AMSELEM Jonathan
BALI Rami
FAYOLLE Samuel
GALEA Nicolas

Master 1 Informatique
TER

Virtualisation d'orchestration de services

Cahier des charges

4 avril 2008

Encadrant : Phillipe COLLET

SOMMAIRE

1. PRESENTATION GENERALE DU PROBLEME

Face à la complexité grandissante des systèmes logiciels, les chercheurs et ingénieurs continuent à imaginer de nouveaux paradigmes. L'approche basée sur la construction d'architectures orientées Service est l'une des plus prometteuses pour gérer cette complexité. Il est communément reconnu que le concept de « Service » facilite l'intégration des systèmes logiciels en masquant la complexité des technologies sous-jacentes et en fournissant une vue globale du système.

Les Web Services[5] définissent une manière standard d'interagir avec des applications distantes en utilisant les technologies du Web. La complexité augmente encore lorsqu'une tâche est basée sur des Web Services dépendant les uns des autres, une application doit donc définir une coordination de ces services.

WS-BPEL[3] , (Business Process Execution Language for Web Services) est une spécification du groupe Oasis qui définit un modèle de coordination et d'orchestration, c'est-à-dire l'enchaînement automatisé, de web-services entre eux. Ainsi, il est possible de définir entièrement un processus métier qui fait interagir des services issus de systèmes différents. Le moteur d'exécution ActiveBPEL[4] assure l'évaluation et l'orchestration des processus Web.

Parallèlement, les architectures orientées service se rapprochent du monde des composants logiciels. Un [1]exemple est la spécification SCA (Service Component Architecture), qui propose un modèle de programmation pour la construction d'applications à base de composants. SCA intègre les orchestrations WS-BPEL comme des composants qui peuvent être assemblés avec d'autres composants SCA.

Durant les années précédentes, l'équipe Rainbow de l'I3S, au sein de laquelle se déroule notre travail, a développé, en collaboration avec France Télécom R&D, un modèle de composants hiérarchiques et dynamiquement reconfigurables appelé FRACTAL[1]. Un ancien groupe de TER a étendu ce modèle en implémentant une boîte à outils, FRACTAL-WS, qui permet de réaliser des ponts entre nos composants FRACTAL et des Web Services.

Notre objectif est d'améliorer Fractal WS pour y intégrer les orchestrations, en représentant ces orchestrations comme des composants Fractal. Ces orchestrations s'exécutant sur le moteur ActiveBpel, seront virtualisées afin de surveiller l'avancement de leurs exécutions et les interactions avec les différents Web Services impliqués dans l'orchestration.

[1] [1], [2], [3], [4], [5] : voir annexe page 19.

1.1 SUJET : VIRTUALISATION D'ORCHESTRATION DE SERVICES

1.1.1 FINALITÉS

Le but de ce projet est d'étendre la boîte à outils Fractal WS pour effectuer des ponts fiables entre composants logiciels Fractal et orchestrations BPEL s'exécutant sur un serveur Active BPEL. Une orchestration devra être représentée par un ou plusieurs composants Fractal. Son déploiement et son exécution devront être contrôlables depuis des composants Fractal, fournissant ainsi une vue virtuelle (complète ou partielle) sur les orchestrations (et leurs déroulement).

1.2 CONTEXTE

1.2.1 SITUATION DU PROJET ET ETUDES DEJA EFFECTUEES

1.2.1.1 FRACTAL

Le modèle de composants Fractal a été défini par France Télécom R&D et l'INRIA depuis 2001. Il se présente sous la forme d'une spécification et d'implémentations dans différents langages de programmation comme Java, C, C++, SmallTalk ou les langages de la plate-forme .NET. Les principales caractéristiques du modéle Fractal sont motivées par l'objectif de pouvoir construire, déployer et administrer des systèmes complexes tels que des intergiciels ou des systèmes d'exploitation. Le modèle est ainsi basé sur les principes suivants :

- **composants composites** (i.e. composants qui contiennent des sous-composants) pour permettre d'avoir une vue uniforme des applications à différents niveaux d'abstraction.
- **composants partagés** (i.e. sous-composants de plusieurs composites les englobant) pour permettre de modéliser les ressources et leur partage, tout en préservant l'encapsulation des composants.
- **capacités d'introspection** pour permettre d'observer l'exécution d'un système.
- **capacités de (re)configuration** pour permettre de déployer et de configurer dynamiquement un système.

