 L'infrastructure comme matière première

Le développement et le déploiement de l'infrastructure deviennent de plus en plus cohérents sur les diverses applications qu'on veut rassembler. Les composants existants, les nouveaux composants peuvent être consolidés au sein d'un cadre d'architecture orientée services bien défini. Un tel ensemble de composants est déployé sous forme de services dans l'infrastructure existante. L'infrastructure sous-jacente devient alors une matière première. Ensuite, à mesure que le couplage des services et du matériel prenant ces derniers en charge devient plus lâche, on peut optimiser le matériel car le programme d'assemblage de services ne dépend plus de l'environnement matériel dans lequel le service fonctionne au moment de l'exécution.

3.6.3 Le temps de développement

Les bibliothèques d'organisation de services Web deviennent les atouts essentiels des organisations en faisant partie du cadre d'architecture orientée services. La création et le déploiement de services à l'aide de ces bibliothèques de services Web réduit considérablement le temps de développement, car les nouvelles initiatives réutilisent les services et les composants existants, réduisent le temps de conception, de test et de déploiement du processus.

3.6.4 L'atténuation des risques

La réutilisation de composants existants réduit le risque d'introduction de nouveaux échecs dans le processus d'amélioration ou de création de nouveaux services. Le risque de maintenance et de gestion de l'infrastructure prenant en charge les services diminue également.

3.6.5 Une architecture centrée autour des processus

Dans une architecture centrée autour des processus, l'application est développée pour le processus. Ce dernier est décomposé en une série d'étapes qui représentent chacune un service. En réalité, chaque fonction de service ou de composant constitue une sous-application. Ces sous-applications peuvent être assemblées pour créer un flux de processus capable de satisfaire les besoins de l'organisation.

3.7 Conclusion

Dans ce chapitre nous avons expliqué en détail le concept d'architecture orientée services car elle constitue la prochaine vague de développement d'applications et comprend des propriétés spécifiques, des composants et des interconnexions qui mettent l'accent sur l'interfonctionnement et la transparence d'emplacement.

Dans la deuxième partie nous introduisons la notion de services Web car ils constituent un ensemble de technologies dynamisantes pour cette architecture, qui devient l'architecture de choix pour le développement de nouvelles applications réactives et adaptatives, puis on va proposer une architecture générale qui permet de concevoir une application collaborative modulaire, flexible et simple qui se base sur une architecture orientée service et surtout sur la notion de services web.

Chapitre 4
La technologie des services web dans la pratique

4.1 Introduction

Avec l'essor du web qui a eu dans les dernières années, il a surgi le besoin de permettre qu'une application cliente invoque un service d'une application serveur en utilisant Internet. Ce besoin a été à l'origine de ce qui est connu comme services web. En tenant en compte que les services web permettent de connecter des applications différentes, l'utilité de cette technologie devient évidente et importante. Pour cette raison les activités de recherche et développement autour du sujet services Web ont un dynamisme très haut. Dans ce chapitre on va détailler la notion de service web avec ces différentes technologies (XML, WSDL, SOAP, etc.…), dans le but de définir par la suite une approche basée sur ce genre de technologies.

4.2 Définition et description de Services Web

Le groupe de W3C qui travaille sur les services Web a utilisé la définition de service Web suivante :

« Un service web est un système logiciel destiné à supporter l'interaction ordinateur-ordinateur sur le réseau. Il a une interface décrite en un format traitable par l'ordinateur (ex : WSDL). Autres systèmes réagissent réciproquement avec le service web d'une façon prescrite par sa description en utilisant des messages SOAP, typiquement transmis avec le protocole http et une sérialisation XML, en conjonction avec d'autres standards relatifs au web » [25].

4.3 Les principales technologies de développement des Services Web

Une caractéristique qui a permis un grand succès de la technologie des services web est qu'elle est construite sur des technologies standards de l'industrie. Dans cette section il y a une description de ces technologies.

4.3.1 XML – eXtensible Markup Language

XML « eXtensible Markup Language » [26] est un format texte simple et très flexible tiré du SGML (voir chapitre 1).

Nous allons prendre un exemple simple pour visualiser graphiquement quelle serait la hiérarchie du document XML correspondant et quelle est la structure du document. L'exemple choisi est celui d'un livre.

```
<Livre>
  <titre>Le super livre</titre>
       <chapitre>
          <numéro>1</numéro>
          <titre>titre du chapitre 1</titre>
          <contenu>blabla blabla</contenu>
       </chapitre>
       <chapitre>
       </chapitre>
</Livre>
```

Figure 4.1 : La représentation d'un livre en xml

XML possède une galaxie de technologies. Parmi celles-ci nous pouvons citer XML Query (langage de requêtes), XSL – eXtensible Stylesheet Language (définition de présentations de documents XML), XSLT – XSL Transformations (langage permettant la transformation de documents XML), XPath – XML Path (langage de requêtes permettant d'accéder à chaque élément d'un document XML).

4.3.2 SOAP : Simple Object Access Protocol

SOAP [27] est un protocole pour l'échange d'information dans un environnement réparti, basé sur le standard XML. Ce protocole consiste en trois parties : une enveloppe qui définit un canevas pour décrire le contenu du message et comment le traiter , un jeu de règles de codage pour exprimer les types de données et une convention pour représenter des appels de procédure éloignés.

SOAP n'est lié à aucun système d'exploitation ni langage de programmation. Il est indépendant du langage d'implémentation des applications client et serveur. En plus, il peut potentiellement être utilisé avec une variété de protocoles (ex : HTTP, HTTP Extension Framework).

4.3.2.1 Modèles d'échange de messages en SOAP

Les messages SOAP sont des transmissions fondamentalement à sens unique d'un expéditeur à un récepteur. Lorsqu'une transmissions d'un message commence (ex : invocation d'un service web), un message SOAP (ou document XML) est généré. Ce message est envoyé à partir d'une entité appelé le SOAP sender, localisé dans un SOAP nœud. Le message est transmis à zéro ou plusieurs nœuds intermédiaires (SOAP intermediates) et le processus fini lorsque le message arrive au SOAP receiver. Le chemin suivi par un message SOAP est nommé message path. La figure suivante montre les éléments qui participent au processus.

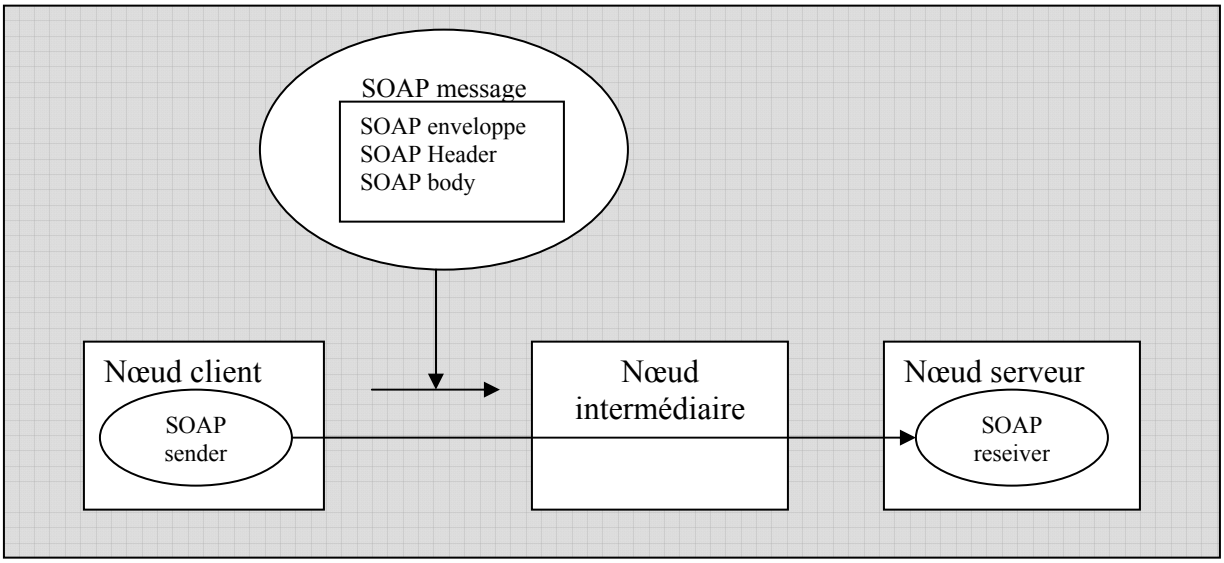

Figure 4.2 : Structure d'un message SOAP

Quand un message SOAP arrive dans une entité SOAP, il suit le processus décrit ci-dessous :

1. Identifier toutes les parties du message SOAP destiné à cette entité.

2. Vérifiez que ces parties sont soutenues par l'entité et traitez-les en conséquence. Si ce n'est pas le cas rejeter le message.

3. Si l'entité n'est pas la destination suprême du message, il faut enlever alors toutes les parties identifiées avant l'expédition du message.

4.3.2.2 Enveloppe SOAP

Un message SOAP est composé d'un élément obligatoire appelé le SOAP enveloppe. L'enveloppe SOAP définit l'espace de nom (namespace : URI permettant de connaître la provenance de chaque balise) précisant la version supportée de SOAP. Cet espace de nom est standard et permet de différencier les éléments du schéma.

L'enveloppe SOAP est constituée d'un en-tête facultatif (SOAP header) et d'un corps obligatoire (SOAP body). Une description générale de l'enveloppe SOAP et ses composants est donnée par un extrait de code que nous verrons au paragraphe [4.3.2.4 Corps SOAP, Figure 4.3].

4.3.2.3 En tête SOAP

L'en-tête peut avoir plusieurs fils (SOAP blocks). Ces fils sont utilisés pour ajouter des fonctionnalités au message comme l'authentification et la gestion des transactions. L'en-tête peut utiliser les attributs mustUnderstand et/ou SOAP actor pour indiquer comment traiter l'entrée et par qui.

L'attribut mustUnderstand peut être utilisé pour indiquer si une entrée d'en-tête est obligatoire ou facultative pour être traitée par le destinataire. Le destinataire d'une entrée d'en-tête est défini par l'attribut d'acteur de SOAP (décrit dans la partie suivant). La valeur de l'attribut de mustUnderstand est "1" ou "0". L'absence de cet attribut est sémantiquement équivalente à sa présence avec la valeur "0".

Un message SOAP voyage du SOAP sender au SOAP receiver, en passant par un groupe de SOAP intermédiaires. Un SOAP intermédiaire est une entité qui est capable de recevoir et transmettre les messages SOAP. Les nœuds intermédiaires aussi bien que le SOAP receiver sont identifiés par une URL. L'attribut acteur de SOAP peut être utilisé pour indiquer le destinataire d'un élément d'en-tête. La valeur de l'attribut d'acteur de SOAP est une URL. Par exemple, l'URL "http: // schemas.xmlsoap.org/soap/actor/next" indique que l'élément d'en-tête est destiné à la première entité SOAP qui traite le message.

4.3.2.4 Corps SOAP

Le corps SOAP contient l'information destinée au receveur. Il doit fournir le nom de la méthode invoquée par une requête, ou le nom de la méthode pour générer la réponse. Il doit aussi, fournir l'espace de nom correspondant au nom du service. Le contenu du corps SOAP est utilisé dans le processus comme le marshaling d'appels RPC et le rapport des erreurs.

Le corps SOAP peut contenir un SOAP fault. Ce bloc est utilisé pour transmettre l'information des erreurs durant le traitement du message.

Malgré que l'en-tête et le corps soient définis comme des éléments indépendants, ils ont une relation : une entrée de corps est sémantiquement équivalente à une entrée d'en-tête destinée pour l'acteur de défaut et avec une valeur d'attribut mustUnderstand de 1.

```
<env:Envelope    xmlns:env="http://www.w3.org/2001/06/soap-envelope/
">
<env:Header>
        <authentication xmlns=http://myDomain/app" >
        <user>admin</user>
        <password>********</password>
        </authentication>
</env:Header>
<env:Body>
<stock: getStockQuoteResponse
xmlns: stock="http://www.acme.com/service ">
<stock_quote>
…
</stock_quote>

</stock: getStockQuoteResponse

        </env:Body>

</env:Envelope>
```

Figure 4.3 : Définition du type d'enveloppe SOAP

4.3.3 Apatche-Axis : une mise en œuvre de SOAP

Axis est essentiellement un moteur de SOAP, mais il inclut aussi :

- Un serveur autonome simple.
- Un serveur servlet comme Tomcat.
- Appui vaste pour WSDL.
- L'outillage d'émetteur (ex : WSDL2Java) qui produit des classes Java à partir du fichier WSDL.
- L'outillage d'émetteur (ex : Java2WSDL) qui produit le fichier WSDL à partir de la classe de l'interface Java.
- Un outil pour surveiller des paquets de TCP/IP [28].

4.3.4 WSDL : Web Service Description Language

Le WSDL [29] est un langage qui permet de décrire les services web, et en particulier, leurs interfaces. Ces descriptions sont des documents XML. WSDL décrit un service web en deux étapes fondamentales : une abstraite et une concrète. Dans chaque étape, la description utilise un nombre de constructions pour favoriser la réutilisation de la description et pour séparer les préoccupations de conception indépendantes (voir figure suivante).

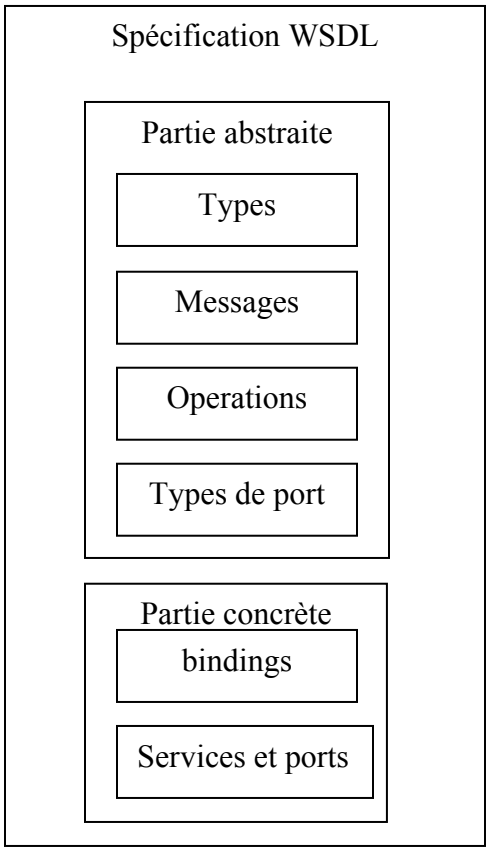

Figure 4.4 : La spécification d'un Service Web avec WSDL

Au niveau abstrait, WSDL décrit un service Web en termes des messages qu'il envoie et reçoit; les messages sont décrits de façon indépendante d'un format spécifique en utilisant un système de types, typiquement un schéma XML. La partie abstraite est composée de définitions de "port type" qui sont analogues aux interfaces des "middleware" traditionnel. Chaque "port type" est une collection logique d'opérations. Une "opération" associe un modèle d'échange de message à un ou plusieurs messages. Un message est une unité de communication avec un service web. Il représente les données échangées dans une unique transmission logique.

Un modèle d'échange de messages identifie l'ordre et la cardinalité des messages envoyés et/ou reçus. Une interface groupe un ensemble d'opérations.

Au niveau concret, un "binding" indique des détails de format de transport pour une ou plusieurs interfaces. Un "endpoint" (point final) associe une adresse de réseau à un "binding"

(attache). Finalement, un service groupe un ensemble d'endpoints qui implémente une interface commune.

4.3.5 UDDI: Universal Description Discovery and Integration

L'objectif primaire d'UDDI [27] est la spécification d'un canevas pour décrire et découvrir des services Web. Le noyau d'UDDI travaille avec la notion de "business registry", qui est un service sophistiqué de noms et répertoires. Plus précisément, UDDI définit des structures de données et APIs pour publier les descriptions des services dans le registre et pour interroger ce registre afin de chercher des descriptions publiées. Parce que les APIs d'UDDI sont aussi spécifiés en WSDL avec une attache SOAP, le registre peut être invoqué comme un service Web (en conséquence, ses caractéristiques peuvent être décrites dans le même registre, comme un autre service).

Les spécifications du « registry » UDDI ont deux buts principaux en ce qui concerne la découverte d'un service: le premier, soutenir les développeurs dans la découverte d'informations sur les services. Le deuxième, permettre la liaison dynamique et en conséquence habiliter les clients pour interroger le registre et obtenir des références aux services d'intérêt.

4.5 Conclusion

En partant du constat du besoin d'une interopérabilité toujours plus grande et d'une évolution des techniques et outils à disposition des développeurs, ce chapitre met en évidence une instance du SOA : les services web. En effet, ces services web seront le centre de nos préoccupations dans la suite de ce mémoire. C'est pourquoi, ce chapitre présente également quelques unes des technologies de ce domaine avec leurs modes de fonctionnement et leurs domaines d'application.

Dans le chapitre suivant nous allons essayer de proposer une architecture générale pour concevoir un environnement collaboratif, cette architecture se base sur les technologies des services web qu'on vient de voir et sur d'autres technologies que introduirons ultérieurement.

Chapitre 5
Architecture proposée

5.1 Introduction

Dans ce chapitre nous allons donner une architecture globale d'une plate-forme collaborative basée sur les différents outils de collaborations, cette architecture englobe plusieurs technologies.

5.2 Architecture proposée

L'architecture qu'on propose pour la conception d'une plate-forme collaborative est une infrastructure composée d'une architecture orientée service SOA qui met en œuvre un certain nombre de services pour la plate-forme utiles pour la collaboration couplée avec une interface client riche qui offre ainsi un degré d'interactivité et une ergonomie comparables aux interfaces utilisateurs standards. L'objectif principal de l'architecture orientée services est donc de décomposer une fonctionnalité en un ensemble de fonctions basiques, appelées services, ce dernier est le composant clef de l'Architecture. Il consiste en une fonction ou fonctionnalité bien définie. C'est aussi un composant autonome qui ne dépend d'aucun contexte ou service externe, fournis par des composants et de décrire finement le schéma d'interaction entre ces services.

L'idée de cette approche est de construire une architecture logicielle globale décomposée en services correspondant aux besoins des utilisateurs.

5.2.1 Les couches de l'architecture

On va se baser sur une architecture trois tiers qui se décompose en trois couches :

5.2.1.1 Premier niveau ou couche présentation

Ce niveau correspond à l'interface utilisateur hébergée au sien du navigateur, cette couche transmet les requêtes effectuées par l'utilisateur à la couche métier, et affiche les résultats retournés par celle-ci.

5.2.1.2 Deuxième niveau ou couche métier

À ce niveau sont définis les traitements que doit réaliser l'application lorsqu'elle reçoit une requête émanant de la couche présentation. Ce niveau correspond à la logique métier de l'application.

5.2.1.3 Troisième niveau ou couche d'accès aux données

Ce niveau gère l'accès aux données. Quant la couche métier a besoin d'accéder à des informations, elle adresse une requête à ce niveau .lorsque les données sont disponibles, elles sont transmises à la couche métier qui va les traiter.

En gardant l'idée de l'architecture trois tiers, on peut définir une nouvelle architecture qui se compose de trois couches, couche présentation, couche métier ou traitement, couche accès aux données, mais qui donne à chacune de ces couches un types et des rôles bien précis (voir figure suivante).

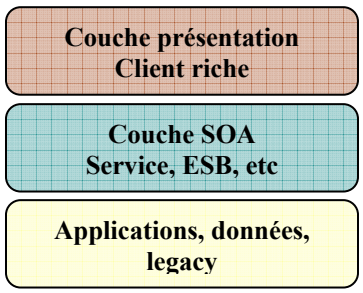

Figure 5.1 : Les différentes couches de l'architecture

La plus basse couche reste une couche d'accès aux données, mais le changement s'effectue au niveau des deux autres couches de la manière suivante :

- La couche du milieu (couche métier) va jouer le rôle d'une architecture SOA proprement dite, avec les services, le processus métier, etc. (voir chapitre architecture orientée service et chapitre les technologies des services web en pratique).
- La couche présentation va exposer ces services via une interface utilisateur en utilisant la notion du client riche.

À présent on va définir quelques concepts principaux sur les quels repose notre proposition.

Parmi ces concepts on va définir la notion du client riche, la plate-forme utilisé pour construire ce type d'application, on définira aussi quelques concepts concernant les services, leurs interactions, le concept d'orchestration, etc. dans le but de représenter par la suite, le schéma générale de notre approche de façon compréhensible.

5.2.2 Concepts utilisés au niveau de la couche présentation (RIA)

5.2.2.1 Le client riche

Un client riche est un compromis entre un client léger qui désigne une application accessible via une interface web en HTML consultable par un navigateur web, où la totalité du traitement se fait au niveau du serveur, et un client lourd qui désigne une application cliente exécutée sur le système d'exploitation, et qui possède généralement des capacités de traitement évoluées et peut posséder une interface graphique sophistiquée.

L'objectif du client riche est donc de proposer une interface graphique, décrite avec une grammaire de description basée sur la syntaxe XML, permettant d'obtenir des fonctionnalités similaires a celles d'un client lourd (glisser, déposer, multi fenêtrage, menu déroulants).
Le client riche permet ainsi de gérer l'essentiel des traitements du coté du serveur. Les données sont ensuite transmises dans un format d'échange standard utilisant la syntaxe XML (SOAP, XML-RPC), puis interprétées par le client riche [30].

Pour développer ce genre d'applications, il existe plusieurs technologies ou standards parmi les quels on peut citer :

XAML (eXtensible Application Marckup language) : un standard XML proposé par Microsoft [31].

XUL (XML-based User interface Language) : un standard XML propose par la fondation Mozzila [32].

Flex : un standard XML proposé par la société Macromedia, il se base sur Flash [33].

Wazaabi : est un RCP Eclipse permettant de créer des applications riches en J2EE et en se basant sur XUL [34].

Openlaszlo : c'est la plate-forme qu'on va utiliser pour mettre en œuvre notre approche, c'est pour cette raison qu'on va donner une vue globale sur son architecture, son système d'événements et surtout son client.

5.2.2.2 Openlaszlo

Openlazslo [35] est une plate-forme open source développée par Laszlo system, elle est comparable à Flex [33] dans le sens où elle permet de développer des applications riches qui seront ensuite visibles via le pluging Flash player, mais peut également générer du HTML en utilisant Ajax.

L'architecture de cette plate-forme mélange la puissance et la facilité d'utilisation de l'architecture client/serveur avec les avantages en administration et en réduction des couts des applications web.

A Mode de déploiement

Les applications OpenLaszlo peuvent être mises à disposition sur le web de deux façons :

A.1 Proxy

Le Serveur OpenLaszloa les fonctions suivantes, il :

- compile les programmes sources et envoie les binaires à exécuter sur le poste client.
- Sert de proxy entre le poste client et d'autres serveurs sur Internet, avec une éventuelle manipulation de données.

A.2 SOLO

Le compilateur OpenLaszlo est utilisé pour "précompiler" les programmes et les binaires résultants sont placés sur le serveur. Quand le client exécute l'application, celle-ci contacte directement d'autres serveurs sans utiliser le serveur OpenLaszlo en tant que médiateur. On appelle ce mode de déploiement "sans serveur", ou "standalone" (voir figure suivante).

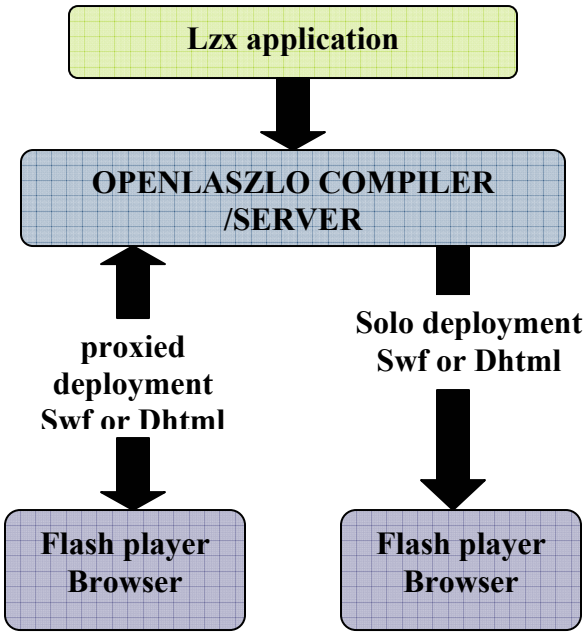

Figure 5.2 : Applications déployées en SOLO ou par le serveur Openlaszlo [36]

B Client et Serveur : Synopsis

Le Serveur OpenLaszlo est une application Java qui s'exécute dans un conteneur de servlets J2EE, elle peut communiquer avec d'autres serveurs et sources de données en utilisant un grand nombre de protocoles. Les applications OpenLaszlo écrites en LZX sont compilées par le Serveur OpenLaszlo et rendues disponibles en tant que bytecode pour le plug-in qui fonctionne dans le navigateur du poste client. L'environnement d'exécution actuellement supporté est le plug-in Flash 5 ou plus, il fonctionne de façon similaire et fiable sur un grand nombre de systèmes d'exploitation et de matériels, dont Windows, Pocket PC, Mac OS, Linux et Solaris

Dans l'environnement OpenLaszlo, le client désigne l'application LZX qui s'exécute dans le navigateur de l'utilisateur, et le serveur désigne le Serveur OpenLaszlo (qui peut communiquer avec d'autres serveurs), ils peuvent se communiquer en utilisant le protocole http, cela se fait par l'envoi du bytecode par le serveur et du XML par le client LZX [36].

Toutes les fonctionnalités supplémentaires de la plate-forme OpenLaszlo, dont le streaming de médias et la mise à jour, s'appuient sur HTTP et HTTPS. Les applications qui s'appuient sur OpenLaszlo peuvent ainsi rester compatibles avec les pare-feux d'entreprise.

B.1 L'architecture d'Openlaszlo coté serveur

Le Serveur OpenLaszlo s'exécute dans un serveur d'applications standard J2EE ou un conteneur de servlets Java avec JRE 1.3 ou supérieur.

Il est composé de cinq sous-systèmes principaux :

- **Le Compilateur d'interface :** constitué d'un compilateur de balises LZX et d'un compilateur de Script. Le compilateur d'interface peut aussi faire appel au compilateur de médias ou au gestionnaire de données pour compiler des médias ou des données qui sont embarquées dans l'application. Les compilateurs de balises LZX et de script transforment la description d'applications par balises et code JavaScript en bytecode exécutable (swf pour Flash) sur le poste client. Ce code est placé dans le cache, d'où il est envoyé au client. Suivant l'application qui est invoquée, il est envoyé soit en tant que fichier .swf, soit en tant que page HTML avec un objet .swf embarqué.
- **Le Transcodeur de média :** convertit l'ensemble de types de médias en un format unique qui puisse être interprété par le système d'affichage du client. Cela permet aux applications OpenLaszlo d'afficher les types de médias supportés d'une façon unique dans une fenêtre commune, sans faire appel à des applications externes ou interpréteurs supplémentaires. Le Transcodeur de média convertit automatiquement les types de données suivants : JPEG, GIF, PNG, MP3, TrueType, et SWF (dessins et animations seulement).
- **Le Gestionnaire de données :** est constitué d'un compilateur de données qui convertit toutes les données en un format de données binaires, compressé et lisible par les applications OpenLaszlo.
- **Le Gestionnaire de Connexion Permanente :** gère les authentifications et un système de messages temps réel pour les applications qui en ont besoin. Il fournit

un système de messagerie et d'avertissement temps réel, par envoi de données avec HTTP.

- **Le Cache :** contient les versions d'applications récemment compilées. La première fois qu'une application OpenLaszlo est appelée, elle est compilée et le fichier SWF résultant est envoyé au client. Une copie est également mise en cache sur le serveur, pour que des demandes suivantes n'aient pas besoin d'attendre la compilation.

B.2 L'architecture d'openlaszlo coté client

L'architecture du client Laszlo est constituée de la Bibliothèque OpenLaszlo Temps-Réel (OpenLaszlo Runtime Library : ORL), une bibliothèque incluse dans toute application OpenLaszlo qui fournit des services temps-réel (comme par exemple une horloge ou une fonction d'attente) Aucune de ces classes ne repose sur des services Flash ou n'utilise le modèle objet de Flash. Le lecteur Flash est seulement utilisé comme un système de visualisation.

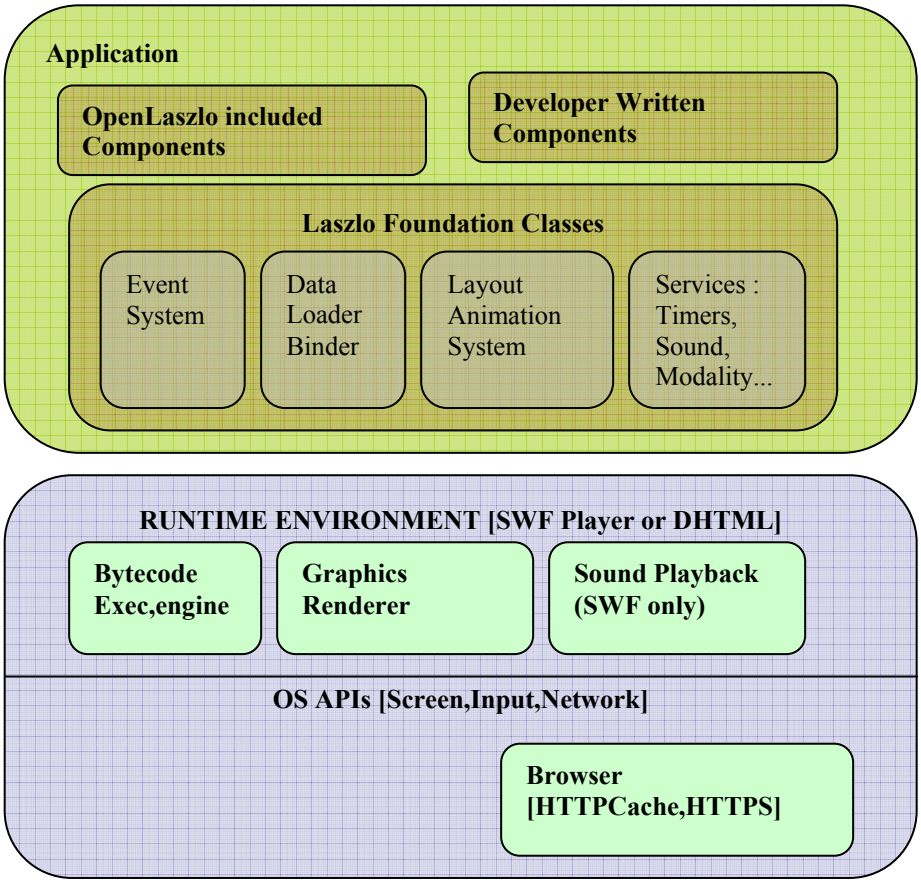

Figure 5.3 : L'architecture du client Laszlo [36]

Il y a principalement quatre composants dans la Bibliothèque OpenLaszlo Temps-Réel (ORL) : le Système d'événements, le Chargeur/Connecteur de données, le système de mise en page et d'animation et un ensemble de Services Système.

- **Le Système d'événements :** reconnaît et prend en charge les événements tels que les clics de boutons de souris ou les envois de données par le serveur. Ce composant permet une programmation standard d'événements sur le client. Par rapport aux applications Internet classiques, les applications OpenLaszlo réduisent la charge des serveurs en permettant aux applications clientes de trier, traiter et valider avec un affichage dynamique suivant l'état de l'application.
- **Le Chargeur de données :** il se place en contrôleur de données, en acceptant les flux de données réseau sur Serveur OpenLaszlo et en les connectant aux éléments graphiques d'affichage tels que les champs de texte, les formulaires et les éléments de menus.
- **Le Système de mise en page et d'animation :** fournit aux applications OpenLaszlo un système de mise en page basé sur des contraintes entre les éléments graphiques ainsi qu'un système d'animations des changements d'états dans l'interface. Ce composant permet de construire une interface dynamique avec très peu de programmation. Cela permet de positionner un nombre variables d'éléments graphiques en utilisant un positionnement relatif ou absolu par pixels.

C Modèle de sécurité

La plateforme OpenLaszlo supporte le protocole sécurisé SSL. Les transmissions de données à travers Internet peuvent être cryptées en utilisant SSL et HTTPS. Les applications OpenLaszlo s'exécutent sur le poste client en utilisant le lecteur Flash, mais ne peuvent pas écrire sur le système local de fichiers ou accéder à l'environnement du poste client.

Les services web et bases de données utilisées par une application OpenLaszlo sont aussi sécurisées en utilisant un modèle d'authentification par utilisateur. Ce mécanisme permet d'éviter l'utilisation du Serveur OpenLaszlo comme un proxy ou passerelle vers des services ou des données non sécurisées

5.2.3 Concepts utilisés au niveau de la couche SOA

5.2.3.1 L'activation de services

L'activation de services consiste à créer une variante d'un service afin de le configurer conformément aux règles de mise à disposition d'un consommateur : options de fonctionnement, choix du niveau de la qualité d'exploitation,…etc. Les règles de mise à disposition forment le contrat d'utilisation du service. Lorsque le contrat est validé par le fournisseur on considère que le service est activé pour une variante et pour un consommateur.