La base du développement Fractal réside dans l'écriture de composants et de liaisons permettant aux composants de communiquer. Un composant Fractal est ainsi une entité d'exécution qui possède une ou plusieurs interfaces. Une *interface* est un point d'accès au composant. Une interface implante un *type d'interface* qui spécifie les opérations supportées par l'interface. Il existe deux catégories d'interfaces : les interfaces *serveurs* (en rouge sur la Figure 1)- qui correspondent aux services fournis par le composant -, et les interfaces *clients* (en vert) qui correspondent aux services requis par le composant.

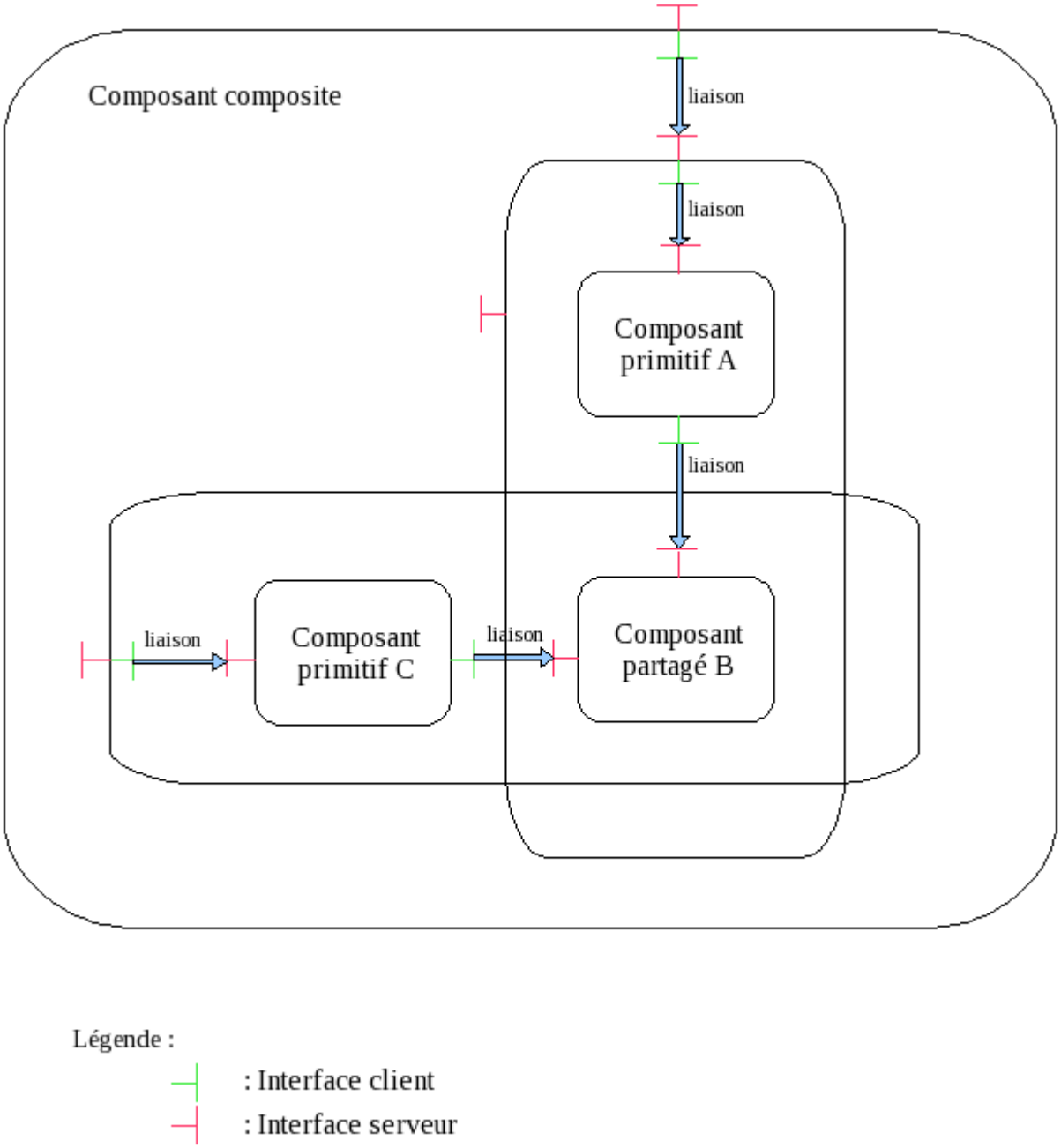


Figure 1

1.2.1.2 FRACTAL-WS

Fractal WS est une boîte à outils qui vise à fournir des moyens pour rendre compatible n'importe quel composant Fractal avec la