La plate-forme d'activation de services met en œuvre le modèle d'architecture de gestion par les contextes (pattern Contexte-Aware) pour la gestion des variantes et des contrats d'utilisation. Cette plate-forme dispose d'une console métier qui permet aux équipes techniques et opérationnelles de paramétrer les données des variantes et des contrats. Certains éléments d'une variante peuvent être directement modifiés par le consommateur si celui-ci dispose des droits délégués par le fournisseur. Il faut alors que la plate-forme d'activation de services autorise le travail collaboratif.

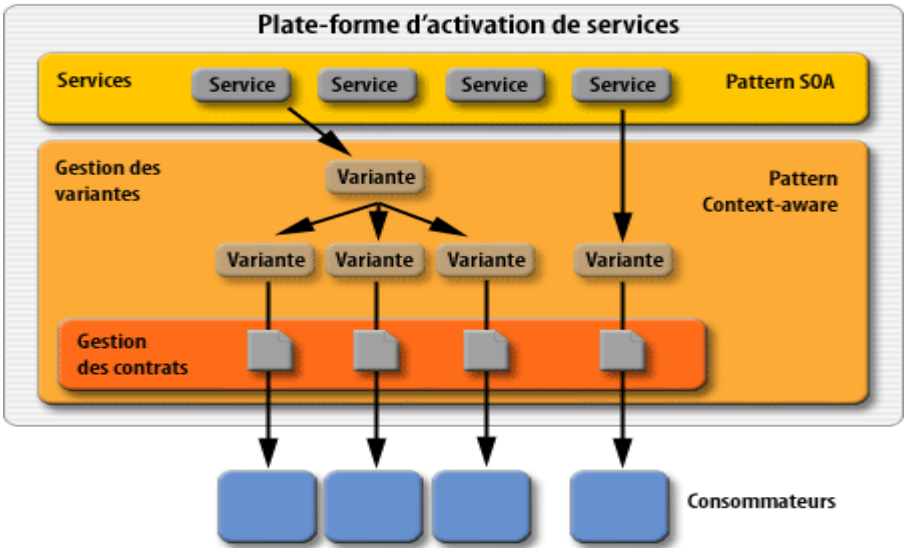

Figure 5.4 : L'activation des services [37]

5.2.3.2 Couplage faible

Un service est en couplage faible quand il n'est pas autorisé à appeler directement un autre service. Il délègue cette responsabilité à un traitement spécialisé que l'on nomme fonction d'orchestration. Grâce au couplage faible, on peut réutiliser un service sans devoir reprendre des services qui lui sont liés.

À partir de cette première définition, on ajoute d'autres principes techniques qui contribuent à découpler les services entre eux :

- Un service est activable indépendamment de sa technologie. Pour ce faire, l'activation se réalise par l'envoi (et la réception) d'un message XML (il ne s'agit donc pas d'une activation d'objet distribué comme en Corba ou DCOM).

- Un service peut être activé suivant un mode asynchrone. Dans ce cas, le service s'abonne à un événement auprès de la fonction d'orchestration.

5.2.3.3 La décomposition SOA

On distingue trois niveaux de décomposition SOA :

- Le premier est la découverte des opérations exposées par les services métiers à partir de la modélisation des processus. Il s'agit d'un regroupement d'activités formant le périmètre fonctionnel que l'on souhaite exposer aux consommateurs.

- Le second niveau consiste à décomposer les opérations et les phases découvertes lors de la modélisation du processus sous la forme de services rattachés aux catégories. Chaque opération et phase devient ainsi un orchestrateur d'appel vers les services exposés par les catégories (service de type externe).

- Le troisième niveau dépend de l'usage ou non d'un langage orienté objet. Il s'agit de décomposer chaque service exposé par une catégorie sous la forme de méthodes attachées aux classes qui constituent la catégorie d'appartenance. Cette décomposition se fait uniquement sur les classes de la catégorie d'appartenance, jamais sur les classes d'autres catégories (principe d'isolation des catégories entre elles), voir figure suivante.

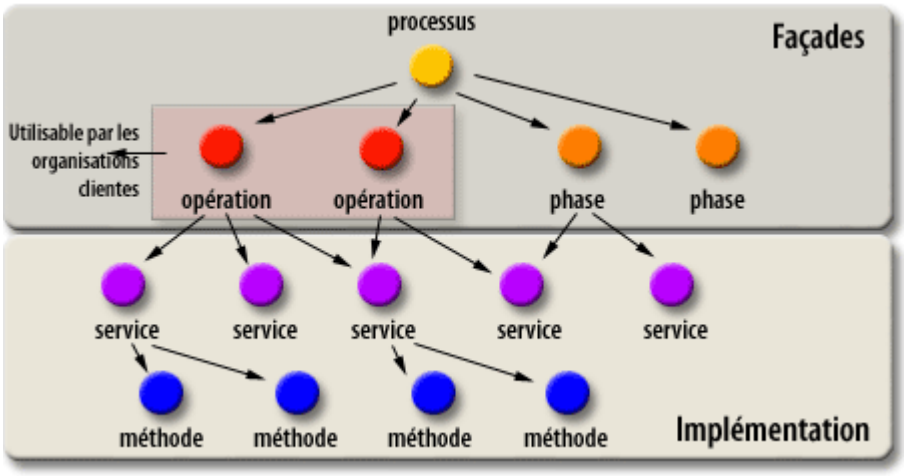

Figure 5.5 : La décomposition de l'architecture [38]

5.2.3.4 La réutilisation des services

Lorsque que l'on souhaite réutiliser un service il est souvent nécessaire de l'adapter au préalable afin qu'il se conforme au nouveau contexte d'utilisation. Cette adaptation ne doit pas nécessiter une duplication du code d'origine, ce qui conduirait à une impasse en génie logiciel (problème d'intégrité et de maintenance de plusieurs codes dupliqués). Pour étendre le code d'origine sans le modifier (principe de fermeture/ouverture du code) il existe trois approches possibles :

- La première consiste à tirer profit des propriétés d'héritage de l'objet et de la programmation par interface. À partir de la classe qui contient le code d'origine du service, on crée une classe fille par variante que l'on spécialise selon les besoins (surcharge ou remplacement des méthodes). Malheureusement ce mécanisme doit être exceptionnel car il conduit à créer des connexions d'héritage qui ne correspondent pas à un objectif de taxonomie au sens de la classification.
- La seconde approche consiste à utiliser le modèle d'architecture de gestion par les contextes (pattern de Context-Aware) qui permet de créer des variantes à l'aide d'un paramétrage du code

- La troisième solution est encore prospective mais s'annonce très intéressante. Il s'agit de l'usage des Aspects (AOP : Aspect Oriented Programming). L'AOP complète les langages

objets (pré-compilateur Java, C#) avec des mécanismes qui permettent d'étendre le code sans le modifier. Pour ce faire on définit dans des points de coupure (Pointcuts) l'endroit où l'on souhaite ajouter du code en indiquant l'événement qui le déclenchera.

5.3 Difficultés rencontrées dans la réalisation d'une plate-forme collaborative complète

Indépendamment de l'approche de développement choisie, la conception de nouvelles applications collaboratives qui soient extensibles se présente comme un grand défi pour les développeurs. Si d'un coté l'utilisation de boîtes à outils s'avère simple et rapide, le degré d'extensibilité obtenu est très restreint. Si les développeurs choisissent de s'appuyer sur des Framework ou des plates-formes, la possibilité d'intégrer de nouveaux composants est également contraignante puisque ces derniers doivent suivre un schéma d'interaction imposé par le système en question. Ou alors les développeurs doivent faire face à la complexité des logiciels afin de pouvoir intégrer des composants disponibles en open source.

Dans la pratique il n'existe que des collecticiels se focalisant sur des aspects spécifiques de la collaboration, et qui ont souvent été développés selon l'approche « From scratch », soit des plates-formes collaboratives, qui ont souvent été développées selon des frameworks très sophistiqués. Ces applications peuvent donner à l'utilisateur un certain niveau d'extensibilité, mais malgré toutes les possibilités d'extensibilité offertes par ces plates-formes, elles sont loin d'être considérées comme une panacée. Outre le coût associé à ces systèmes, l'utilisateur se trouve toujours limité à une plate-forme donnée. Autrement dit, il est confronté à un choix limité d'outils de collaboration.

Ainsi, des utilisateurs finissent par composer leurs propres "environnements collaboratifs" en choisissant, selon leurs besoins, différentes applications offrant des fonctionnalités spécifiques, en les exécutant en même temps mais indépendamment les unes des autres.

Réaliser une plate-forme collaborative qui répond aux exigences des collaborateurs est une tâche très difficile même avec l'utilisation d'une architecture orientée service qui offre beaucoup

de solutions d'interopérabilité et d'extensibilité, les développeurs doivent développer les différents services de collaboration (exemple : le tableau blanc, la vidéo conférence, le bureau partagé, le chat, le forum, le wiki, …etc) en prenant en compte leurs domaines d'utilisation (exemple : l'enseignement à distance, la gestion des connaissances,..etc).

Pour remédier à cela nous proposons une approche permettant à plusieurs plates-formes collaboratives de communiquer ensemble à l'aide de ce qu'on appelle l'intégration de modules en se basant sur l'architecture orientée service , et plus précisément les services web. Avant de détailler cette approche nous allons donner une petite introduction sur la technique d'intégration des outils de collaboration.

5.4 Vers l'intégration d'applications existantes

La problématique associée à l'intégration d'applications existantes est l'un des principaux axes de recherche qui concerne la possibilité de faire interopérer différents systèmes. L'interopérabilité ou l'intégration entre les applications facilite la coopération au sein d'une organisation ou entre différentes organisations sans les contraindre à utiliser exactement les mêmes systèmes ni à posséder les mêmes équipements.

La notion d'intergiciel a fait son apparition afin de faciliter le développement d'applications réparties et de gérer les interactions entre ces applications via des plates-formes hétérogènes (par exemple diversité du matériel, des systèmes d'exploitation ou des langages de programmation). Un intergiciel représente ainsi une solution architecturale au problème de l'intégration de différentes applications tout en imposant à ces applications une interface de service commune.

Parallèlement à l'évolution des intergiciels et d'internet, les technologies Web ont émergé et remporté un succès important, en permettant des interactions simples avec des ordinateurs à distance. Plus récemment, les services Web, qui reprennent la plupart des principes du Web en les appliquant à des interactions ordinateur-ordinateur, se présentent comme une technologie potentielle pour l'intégration d'applications à travers l'Internet.

Le schéma suivant illustre de façon détaillée l'interaction entre une application cliente et un service web.

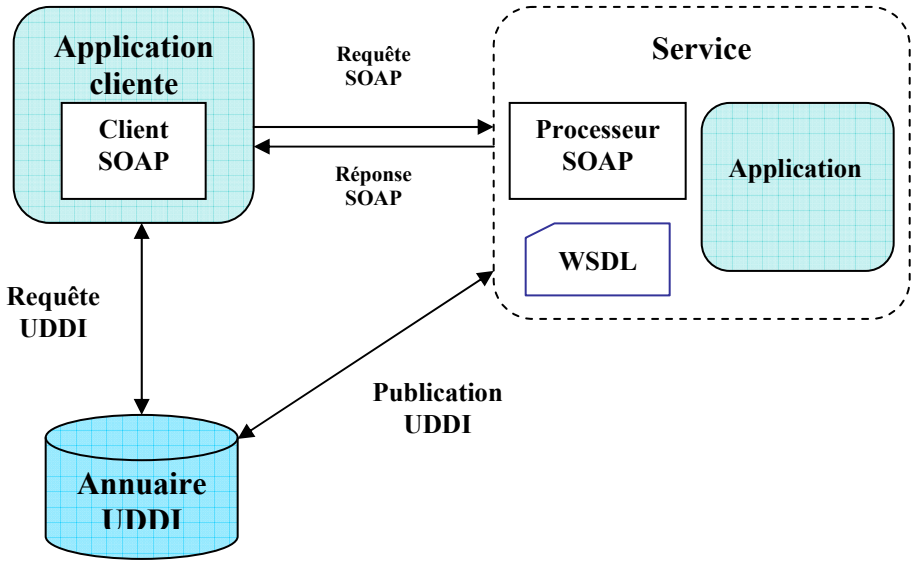

Figure 5.6 : Exemple d'interaction simple entre une application cliente et un service Web

Ce schéma nous montre que SOAP représente une sorte de façade qui masque l'hétérogénéité technique de l'application ou des données sous-jacentes. Nous avons donc une surcouche logicielle intercalée entre le programme développé (dans un langage propriétaire) et le monde extérieur. L'interopérabilité est ainsi un aspect intrinsèque des services Web. Grâce à leur formalisation standard définissant des normes ouvertes, ils fournissent un moyen technique pour exposer les interfaces de programmation de leurs applications dans un format compréhensible par n'importe quelle autre application. Des applications implémentées dans divers langages de programmation et sur diverses plates-formes peuvent ainsi employer des services Web pour interagir et échanger des données à travers des réseaux informatiques.

5.5 Le cadre général d'intégration

Pour réaliser l'intégration d'applications collaboratives qui sont à l'origine indépendantes ou plus précisément des outils de collaboration synchrones ou asynchrones indépendants, on peut définir un environnement général d'intégration qui s'appuie sur des applications clients serveur

développées pour le web, comme cela est illustré dans la figure suivante. Nos besoins de base pour réaliser cette intégration sont :

- L'indépendance vis-à-vis de la plate-forme qui découle de l'hétérogénéité des systèmes repartis actuels.
- L'extensibilité et la flexibilité de plate-forme.

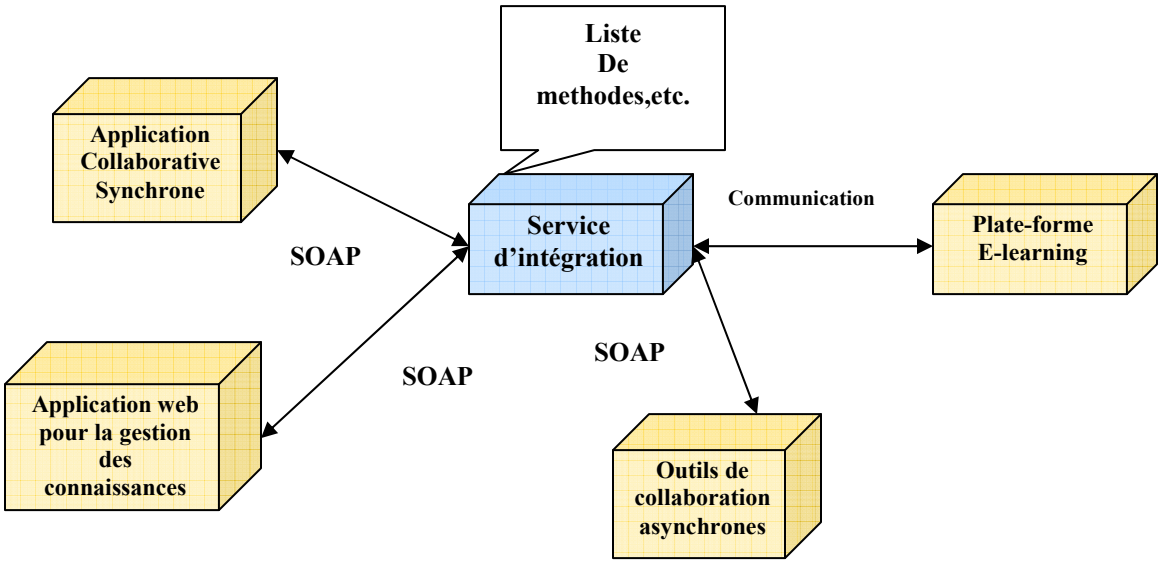

Figure 5.7 : Schéma général de l'intégration

Ce schéma montre qu'on peut interfacer deux plates-formes collaboratives par un service web d'intégration. Sa tache consiste à jouer deux rôles, c'est-à-dire d'un coté il joue le rôle du serveur et d'un autre coté le rôle du client.

5.6 La structure générale de la plate-forme

En s'appuyant sur le schéma d'intégration vu précédemment, on peut définir une architecture repartie dont le cœur est un service d'intégration.

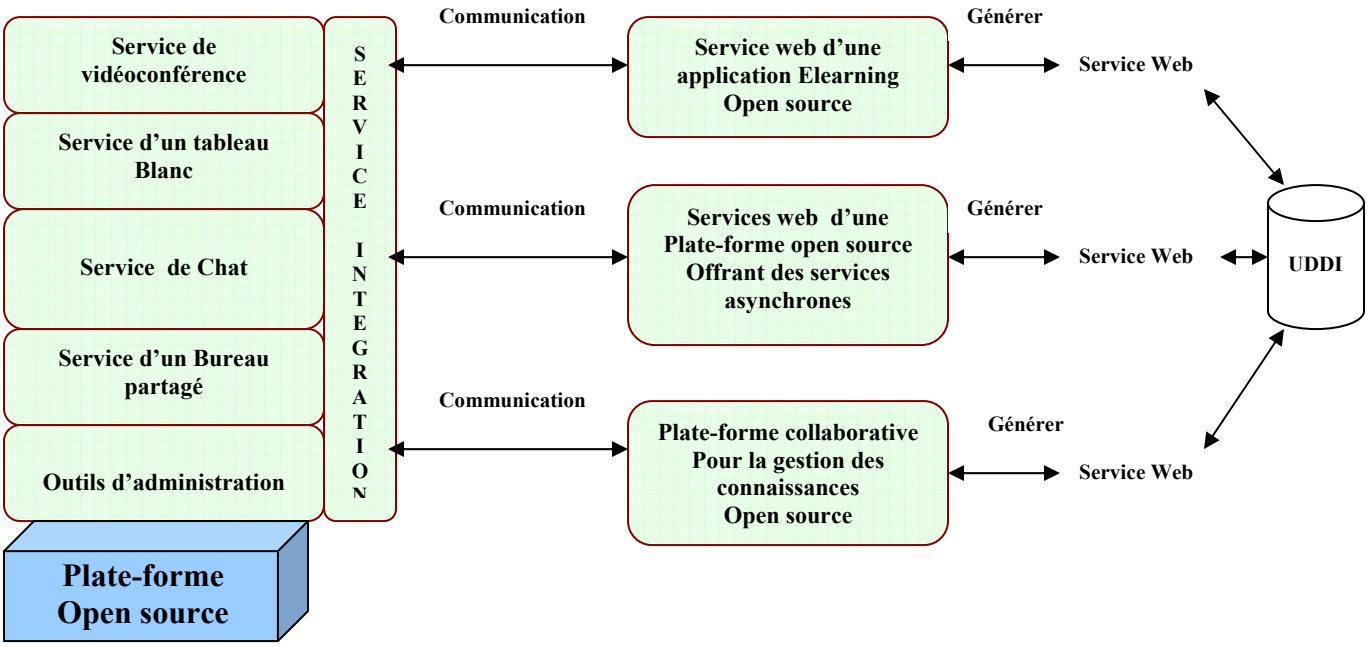

Figure 5.8 : L'architecture générale de la plate-forme

Le schéma précédant montre qu'on peut intégrer des applications collaboratives déjà existantes en open source dans une infrastructure collaborative basée services web, cela se fait en se basant sur les services web. Initialement on va définir un service web qu'on va nommer « service d'intégration », on va l'intégrer dans une plateforme collaborative open source basée sur les services de collaboration synchrone dans le but de rendre cette plate-forme capable de communiquer avec n'importe quelle plate-forme collaborative. Le rôle principal de ce service est de récupérer les paramètres nécessaires du serveur d'application de la plate-forme qu'on veut intégrer par la création d'un fichier XML qui englobe tous les paramètres concernant cette application.

Ce service peut être enregistré dans un annuaire UDDI comme étant un nouveau service qui sera prêt à être utilisé par d'autres applications clientes, ce registre sert donc de répertoire des applications intégrées, dans le quel on trouve les informations nécessaires pour communiquer avec le service généré (identificateur et URL).

5.7 Conclusion

Dans ce chapitre nous avons pu constater la difficulté associée à la conception d'une plate-forme collaborative extensible offrant plusieurs services de collaboration. Nous avons proposé une architecture générale basée sur l'intégration d'applications existantes (open source). Ces applications peuvent être des applications collaboratives client/serveur accessibles depuis le Web ou bien des applications déjà basées services web.

Pour mettre en œuvre cette approche, le meilleur choix est d'utiliser les technologies de services Web. Pour cela on peut utiliser un service web qui peut communiquer avec l'interface service web d'une autre application externe dans le but de générer un nouveau service accessible pour d'autres applications.

Chapitre 6
Expérimentation

6.1 Introduction

Dans le chapitre précédant nous avons vu la difficulté de concevoir une plate-forme collaborative offrant les différents outils de collaboration qui peuvent être utilisé dans plusieurs domaines nécessitant une collaboration tel que (la gestion des connaissances, l'enseignement à distance, etc).

Pour résoudre ce problème, nous avons proposé une architecture globale basée sur l'intégration d'outils open source en utilisant les services Web. Pour valider cette approche nous avons travaillé sur deux plates-formes open-source, la première s'appelle Openmeetings qui est une plate-forme collaborative basée services Web offrant les différents outils de collaboration synchrones (le chat, le tableau blanc partagé, la vidéoconférence et le bureau partagé), et la deuxième plate-forme s'appelle compendium destinée pour la gestion des connaissance.

Dans une première partie, nous présentons les deux outils open source qui sont en fait la base de notre implémentation en décrivant les deux architectures avec les différents modules existants. Dans une deuxième partie nous allons appliquer notre approche d'intégration sur ces deux plates-formes.

6.2 Les Bases d'Openmeetings

Openmeetings [39] est une application open source basée sur le client riche qui offre des services de collaboration synchrones :

- Une vidéoconférence.
- Un tableau blanc partagé avec des capacités de dessin, d'écriture et d'édition, de glissé/déposé, l'intégration d'images depuis une bibliothèque et l'ajout de symboles.
- La possibilité de sauvegarder le tableau entre deux sessions.
- Un bureau partagé.
- L'importation de documents (jpg, .jpeg, .gif, .png, .ppt, .odp, .odt, .sxw, .wpd, .doc, .rtf, .txt, .ods, .sxc, .xls, .sxi, .pdf).
- L'envoi d'invitations et de liens direct à une rencontre.
- La modération.
- Un système d'organisation des participants.
- La possibilité de créer des salles de conférence publiques ou privés.
- Multi-langue et personnalisable.

Cette plate-forme a été développée en OpenLaszlo (voir chapitre Architecture proposée, partie client riche), dans le but de réaliser une architecture qui comprend à la fois la technologie des clients riches et la technologie des architectures orientées services.

6.2.1 L'architecture d'Openmeetings

L'architecture générale de la plate-forme est une architecture orientée service qui offre à l'utilisateur un service de vidéoconférence qui se base sur un serveur flash red5 utilisé pour la diffusion de l'audio et de la vidéo, un service de tableau blanc et un bureau partagé pour compléter le travail collaboratif entre les différents utilisateurs (travailler sur des images, un fichier PDF, un fichier power point ou world), pour cela plusieurs convertisseurs de documents sont utilisés (ImageMagick,Ghostscript,SWFTools) voir schéma suivant .

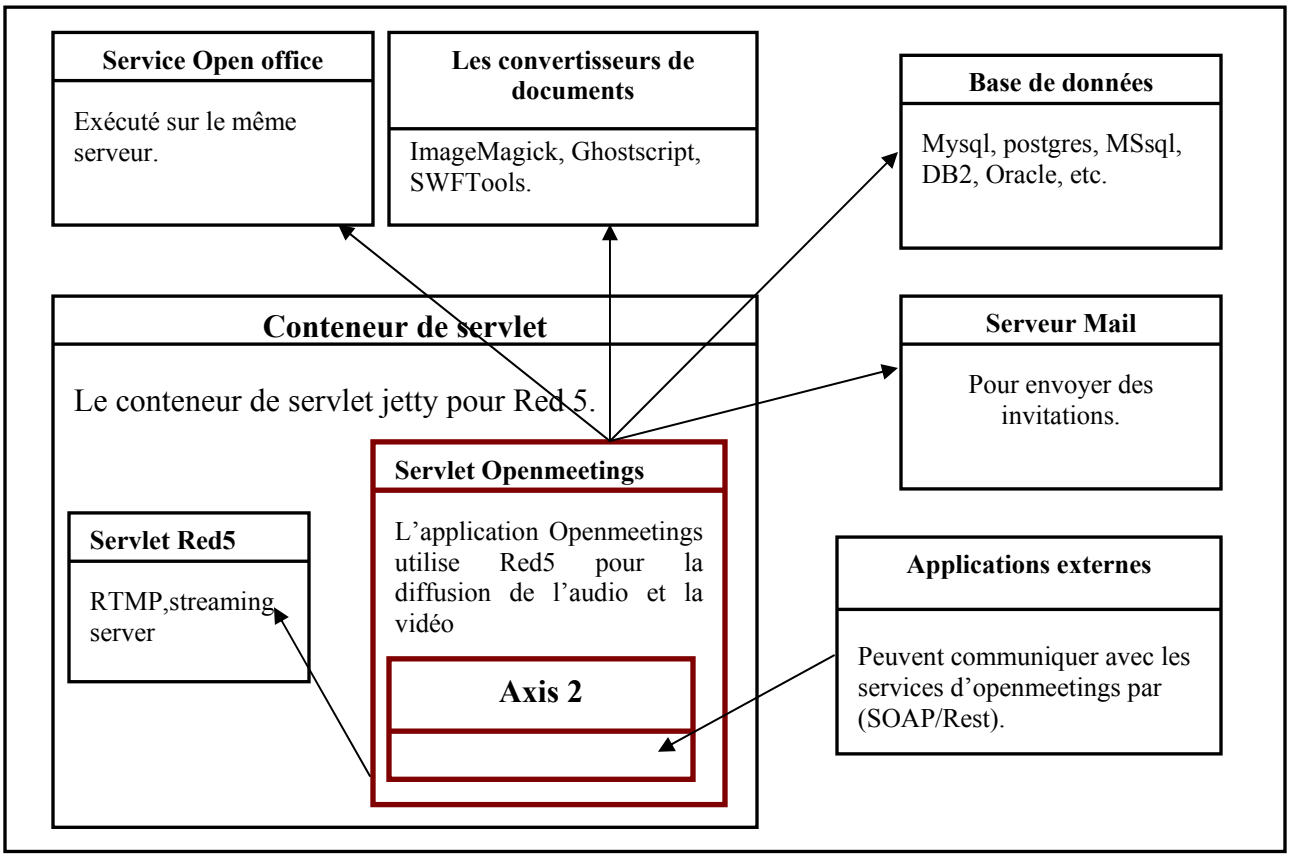

Figure 6.1 : Architecture générale d'Openmeetings

Ce schéma montre l'extensibilité de la plate-forme Openmeetings car grâce à son architecture, elle offre à d'autres applications externes la possibilité de communiquer avec elle en utilisant les messages SOAP et Rest car son cœur est basé sur Axis2.Elle peut aussi communiquer avec plusieurs bases de données tel que Mysql, postgres, MSsql et oracle, et avec un serveur Mail si un utilisateur veut inviter d'autres participants à rejoindre la conversation.

6.2.2 L'interaction par les services Web

Dans la partie juste au dessus, on a vu que l'architecture d'Openmeetings est basée sur l'utilisation des services Web. Dans cette partie on va voir le schéma général d'interaction entre la plate-forme et le client SOAP.

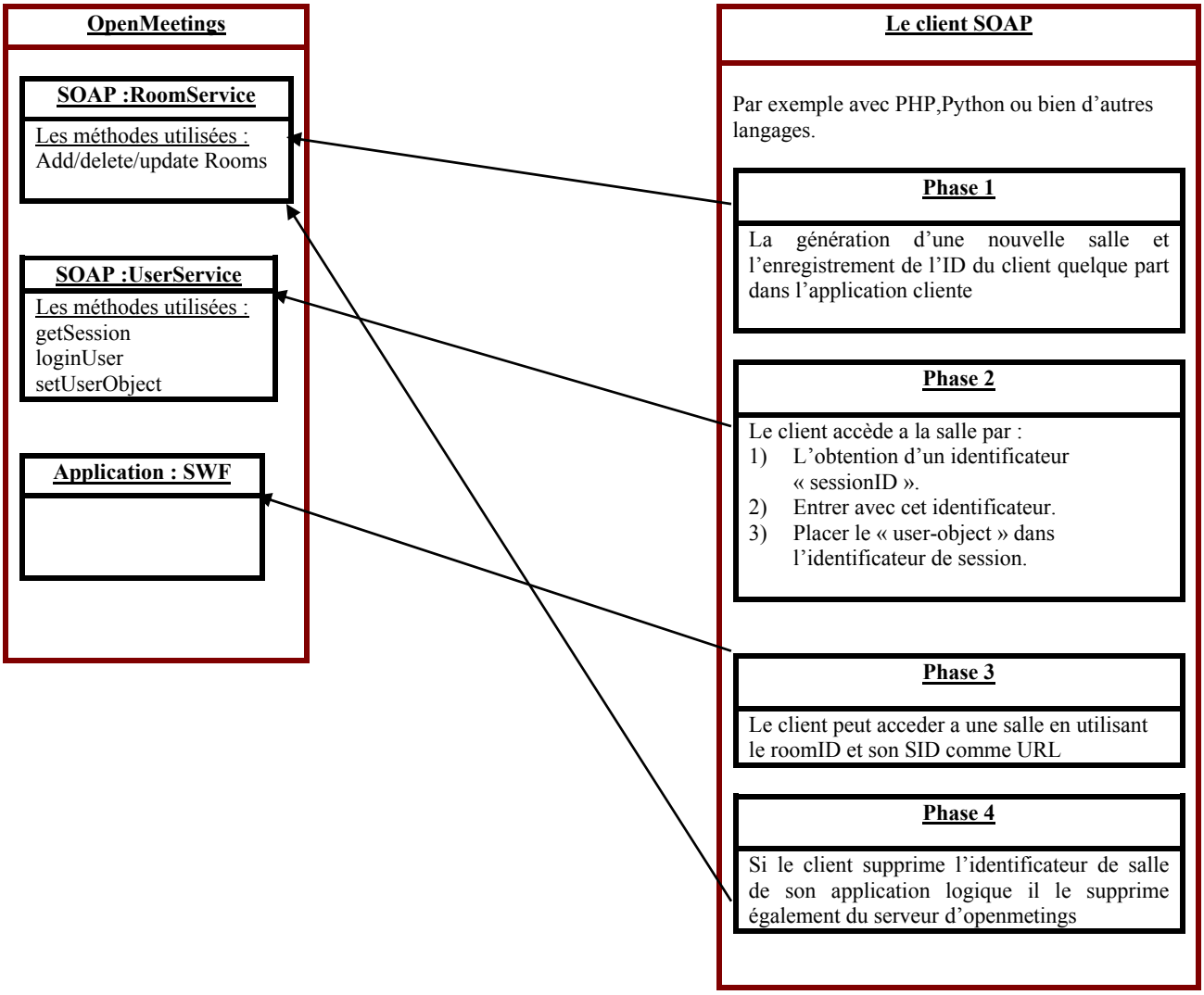

Figure 6.2 : L'interaction entre le serveur de l'application et le client SOAP

Ce schéma illustre l'interaction entre le client SOAP et le serveur de l'application, cette interaction se fait en quatre phases :

Phase 1 : lorsque l'utilisateur se connecte, il peut créer une nouvelle salle de conférence cela se fait en appelant le service de salle « Roomservice ».

Phase 2 : le client accède à la salle en suivant ces étapes :

- L'obtention d'un identificateur « sessionID ».
- Accéder à la salle par cet identificateur.
- Placer le « user-object » dans l'identificateur de session.

Cela se fait en appelant le service utilisateur.

Phase 3 : le client peut accéder à n'importe quelle salle en utilisant son identificateur et l'identificateur de la salle comme URL.

Phase 4 : l'utilisateur peut sortir d'une salle en supprimant l'identificateur de la salle de son application logique, cela se fait en appelant le service de la salle.

6.2.3 L'implémentation des services Web dans Openmeetings

Openmeetings offre aux clients un ensemble de services parmi lesquels on peut citer : Service de chat, service de vidéoconférence, service de configuration, service de tableau blanc partagé, service de bureau partagé, service pour la configuration de la langue utilisée, service pour la gestion des erreurs, service d'organisation, service utilisateur, service pour la gestion des fichiers, etc.

Comme notre but est d'ajouter un nouveau service on va se baser sur trois types de services qui jouent un rôle très important dans l'intégration des applications externes.

6.2.3.1 Les services utilisateur (Userservice)

Ce service contrôle les utilisateurs, les sessions et les erreurs. Dans ce service on trouve plusieurs méthodes :

getSession :

La méthode : public Sessiondata getSession ()

La description : elle charge l'identifiant de la session « sessionID » avant de faire toute autre opération puis renvoie un objet de type Sessiondata.

Paramètres : pas de paramètre.

loginUser :

La méthode : public Long loginUser(String SID, String username, String userpass)

La description: utilise le SID renvoyé par la méthode getSession, renvoie une valeur positive, si négative alors c'est une erreur, et la il faut appeler la méthode getErrorByCode.

Les paramètres :

type	nom	description
string	SID	Le SID renvoyé par getSession.
string	username	C'est le nom d'utilisateur de l'application, l'utilisateur doit avoir les droits de l'administration.
string	userpass	Le mot de passe de l'application.

Tableau 6.1 : Les paramètres de la méthode « LoginUser »

getErrorByCode :

La méthode: public ErrorResultgetErrorByCode (String SID, Long errorid, Long language_id)

La description: elle charge un objet d'erreur (error-object) si une méthode renvoie un résultat négatif (l'identifiant d'erreur ou bien « errorid »).Elle a besoin d'un identifiant de langage (language_id) pour spécifier dans quelle langue le message d'erreur s'affiche.

Les paramètres :

type	nom	description
string	SID	Le SID renvoyé par getSession.
Long	errorid	La valeur négative renvoyée par n'importe quelle méthode.
Long	Language_id	C'est l'identificateur de langage.

Tableau 6.2 : Les paramètres de la méthode « getErrorByCode »

setUserObject :

La méthode: public Long setUserObject(String SID, String username, String firstname, String lastname, String profilePictureUrl, String email)

La description: place un objet de session « sessionobject » pour un certain SID, après on peut utiliser le SID et l'identifiant de la salle « roomid » pour entrer dans n'importe quelle pièce.

Les paramètres :

type	nom	description
String	SID	L'identifiant de la session.
String	username	N'importe quel nom d'utilisateur.
String	firstname	Le prénom.
String	lastname	Le nom.
String	profilePictureUrl	Le lien vers l'image du profile.
String	email	L'email.

Tableau 6.3 : Les paramètres de la méthode « SetUserObject »

6.2.3.2 Les services de fichier (fileservice)

Dans ce service on trouve plusieurs méthodes :

getListOfFiles:

La méthode: public LiberaryObject getListOfFiles(String SID, String moduleName,String parentFolder, Long room_id)

La description: renvoie un objet de bibliothèque, cet objet contient des dossiers et une liste de fichiers et il peut contenir un objet de présentation.

L'objet de présentation est une liste de petites images (thumbnails) c'est-à-dire la version de l'image dont la taille est réduite par rapport à l'originale avec la référence des fichiers SWF et PDF convertis.

Les paramètres :

type	nom	description
String	SID	Le SID de l'utilisateur, il doit être marqué comme loggedin.
String	moduleName	La valeur est « videoconf1 ».
String	parentFolder	Le dossier parent, pour cela on utilise '/'.
Long	Room_id	L'identifiant de la salle la ou on veut afficher les fichiers.

Tableau 6.4 : Les paramètres de la méthode « getListOfFiles »

deleteFile:

La méthode : public Boolean deleteFile(String SID, String fileName, String moduleName, String parentFolder, Long room_id)

La description: supprime les fichiers et les dossiers.

Les paramètres :

type	nom	description
String	SID	Le SID de l'utilisateur, il doit être marqué comme loggedin.
String	fileName	Le fichier ou le dossier a supprimé.
String	moduleName	Le nom du module, la valeur « videoconf1 » si on est dans une salle de conférence.
String	parentFolder	Le dossier père.
Long	Room_id	L'identifiant de la salle la ou se trouve le fichier.

Tableau 6.5 : Les paramètres de la méthode « deleteFile »

6.2.3.3 Les services de la salle (RoomService)

getRoomsPublic :

La méthode : public RoomsList getRoomsPublic(String SID, Long roomtypes_id)

La description: Renvoie un objet de type RoomsList qui contient une liste d'objets (Room-Objects), chaque objet contient le type de la salle avec toutes les informations concernant cette salle.

Le type de la salle peur être « 1 » pour la salle de conférence et « 2 » pour la salle d'audience.

Les paramètres :

type	nom	description
String	SID	Le SID de l'utilisateur, il doit être marqué comme loggedin.
Long	Roomtypes_id	Le type de salles que vous voulez obtenir.

Tableau 6.6 : Les paramètres de la méthode « getRoomsPublic »

getRoomTypes :

La méthode : Public List getRoomTypes(String SID)

La description : Renvoie une liste d'objets.

Les paramètres :

type	nom	description
String	SID	Le SID de l'utilisateur, il doit être marqué comme loggedin.

Tableau 6.7 : Les paramètres de la méthode « getRoomTypes »

getRoomByld :

La méthode : public Rooms getRoomById(String SID, long rooms_id)

La description: renvoie un objet de Rooms.java

Les paramètres :

type	nom	Description
String	SID	Le SID de l'utilisateur, il doit être marqué comme loggedin.
Long	Rooms_id	L'identifiant de la salle que vous voulez obtenir.

Tableau 6.8 : Les paramètres de la méthode « getRoomByld »

Il ya aussi la méthode getRooms, addRoom, updateRoom et deleteRoom.

6.3 La gestion des connaissances avec compendium

Compendium [41] est un médium visuel dialogique de modélisations du discours concernant des problèmes à résoudre, c'est un outil qui, entre les mains d'utilisateurs compétents, facilite la capture et la structuration des idées, et permet non seulement de modéliser le discours, mais aussi de modéliser les domaines de problèmes d'une manière qui invite et structure les contributions, et ce de manière synchrone ou asynchrone. Il est optimisé afin d'être utilisé dans ce qui constitue sans doute l'environnement d'utilisation le plus extrême pour un outil de représentation de la connaissance : la modélisation collaborative en temps réel. Le logiciel est une application Java gratuite fonctionnant sur toutes les plates-formes et incluant le code source. Les

Téléchargements et les autres ressources communautaires sont coordonnés par le Compendium Institute, une association à but non lucratif.

L'approche Compendium porte l'accent sur la modélisation collaborative de l'information, des idées et arguments, afin d'obtenir un ajout de valeur immédiat pour les utilisateurs (mémoire à court terme utile), ainsi que de poser les bases d'une mémoire à long terme nécessaire à la gestion des connaissances. L'affichage de Compendium présente un certain nombre d'affordances (La capacité d'un objet à suggérer sa propre utilisation) visuelles permettant de saisir d'un trait l'information concernant l'état d'une analyse, ce qui n'est immédiatement évident ni dans un document textuel traditionnel, ni dans d'autres approches de cartographie conceptuelle. Cela inclut les problèmes irrésolus, les idées concurrentes, dans quelle mesure des faits extérieurs sont utilisés pour renforcer les idées, ainsi que la 'profondeur' de réutilisation de nœuds et d'étiquetage (un indicateur du degré de modélisation utilisé). Compendium permet d'exporter le Map sous forme HTML, XML, JPG etc. Compendium est actuellement utilisé dans des grands projets tels que la représentation des métadonnées scientifiques sur l'exploration de la planète Mars de la NASA.

6.3.1 Les bases du logiciel

Cette partie présente une introduction générale sur les types de nœuds de compendium, les approches de liens hypertextes, la recherche, l'importation et l'exportation et d'autres mécanismes du logiciel.

6.3.1.1 Nœuds, liens et vues

Dans compendium les idées sont exprimées sous forme d'icones (ou nœuds), elles peuvent être déplacées et être reliées à d'autres idées par l'intermédiaire des liens.
On trouve les nœuds dans ce qu'on appelle des vues, une vue peut contenir plusieurs autres vues.il ya deux type de vue :

- Une carte dans la quelle on peut placer les nœuds n' importe où dans un espace 2D.
- Une liste qui organise les nœuds dans une colonne classable.

Tous les nœuds peuvent être crées par un click droit de la sourie et par un raccourci clavier pour les utilisateurs les plus expérimentés.

Le tableau suivant montre les principaux types de nœud qui existe dans compendium.

Type de nœud	Quant est ce qu'on l'utilise	Raccourcis clavier
Question	Pour poser une ou lire une question dans votre domaine.	Q, ?
Idée	Pour donner une réponse.	A, !
Liste	-Pour créer une liste de nœuds. -Pour placer les résultats d'une recherche (ex : pour créer un -catalogue d'articles). -Pour créer une collection de nœuds qui n'ont pas besoin d'être lié a d'autres nœuds (les liens associatifs).	L
Carte	Pour créer une image des rapports entre les idées. Pour grouper les idées et les questions ensemble. Pour créer des liens associatifs entre les nœuds.	M

Tableau 6.9 : Les différents types de nœuds

Il existe aussi d'autres types de nœuds tels que :

- **Pro (+):** pour partager ou soutenir une idée.
- **Con (-):** pour plaider contre une idée.
- **Reference(R) :** pour créer un lien vers un dossier externe (ex : document, image, bilan,….etc.).
- **Note (N):** pour fournir des informations supplémentaires au sujet d'un autre nœud ou de la vue courante.
- **Decision(D)** : pour résoudre une question : soit faire le lien entre la question et l'idée soit un click droit sur l'idée pour la transformer en nœud de décision.

Le schéma suivant montre les différents types de nœuds :

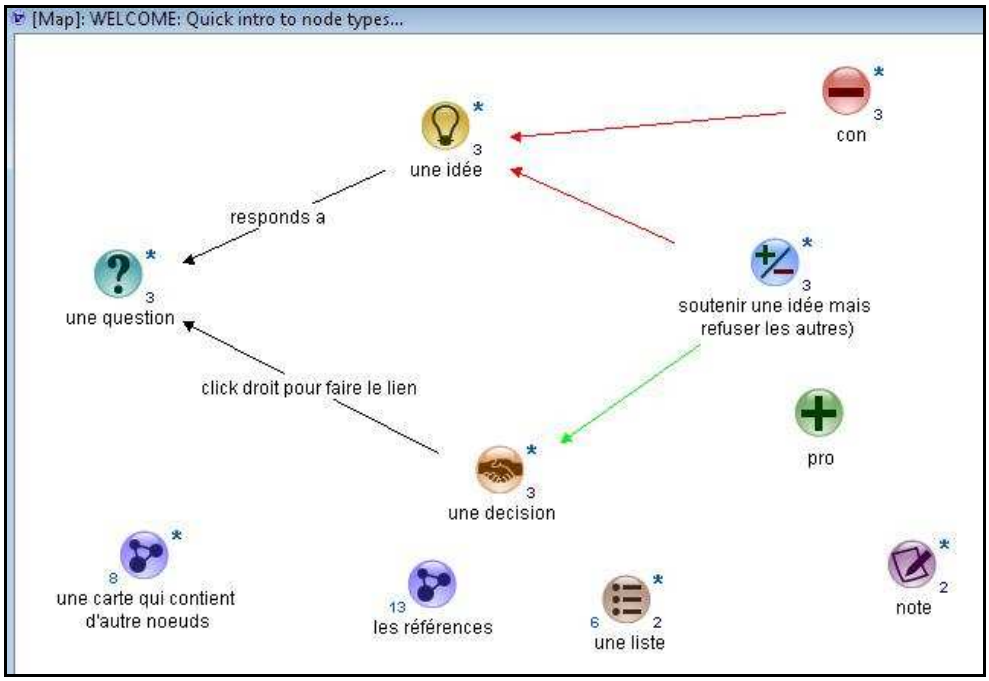

Figure 6.3 : Les différents types de nœuds

6.3.1.2 Les nœuds de références

Les nœuds de références sont utilisés pour lier les documents aux sites web, pour utiliser ces références il faut simplement ajouter le fichier ou le dossier dans une liste ou dans une carte.si compendium reconnaitra le type de fichier, il lui donnera la bonne icône.

La carte ci-dessous montre les différents nœuds de références :

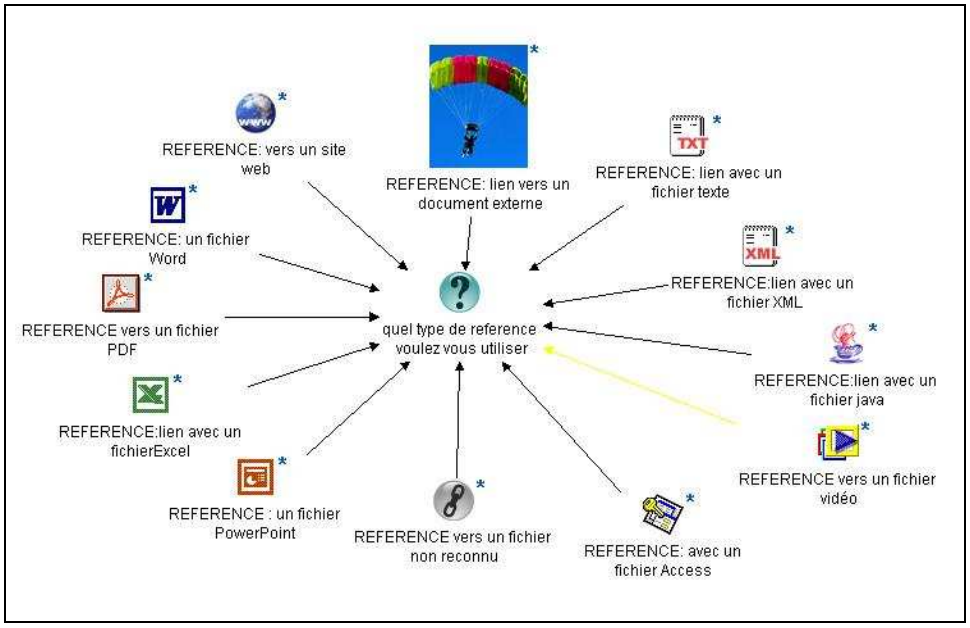

Figure 6.4 : Les différents types de références

La création d'un nœud de référence se fait en appuyant sur la touche R, et puis en spécifiant le dossier ou l'URL de référence manuellement.

6.3.1.3 Les principes des liens hypertexte dans compendium

Compendium permet de réaliser des rapports entre les informations de plusieurs manières :

- **Associative** : lien entre les idées dans le même contexte.
- **Transclusive** : lien entre une même idée dans des contextes différents.
- **Catégorique** : lien entre les idées qui appartiennent à la même catégorie.

6.3.2 Les techniques utilisées par compendium

Dans cette partie on va présenter les trois techniques de base de compendium utilisées pour contrôler l'information et les idées.

6.3.2.1 L'étiquetage de nœuds

Ajouter des étiquettes aux nœuds facilite la recherche, les correspondances et surtout la création des cartes et des listes automatiquement c'est-à-dire sans avoir utilisé le travail manuel.

6.3.2.2 La recherche

La recherche est une manière automatique pour trouver les idées dans la base de données. Compendium parcourt toutes les cartes et les listes pour trouver le nœud qui correspond aux critères de recherche. Les critères de recherche peuvent être par mot-clé, par auteur, par type de nœud, par date de création ou de modification, etc. Une des utilisations les plus puissantes de la recherche est de construire des catalogues de nœuds qui partagent des étiquettes communes.

6.3.3.3 Les catalogues (listes)

Les catalogues sont des collections de nœuds dans la même ou les mêmes catégories. Un catalogue est typiquement produit en recherchant une base de données de nœuds avec une ou plusieurs étiquettes, et en insérant les résultats dans une liste. Les nœuds individuels peuvent naturellement être ajoutés manuellement.

Par exemple une université peut maintenir des catalogues de types de corps enseignant, de cours, de régions, de supports et du personnel.

6.3.3.4 La publication des cartes sur le web

Le world wide web est la manière la plus simple pour accéder aux informations en fournissant les ressources par un navigateur web.

Compendium peut convertir ses projets en deux formats web :

Une liste textuelle des nœuds : visualisables par tous les navigateurs web**,** et sous la forme désirée (ex : Microsoft Word),

Les cartes visuelles : elles peuvent être consultées à traves un navigateur web (internet explorer seulement) voir figure suivante.

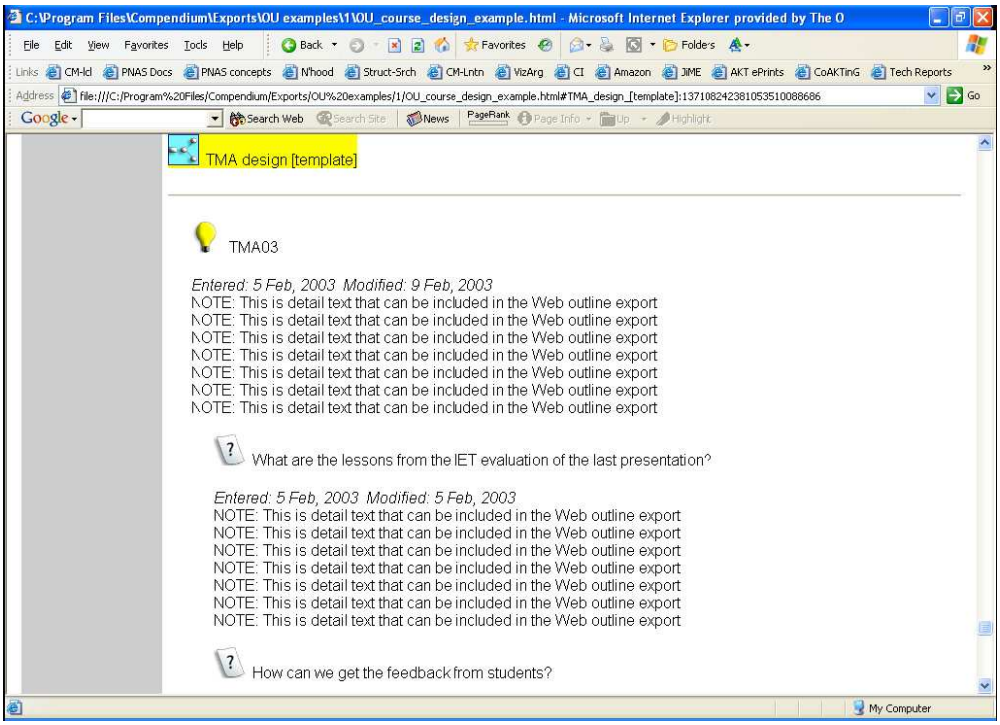

Figure 6.5 : L'exportation des données sur le web

Cette figure montre l'exportation des données sur le web, on peut ouvrir ces données avec Microsoft World en raison d'une modification par exemple ou d'ajout d'une image, etc.

6.4 Rendre compendium accessible par le web

Compendium est une application client serveur qui s'exécutera sur n'importe quel système d'exploitation, elle est très puissante en terme de structuration des idées, donc elle offre un très grand avantage pour permettre la collaboration entre plusieurs utilisateurs qui ne se trouvent pas au même endroit, cela en facilitant la communication et le partage des idées entre eux dans divers domaines (exemple : l'enseignement à distance, gestion des connaissances, domaine médical, etc.).

Le problème majeur de cette plate-forme c'est qu'elle est conçue d'une manière standalone c'est-à-dire elle n'offre pas une interface avec le Web. Pour résoudre ce problème et rendre cette

application accessible par le Web, Simon Buckingham et All [41] ont proposés une nouvelle approche qui repose sur l'exportation des cartes sous forme HTML.

Dans la partie suivante on va définir cette approche et on va proposer une nouvelle approche qui intègre compendium avec Openmeetings dans le but de définir un environnement collaboratif basé outils synchrones et intégrant des possibilités de collaboration dans des divers domaines.

6.4.1 La nouvelle architecture de compendium

Pour rendre cette plate-forme accessible depuis le Web une nouvelle approche a été proposée par le compendium institute qui repose sur la création d'une nouvelle version de compendium qui s'exécutera sur un serveur web cela en se basant sur l'exportation des données sur le web (voir figure suivante).

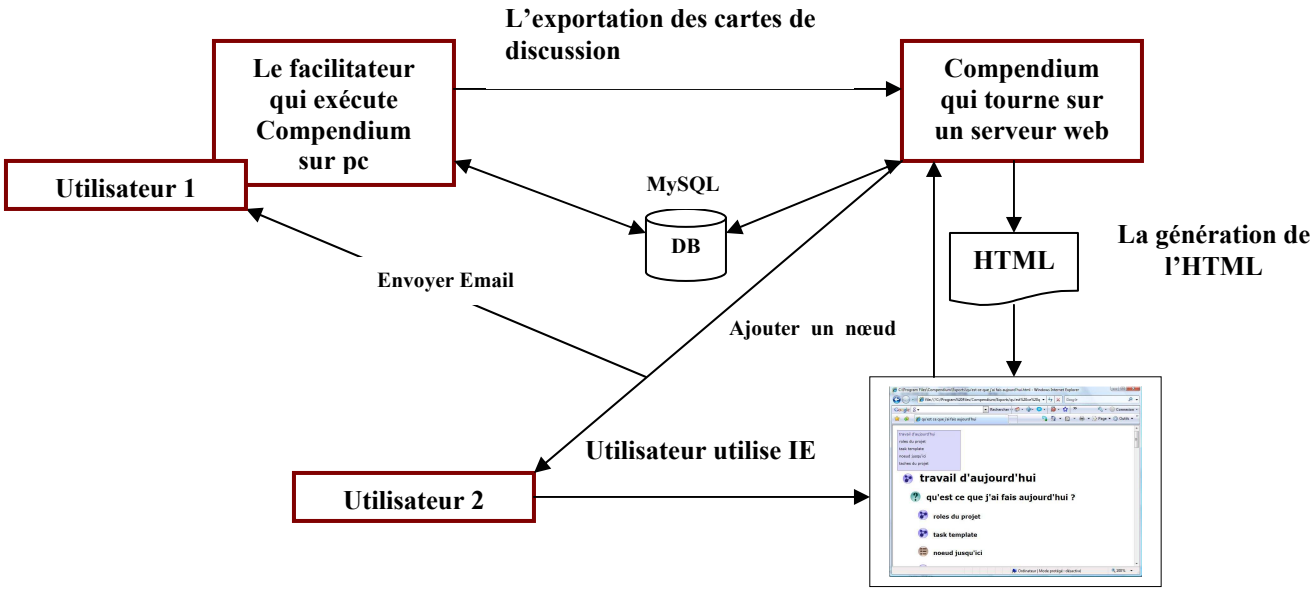

Figure 6.6 : L'accessibilité de compenduim depuis le Web

Ce schéma illustre l'extension de compendium vers le Web .Le serveur qui exécute le Web compendium et qui se connecte à la base de données MySQL de compendium reçoit les cartes qui ont été exportées et génèrera après leurs codes en html sur le site Web. L'ajout des nœuds sur le site génère un email pour tous les utilisateurs. Le Web compendium et la version facilitateur de

compendium sur PC se connectent sur la même base de données. Les utilisateurs peuvent utiliser internet explorer pour participer à une discussion, cela se fait sans avoir installé l'application sur leurs ordinateurs. Après l'ajout d'un nœud, la base de données MySQL se met à jour automatiquement par la régénération des cartes et l'actualisation des navigateurs web de chaque participant.

6.4.2 Changement du code source

L'ancien code source de compendium a été changé de façon à garder ses différentes fonctionnalités. Le seul changement était par l'élimination de la deuxième identification de l'utilisateur. L'idée de l'exportation des cartes a été crée par l'utilisation d'une étiquette qui s'appelle ″DISCUSSION_MAPS_PARENT″. Une seule carte dans la base de données peut avoir cette étiquette. Cette carte avec les autres sous cartes sont exportées sur le web quant le facilitateur sélectionne le ″export to web″.

Pour rendre le facilitateur et la version serveur web de compendium simple et flexible. Un fichier de propriétés est utilisé, ce dernier contient le nom de l'utilisateur et le mot de passe, le nom de la base de données, le nom de l'administrateur, avec un ensemble de variables utilisés par l'application (voir figure suivante).

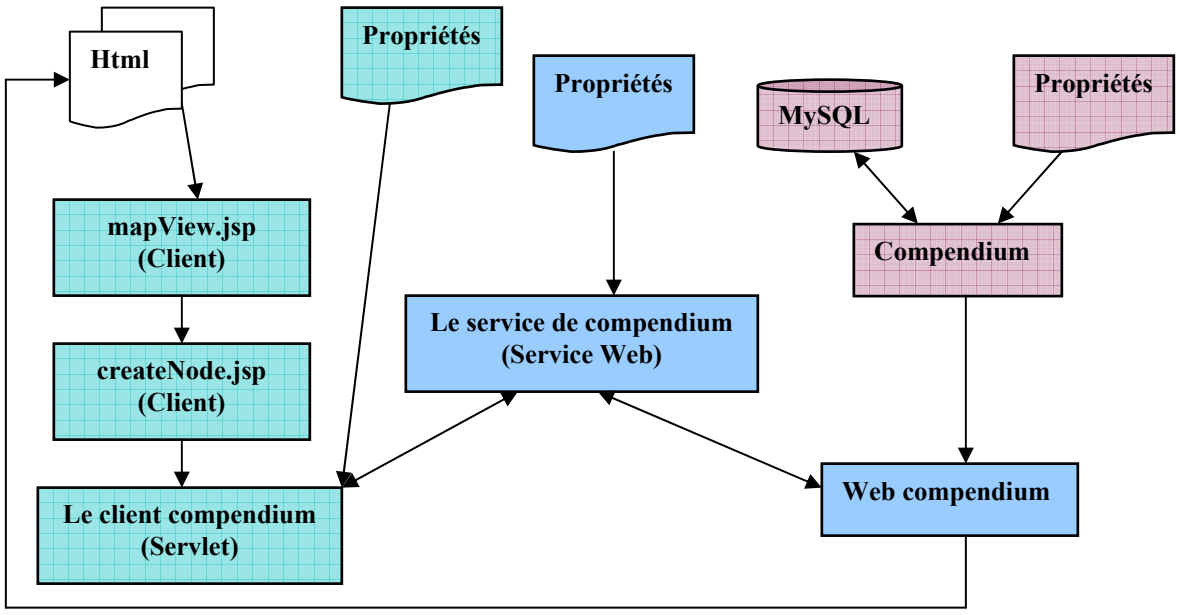

Figure 6.7 : L'architecture de l'application web de compendium

Ce schéma nous montre l'architecture générale de la plate-forme compendium destinée pour le web, la communication entre les composants se fait de manière hiérarchique, le facilitateur compendium communique avec un ensemble de propriétés, avec la base de données, et enfin avec le Web compendium par l'exportation des cartes. Le web compendium à son tour communique avec le web service compendium qui lui permettra de générer un code html pour le client.

6.4.3 Le développement du serveur Web compendium

La version facilitateur de compendium et le site web utilisé par les utilisateurs doivent envoyer des requêtes au service web du web compendium. Le client compendium crée le message SOAP du client. La servlet « clientcompendium.jsp » constitue le client du service web du web compendium. La nouvelle classe qui hérite les propriétés de la classe « UIDialog » représente le service web client de la version facilitateur de compendium. Le service web génère deux services ; AddNode (ajouter un nœud) et exportNodes (exporter un nœud), après il fait des appels pour exécuter le web compendium.

Compendium tourne sur le serveur en communiquant avec la même base de données utilisée par le facilitateur. Pour éviter la collision, le service web compendium est synchronisé de manière à accepter qu'une seule requête lorsqu'il ya plusieurs requêtes au même temps.

6.5 La communication entre les deux plates-formes

Pour garder les différentes fonctionnalités des deux plates-formes (Openmeetings et compendium), et réaliser une plate-forme qui intègre à la fois les différents services offerts par Openmeetings et le service de gestion des connaissances de compendium qui permet d'enrichir la communication et la collaboration entre participants, nous avons créer un service web appelé « service d'intégration » qui joue le rôle d'interface entre les deux plates-formes (voir figure suivante).

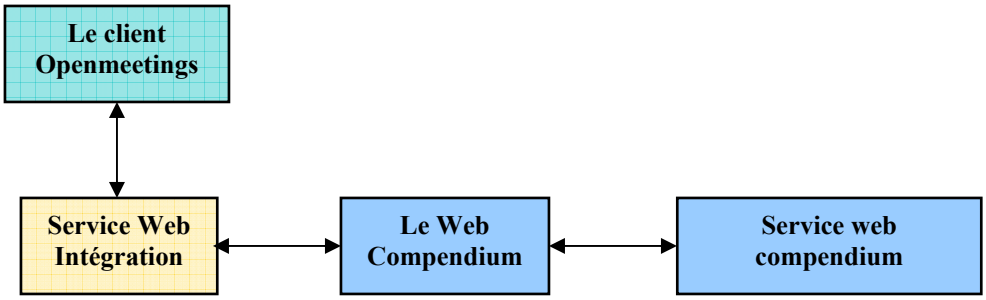

Figure 6.8 : La communication à l'aide d'un service web

Le service d'intégration joue deux rôles, lorsque le modérateur crée une salle compendium avec l'identifiant « 3 » cela se fait en créant une nouvelle session (appelle du service session et service de la salle), le service d'intégration va communiquer avec le web compendium qui lui permettra de générer une vue pour le client Openmeetings.

6.6 Quelques scenarios de test

Pour déployer la plate-forme il faut installer un serveur flash qui s'appelle Red5, il permet grâce à son protocole RTMP (REAL Time Messaging Protocol) de faire du streaming audio et vidéo. Installer XAMP qui est un ensemble de logiciel permettant de mettre en place un serveur web qui englobe (Apache et Mysql), un serveur FTP et un serveur de messagerie (voir figure suivante).

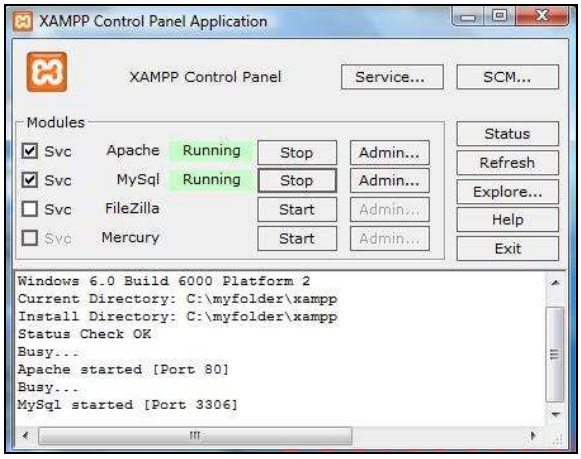

Figure 6.11 : Le lancement du XAMPP

Lors du lancement de l'application une fenêtre d'authentification s'ouvre, pour demander le nom d'utilisateur et le mot de passe (voir figure suivante).

Figure 6.12 : La fenêtre d'authentification

Apres la validation du nom de l'utilisateur et du mot de passe, l'utilisateur peut accéder à n'importe quelle salle (voir figure suivante).

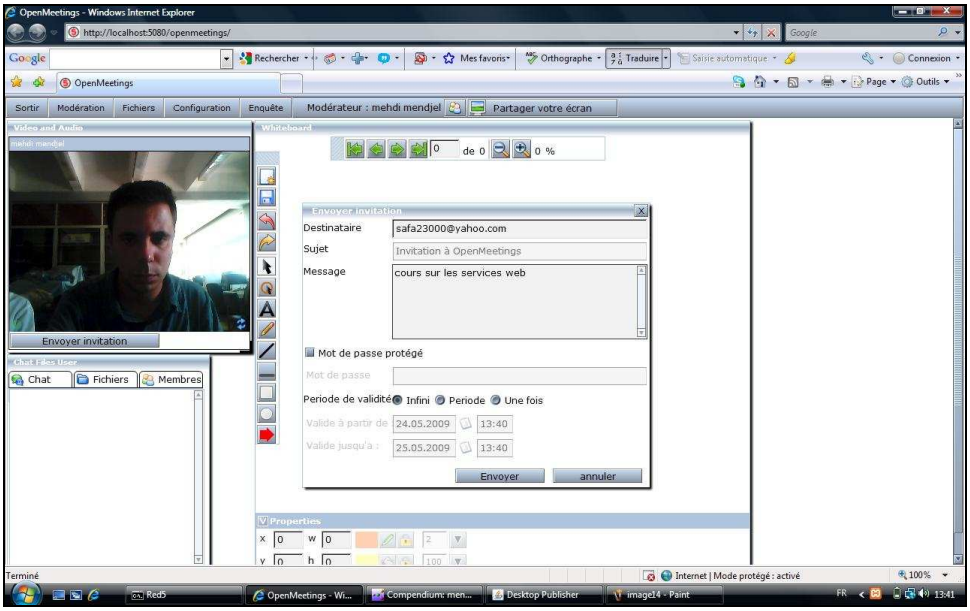

Figure 6.13 : L'administrateur accède à l'auditorium

Cette figure montre une interface de la salle d'auditorium où l'utilisateur peut envoyer des invitations à d'autres utilisateurs pour rejoindre la conversation. Cette interface est composée d'un ensemble de modules :

- Un module de chat, pour permettre à l'ensemble des utilisateurs de communiquer entre eux en envoyant des messages.
- Un tableau blanc, pour permettre au modérateur de faire des démonstrations en cas de nécessité.
- Un module d'invitation utilisé pour inviter les participants à rejoindre la conversation.

Un utilisateur peut aussi partager son bureau en faisant appelle au service bureau partagé par un seul clic sur le bouton « partager ton écran ».

Une fenêtre de propriété s'ouvre, pour donner à l'utilisateur la possibilité de choisir la manière avec laquelle il veut partager son écran.

Cette fenêtre englobe un ensemble de propriétés :

- Un bouton pour démarrer et arrêter le partage d'écran.
- La possibilité de choisir la dimension de la fenêtre de partage.

- La qualité de partage (voir figure suivante).

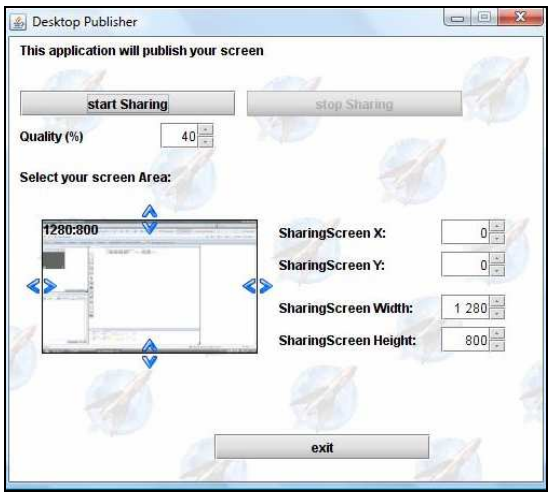

Figure 6.14 : Les propriétés de l'écran à partager

Apres le démarrage du partage d'écran les participants peuvent voir l'écran partagé.

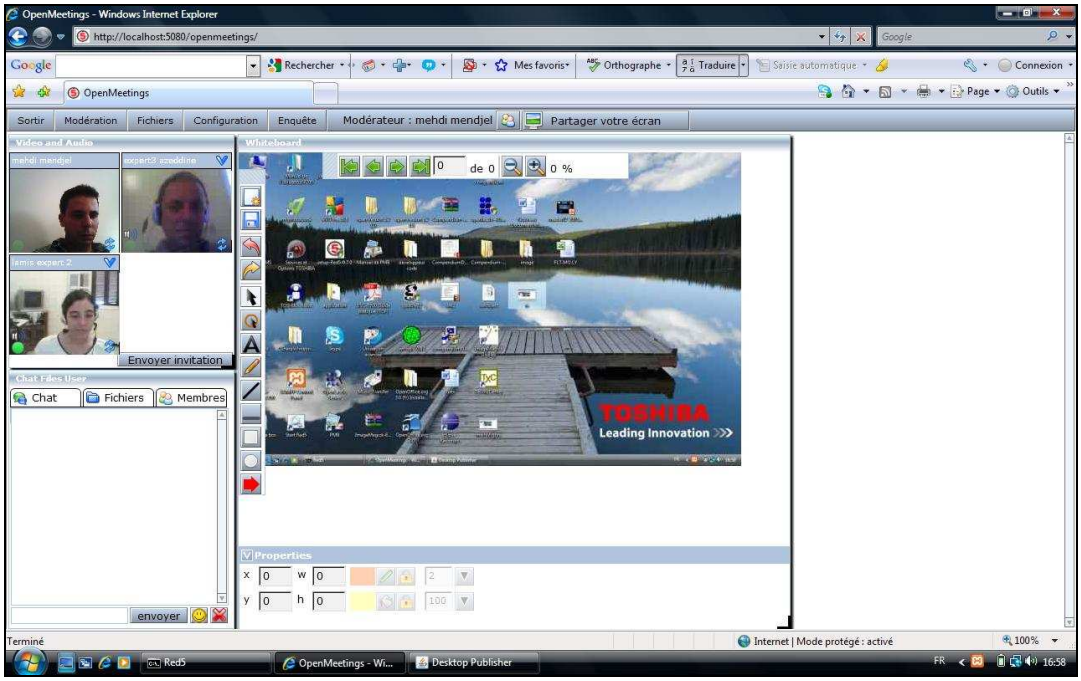

Figure 6.15 : Partage d'écran par un utilisateur

6.7 L'adaptation de la plate-forme aux besoins des clients

Pour rendre cette nouvelle plate-forme plus flexible et adaptable aux besoins des participants, on a suivi le même principe implémenté dans openmeetings, c'est-à-dire lorsqu'un client veut accéder à une salle, un fichier XML se générera automatiquement comprenant les différentes fonctionnalités de la salle choisie par le participant (voir figure suivante).

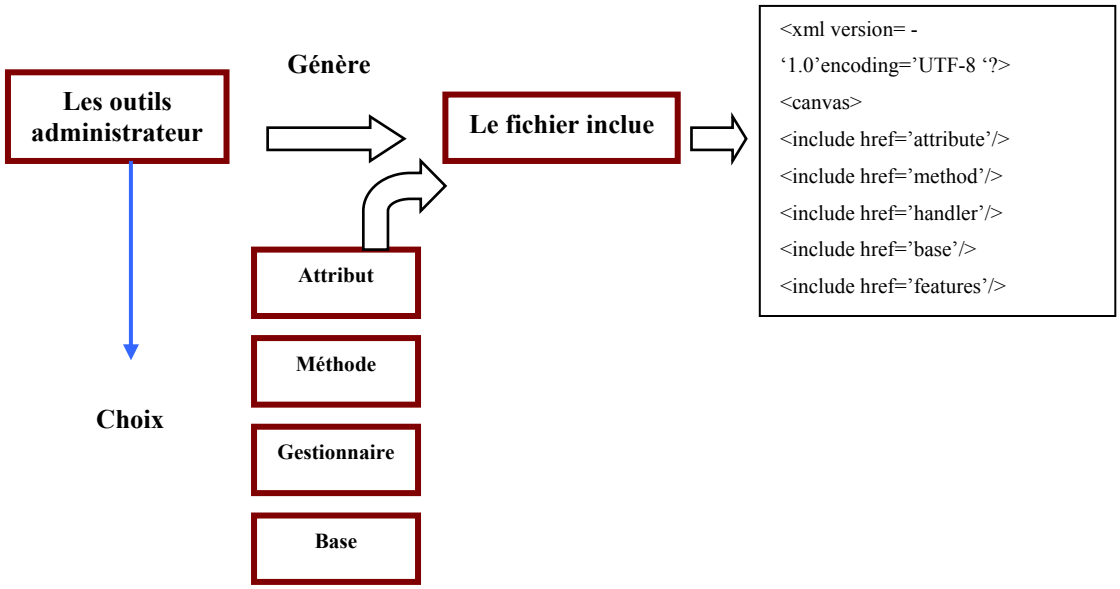

Figure 6.16 : Les étapes d'adaptation de la plate-forme

Ce schéma illustre les différentes étapes d'adaptation :

- On peut choisir n'importe quel dispositif en utilisant l'outil d'administrateur.
- Le fichier inclue se génèrera automatiquement selon les choix.
- Le fichier main.lzx compilé en un fichier swf pour créer le module voulu, ce module sera appelé au même temps que la salle de conférence.

Le schéma suivant illustre un scénario d'adaptation :

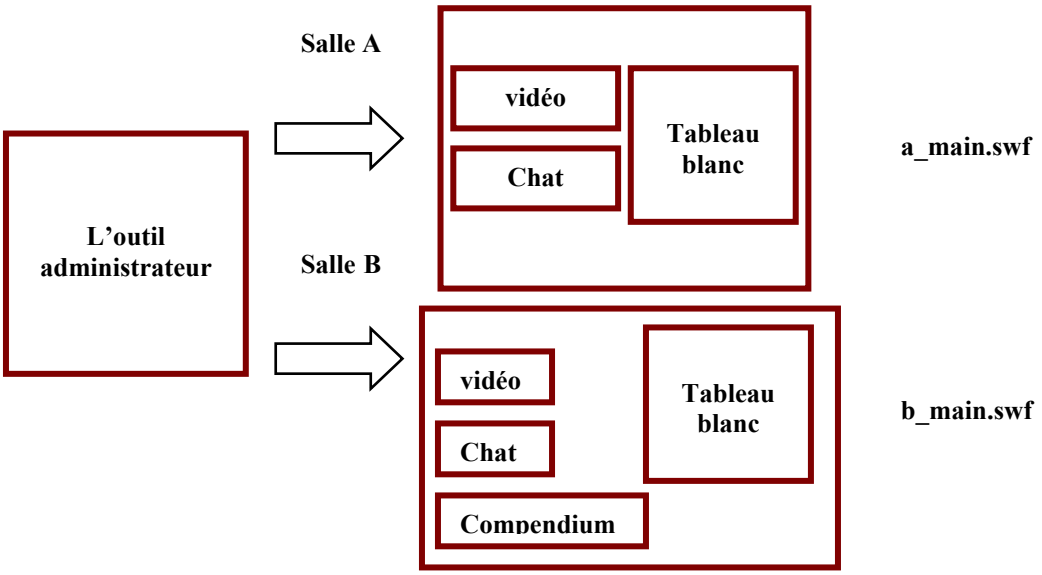

Figure 6.17 : L'adaptation selon les besoins du client

En fonction du choix de l'outil administrateur le fichier main.lzx sera compilé en un fichier swf pour créer le module voulu. Par exemple le fichier a_main.swf crée une salle avec trois modules, le chat, la vidéo et le tableau blanc (voir figure suivante).

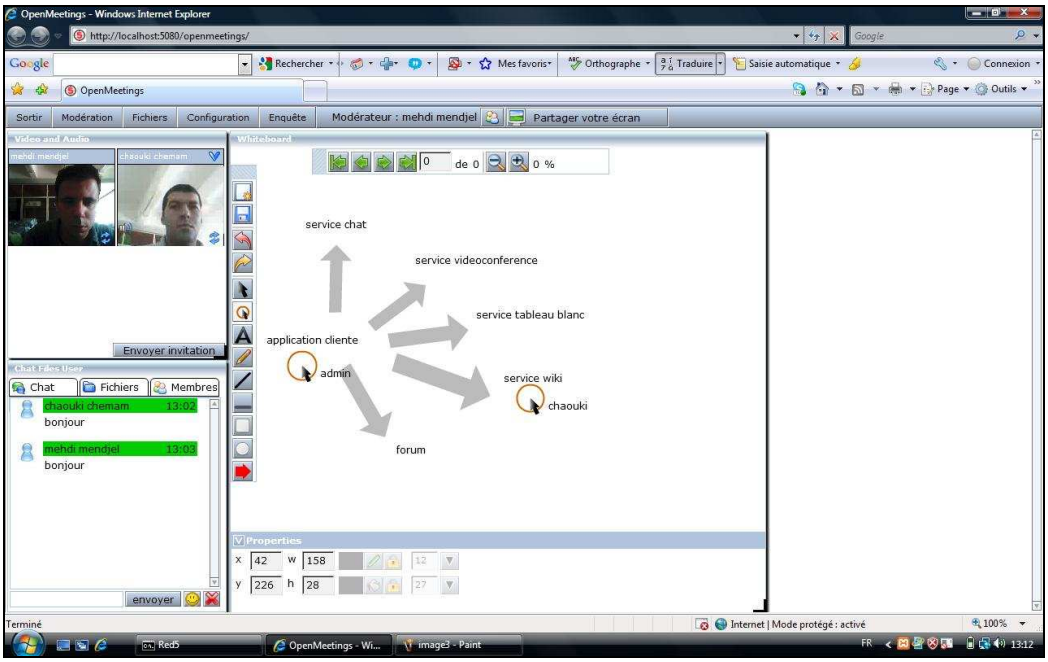

Figure 6.18 : La génération du fichier a_main.swf

Cette interface montre que les participants peuvent utiliser plusieurs services dans une même salle. Ils peuvent utiliser le chat pour échanger des messages, ils peuvent utiliser le tableau blanc et le bureau partagé avec toutes leurs fonctionnalités (des capacités de dessin, d'écriture et d'édition, glisser/déposer, l'enregistrement du travail entre deux sessions, etc.), et ils peuvent aussi utiliser le module de la vidéoconférence.

Par contre le fichier b_main.swf crée une salle avec quatre modules, le chat, la vidéo, le tableau blanc et compendium.

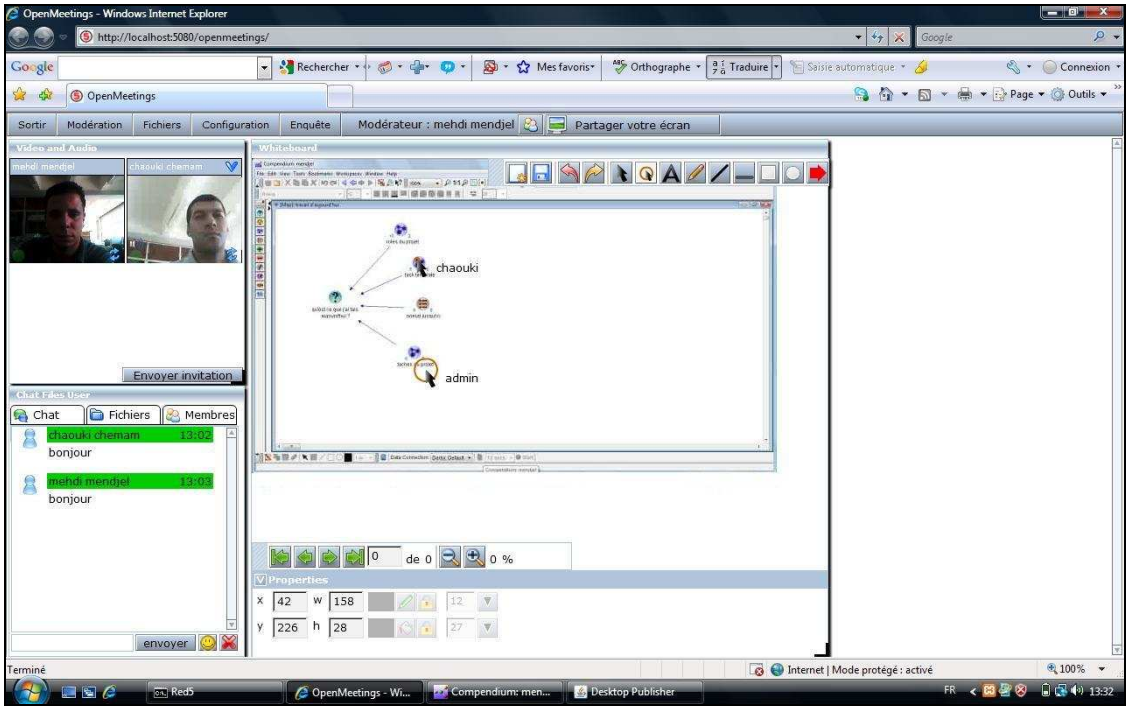

Figure 6.19 : Salle avec l'outil compendium

Ici les utilisateurs peuvent communiquer et collaborer ensemble en utilisant les différents outils de collaboration disponibles, ils peuvent aussi partager leurs idées en utilisant les outils offerts par compendium.

6.8 Conclusion

Nous avons vu dans ce chapitre une expérimentation sur l'utilisation et le déploiement de nos deux plates-formes. La première qui s'appelle Openmeetings, elle offre aux utilisateurs les différents outils de collaboration synchrones. Pour exploiter ces outils nous avons choisi d'utiliser une deuxième plate-forme qui s'appelle compendium et qui permet la structuration et la modélisation des idées des participants.

Nous avons opté pour une intégration de compendium dans le but de :

- Montrer la puissance des services web surtout au niveau de l'hétérogénéité et de la réactivité, car on a pu faire communiquer deux service web conçus avec des langages de programmation et dans des environnements totalement différents.
- Exploiter l'approche faiblement couplée des services web.
- Montrer l'extensibilité de la plate-forme, cela se fait par l'intégration de n'importe quel autre environnement basé sur des services web.

Conclusion Générale & perspectives

7.1 Conclusion

Les travaux de recherche développés dans ce mémoire s'inscrivent dans le domaine des plates-formes collaboratives intégrant les différents outils de collaboration (synchrones et asynchrones). Plus précisément, nous avons proposé une approche générale pour créer un environnement collaboratif qui se base sur les différents outils de collaborations synchrones s'appuyant sur une intégration d'applications collaboratives (ou collecticiels) existantes.

Dans la première partie de ce mémoire nous avons présenté un bref historique sur l'évolution des plates-formes collaboratives surtout avec l'apparition du web, nous avons défini les principaux concepts qui font parti du travail collaboratif avec les différentes architectures existantes. Nous avons consacré un deuxième chapitre pour parler des différents outils de collaborations, leurs domaines d'application, en citant les différentes approches de développement avec leurs avantages et limitations. Ceci pour pouvoir positionner notre approche, qui se base sur une architecture orientée services basée sur l'intégration des plates-formes dans le but d'accomplir une tache collaborative dans un domaine bien précis.

Comme notre approche est basée sur les architectures orientées services, nous avons expliqué en détail le concept général de cette architecture qui constitue la prochaine vague de développement et qui comprend des propriétés spécifiques, des composants et des interconnexions qui mettent l'accent sur l'interfonctionnement et la transparence d'emplacement.

Dans la deuxième partie nous avons proposé une approche générale pour concevoir un environnement collaboratif simple et extensible. Cette approche se base sur l'intégration d'applications existantes en utilisant le principe des services web qui est une nouvelle technologie permettant l'interopérabilité et l'extensibilité des applications.

Dans la pratique ceci nous a conduit à définir deux applications existantes, une application qui offre des outils de collaboration synchrones en les exposant comme étant des services web, et une deuxième application qui se base sur le concept de la gestion des connaissances qui permet de faciliter la capture et la structuration des idées.

Donc notre but était de proposer un environnement collaboratif qui offre un outil de vidéoconférence, un tableau blanc un outil de chat, un bureau partagé et enfin un outil qui permet

de structurer les idées des participants, modéliser leurs discours et leur domaines de problèmes d'une manière synchrone et asynchrone.

7.2 Limitations et perspectives

Des limitations peuvent être observées dans notre approche, parmi les quelles on peut citer :

- L'approche est basée sur l'intégration des applications par le billet des services web, donc l'implémentation va se baser sur les propriétés des deux plates-formes.
- Le nombre de participants à une conférence est limité à quatre.
- La plate-forme n'a pas été testée sur un grand réseau et avec un nombre important de participants, où d'autres problèmes peuvent surgir.

Les améliorations que nous proposons sont les suivantes :

- Notre premier objectif consiste à compléter l'implémentation de notre approche surtout par l'intégration d'autres outils de collaborations asynchrone pour rendre notre environnement plus complet.
- Notre deuxième objectif consiste à poursuivre nos travaux pour une mise en œuvre complète d'un environnement collaboratif qui ne se base pas uniquement sur l'intégration des plates-formes basées sur des services web, mais peut aussi intégrer d'autres plates-formes client serveur. Cela se fait par l'implémentation d'un module constitué d'un ensemble de services qui permet d'intégrer un maximum de plates-formes et d'outils de collaboration en les considérant comme étant des boites noires (c'est-à-dire sans modifier leurs codes internes).
- Se pencher vers une hybridation entre les services web, les grilles de calcul et les SMA (système multi agents), pour pouvoir développer un environnement collaboratif simple, extensible, réactif et puissant.

Références bibliographiques

[1] Philippe d'Anfray, Frédéric Desprez et Raphäel Bolze le projet Décrypthon, http://2007.jres.org/planning/pdf/102.pdf

[2] Atul Khanna John Zinky, The revised ARPANET Routing Metric, BBN Communication Corporation, http://www.cs.princeton.edu/~jrex/teaching/spring2005/reading/khanna89.pdf

[3] JAVA (langage) Wikipedia, http://fr.wikipedia.org/wiki/Java_ (langage).

[4] Sylvain RAMPACEK, Sémantique, interactions et langages de description des services web complexes, thèse de doctorat 2006, http://iutdijon.u-bourgogne.fr/iq/srampacek/fichiers/These-Sylvain-Rampacek-20061127.pdf

[5] Origine des norms SGML, HTML et XML, Standard Generalized Markup, Language, http://www.jalix.org/ressources/miscellaneous/infodoc/_Loria-Evry/04.xml.pdf

[6] C. Ellis, S. Gibbs ET G. Rein, "Groupware, Some issues and experiences", Communications of the ACM, Vol. 34, No. 1, Janvier 1991.

[7] Graham, T.C.N., Grundy, J. "External Requirements of Groupware Development Tools" Engineering for Human-Computer Interaction, Kluwer Academic Press , Heraklion (Grèce), 1999

[8] Alan Dix, "the lattice of value designing products for self-growth", http://www.hiraeth.com/alan/ebulletin/lattice-of-value/lattice-of-value.pdf

[9] J.Demongeot, P. Le Beux et G.Weil, Travail cooperatif et groupware, http://www.spim.jussieu.fr/doc/pdg/InfoSante/7-14.pdf

[10] K.Schmidt, T.Rodden, "Putting it all together: Requirements for a CSCW platform",pp 157-176 Amsterdam, http://www.itu.dk/~schmidt/papers/cscw_platform.pdf

[11] Martine Chomienne, "la vidéoconférence: un outil pédagogique à exploiter", publication octobre 2007.
http://www.profweb.qc.ca/fileadmin/user_upload/Dossiers/Dossier_Visioconference/Visio_MartineChomienne_11oct07.pdf

[12] Chunting Huang, University of Texas at Austin, "Videoconferencing", http://www.edb.utexas.edu/multimedia/PDFfolder/Videoconferencing.pdf

[13] Glenn Broadhead Illinois Institute of Technology," Using CU-SeeMe for Online Training", http://www.stc.org/edu/48thConf/files/stc_gjb.pdf

[14] Bernard MATAGNE et Sonia SEHILI,"la vidéoconférence en classe",http://www.rtsq.qc.ca/communiquer/videoconference_en_classe.pdf

[15] Thierry Turletti, The INRIA Videoconferencing System (IVS), octobre 1994.

[16] International telecommunication union,"video codec for audiovisual services", http://www.h261.com/doc/h.261.pdf

[17] Poellhuber, Bruno & Chomienne, Martine (2007). Cégeps en réseau : un projet de télécollaboration et d'innovation pédagogique pour revitaliser les programmes techniques à petites cohortes.

[18] JM.Kaçmann Red Hat France, "Présentation Red Hat linux et le logiciel libre", http://www.ntsys.fr/seminaire/seminaire20040330/download/Presentation_RedHat jmk.sxi.pdf

[19] le site officiel de la plate-forme moodle, http://moodle.org

[20] Rapport final évaluation de sakai, juillet 2006,
http://www.sakaiquebec.org/en/documents/Rapport-evaluation-Sakai.pdf

[21] Regis ADAM, "mise en place d'un serveur Open-Xchange",
http://redg.blogdns.org/Ox/OX.pdf

[22] Guillermo de Jesús Hoyos Rivera et All, " Colab co-navigation sur le web".
http://www.inf.ufes.br/~rgomes/articles/colab-final.pdf

[23] John Hunt, JayDee Technology Ltd,"Smalltalk and object orientation",
http://stephane.ducasse.free.fr/FreeBooks/STandOO/Smalltalk-and-OO.pdf

[24] Matthieu Gallien et All Université de Toulouse,"d'une approche modulaire a une approche orintée composant pour le développement de systèmes autonomes : Défis et principes"
http://www.laas.fr/~felix/publis-pdf/car08.pdf

[25] W3C working group note 2004,"Web Services Architecture",
http://www.w3.org/TR/ws-arch/#id2260892

[26] Fabrice Rossi, Université Pari IX, "Introduction a XML".
http://www.librecours.org/documents/2/238.pdf

[27] Eric van der Vlist, "Le tryptique SOAP/WSDL/UDDI ", Web service convention, juin 2004.
http://dyomedea.com/papers/2004-wsc/3-soap.pdf

[28] Mohamed Karami, "Déploiement et Appel aux web services java avec Axis".
http://superdown.sourceforge.net/Pages/wsaxis/wsaxis.pdf

[29] Yasser shohoud, "Introduction to WSDL".

http://w2ks.dei.isep.ipp.pt/labdotnet/recursos/wsdl.pdf

[30] Joshua Duhi Novembre 2003,"Rich Internet Application", white paper.

http://www.adobe.com/platform/whitepapers/idc_impact_of_rias.pdf

[31] Sellay bdelKader fevrier 2007, "XAML".

http://moncvplus.site.voila.fr/Zammel.pdf

[32] Maxime Alexandre Aout 2007, "Introduction au XUL à travers le framework Mozilla XPFE".
ftp://ftp-developpez.com/m-alexandre/articles/xul/presentation/mozilla.pdf

[33] Nick Velloff, "Using Flash and Flex together".

http://www.velloff.com/Flex_Component_Kit_Examples/FinalPresentation.pdf

[34] http://sites.google.com/a/g-e-t.fr/tech-day/ria

[35] OpenLaszlo An Open Architecture Framework For Advanced Ajax Applications, Novembre 2006.
http://www.openlaszlo.org/whitepaper/LaszloWhitePaper.pdf

[36] Le guide d'OpenLaszlo ,Chapitre 1: Architecture d'OpenLaszlo.

http://svn.openlaszlo.org/openlaszlo/trunk/docs/guide_fr/architecture.html

[37] Orchestra Networks 2004, "Gestion des variantes de services dans les architectures SOA "
http://www.orchestranetworks.com/fr/soa/pdf/article_variantessoa.pdf

[38] Geay Sébastien,Repetto pierre,Vicard Sébastien,juin 2005,"Travaux d'études de licence d'informatique, les architecture orientées services".
http://deptinfo.unice.fr/twiki/pub/Linfo/JournauxDeBord/Geay-Repetto-Vicard-Rapport.pdf

[39] http://code.google.com/p/openmeetings/

[40] ARGOTE, L.INGHAM, P 2000, – Knowledge transfer: a basis competitive advantage in firms,
Organizational Behavior and Human Decision Processes, 82(1) p 150-169.

[41] Compendium Institute. http://compendium.open.ac.uk/institute/

[42] le site official de wikipédia.

http://fr.wikipedia.org/wiki/Accueil

Annexe

Glossaire

Ajax (Asynchronous Java Script and XML) désignant une solution informatique libre pour le développement d'applications web. Ajax n'est pas une technologie en elle-même, mais un terme qui évoque l'utilisation conjointe d'un ensemble de technologies libres couramment utilisées sur le web par exemple (XML et HTML) [42].

Apache Apache HTTP Server, souvent appelé Apache, est un logiciel de serveur HTTP produit par l'Apache software fondation. C'est le serveur HTTP le plus populaire du Web. C'est un logiciel libre avec un type spécifique de licence, nommée licence Apache [42].

BSD (Berkeley Software Distribution) c'est une licence libre utilisée pour la distribution de logiciels. Elle permet de réutiliser tout ou une partie du logiciel sans restriction, qu'il soit intégré dans un logiciel libre ou propriétaire [42].

CORBA (Common Object Request Broker Architecture) est une architecture logicielle pour le développement de composants et d'ORB (Object Request Broker).ces composants qui sont assemblés afin de construire des applications complètes, peuvent être écrits dans des langages de programmation distincts, être exécutés dans des processus séparés, voire être déployés sur des machines distinctes [42].

Framework est un ensemble de bibliothèques, d'outils et de conventions permettant le développement d'applications.il fournit suffisamment de briques logicielles et impose suffisamment de rigueur pour pouvoir produire une application aboutie et dont la maintenance est aisée.ces composants sont organisés pour être utilisés en interaction les uns avec les autres [42].

Ghostscript est le nom d'un ensemble d'outils fournissant :

- Un interpréteur pour le langage PostScript, offrant la possibilité d'offrir des fichiers PostScript vers un grand nombre de formats Bitmap, de les afficher ou encore de les imprimer vers des imprimantes ne supportant pas le langage PostScript.
- Un interpréteur pour le format PDF, avec les mêmes fonctionnalités.

- La possibilité de convertir des fichiers PostScript en PDF er vice versa.

GPL (Licence Publique Générale) c'est une licence qui fixe les conditions légales de distribution des logiciels libres du projet GNU [42].

HTML (HyperText Markup Language) Langage de balise servant a la publication de pages web sur internet.les bases de HTML ont été développées dans le but de pouvoir écrire des documents hypertextes liant les diverses ressources d'internet [42].

HTTP (Hypertext Transfer Protocol) le protocole de transfert hypertexte est un protocole de communication client-serveur développé pour le World Wide Web.il est utilisé pour échanger toute sorte de données entre un client http et un serveur http [42].

ImageMagick est un logiciel libre, comprenant une bibliothèque, ainsi qu'un ensemble d'utilitaires en ligne de commande permettant de créer, de convertir, de modifier et d'afficher des images dans un très grand nombre de formats [42].

JAVA Java est à la fois un langage de programmation et une plateforme d'exécution. Le langage Java a la particularité principale d'être portable sur plusieurs systèmes d'exploitation tels que Windows, MacOS ou Linux. C'est la plate-forme qui garantit la portabilité des applications développées en Java [42].

J2EE (Java Entreprise Edition) est spécification pour la technologie Java de Sun plus particulièrement destinée aux applications d'entreprise [42].

JRE (Java Runtime Environment) ou environnement d'exécution java, c'est un ensemble d'outils permettant l'exécution de programmes java sur toutes les plates-formes supportées.
La JRE est constituée d'une JVM (machine virtuelle JAVA) pour interpréter le code java et le convertir en code natif, et d'une bibliothèque standard à partir de laquelle doivent être développés tous les programmes en java [42].

MySQL est un système de gestion de base de données (SGBD) développé dans un souci de performances élevées, ce qui signifie qu'il est d'avantage orienté vers le service de données déjà en place que vers celui de mises a jour fréquentes et sécurisées.il est multi-thread et multi-utilisateurs [42].

ORB (Object Request Broker) est l'ensemble de fonctions (classes Java, bibliothèque, C++, etc…), qui implémentent un bus logiciel par lequel des objets envoient et reçoivent des requêtes et des réponses, de manière transparente et portable : il s'agit de l'activation ou de l'invocation par un objet, et a distance d'une méthode d'un autre objet distant [42].

PDA (Personal Digital Assistant) est un assistant personnel ou un ordinateur de poche basé sur le principe d'une calculatrice évoluée, il sert d'agenda, de carnet d'adresses et de bloc-notes.il est doté d'un clavier, avec des petites touches ou d'écran tactile, associé alors a un stylet [42].

PDF (Portable Document Format) le format de document portable généralement abrégé PDF, est un format de fichier informatique créer par Adobe Systems .c'est un format ouvert dont les spécifications sont publiques et utilisables librement et gratuitement.il est dérivé du format PostScript et contient des données au format XML [42].

P2P (Peer-To-Peer) souvent abrégé "P2P", est un modèle de réseau informatique qui s'oppose strictement au modèle client-serveur. Le partage des fichiers avec ce type de réseau se fait d'égal à égal, c'est-à-dire les différentes machines ont à la fois le rôle de serveur et du client [42].

Red5 est un serveur flash gratuit et libre qui reprend les fonctionnalités de Flash Media Server de Adobe.il permet de faire du streaming audio ou vidéo, partage d'objet distant (remoting), de la synchronisation de données, etc. Contrairement a Flash Media Server 2, les applications coté serveur peuvent êtres écrites en JAVA mais aussi avec d'autres langages de script comme Java script [42].

RPC (Remote Procedure Call) est un protocole permettant de faire des appels de procédures sur un ordinateur distant à l'aide d'un serveur d'application. Ce protocole est utilisé dans le modèle client-serveur et permet de gérer les différents messages entre ces différentes entités [42].

SQL (Structured Query Language) le langage structuré de requêtes est un pseudo-langage informatique (de type requête) standard et normalisé, destiné à interroger ou à manipuler une base de données relationnelle [42].

SSL (Secure Sockets Layer) c'est le protocole de sécurisation des échanges sur internet, développé à l'origine par Netscape.il a été renommé en « Transport Layer Security » par l'IETF (Internet Engineering Task Force) suite au rachat du brevet de Netscape [42].

SWFTools est un ensemble d'outils logiciel permettant de créer et de manipuler des fichiers SWF, ce format de fichiers est utilisé par Adobe Flash (logiciel pour créer des animations) [42].

TCAO (Travail coopératif assiste par ordinateur) c'est le domaine qui représente l'étude des outils et techniques du groupeware ainsi que leurs effets sociaux, psychologiques et organisationnels [42].

TIC (Technologies de l'information et de la communication) regroupent les techniques utilisées dans le traitement et la transmission de l'information, principalement de l'informatique, de l'internet et de la télécommunication [42].

W3C (World Wide Web Consortium) abrégé par le sigle W3C, est un organisme de standardisation a but non lucratif ,fondé en octobre 1994 comme un consortium « association ou collaboration temporaire » chargé de promouvoir la compatibilité des technologies du web telles que XML,HTML,RDF,CSS,SOAP,etc…[42].

XPath est le langage de requêtes élémentaire dans XSLT, il détermine si une règle s'applique Via son attribut, et peut aussi servir à extraire des contenus du document XML transformé par le programme XSLT.
XPath peut être utilisé comme langage de requêtes dans les bases de données XML, souvent en tant que sous-ensemble de X-Query [42].

X-Query est un langage de requêtes permettant non seulement d'extraire des informations d'un document XML, ou d'une collection de document XML, mais également d'effectuer des calculs complexes à partir des informations extraites et de reconstruire de nouveaux documents ou fragments XML [42].

XSL (eXtensible Stylsheet Language) c'est le langage de description de feuilles de style du W3C associé à XML.une feuille de style XSL est un fichier qui décrit comment doivent êtres présentés Les documents XML basés sur une même définition de type de documents (DTD) ou un même schéma [42].

XSLT (eXtensible Stylsheet Language Transformations) défini au sein de la recommandation XSL du W3C comme étant un langage de transformation d'un document XML vers un autre, que ce soit un dialecte XML (XHTML, XSL-FO, HTML, etc...) ou bien un autre type de document [42].

La structure générale du code source