technologie des Web Service et réciproquement. L'une des interfaces fournies par un composant Fractal peut être transformée en un Web Service, les rendant ainsi accessibles par le biais de protocoles Web. D'autre part, tout Web service (externe) peut être consulté par un composant Fractal, en utilisant un proxy dédié. Des services ou des composants sont générés afin que la passerelle soit opérationnelle. Fractal WS est basé sur l'implémentation de référence “Julia”, en Java , du modèle de composants Fractal.

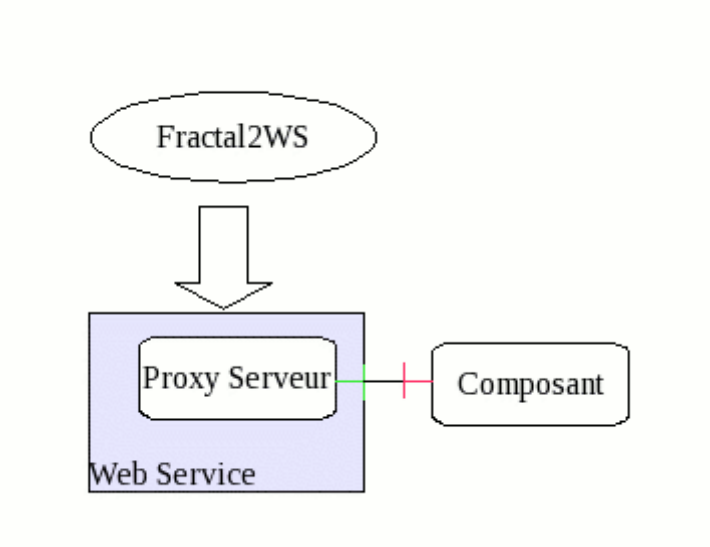


Figure 2 Fractal2WS

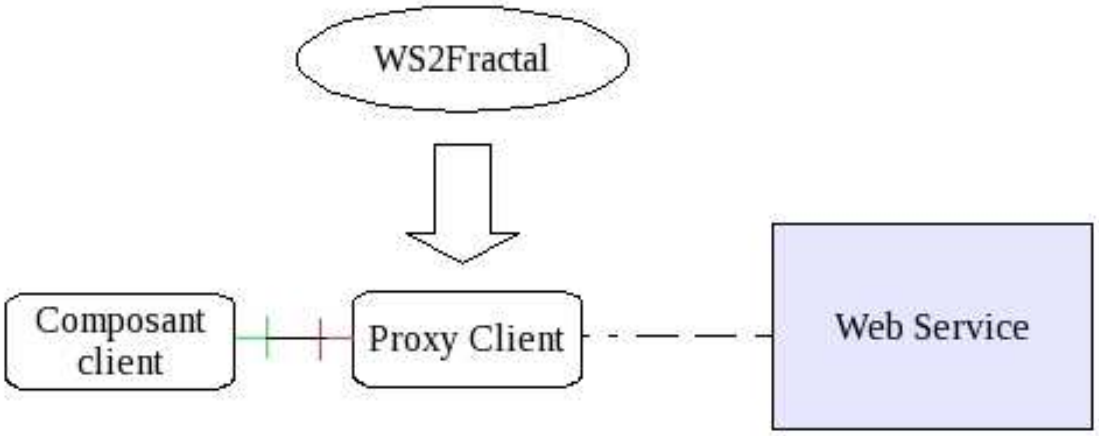


Figure 3 WS2Fractal

1.2.2 ÉTUDES MENEES SUR LES SUJETS VOISINS

IBM de leur coté ont créé un système de composants : Service Component Architecture (SCA), ensemble de spécifications définissant un modèle d'implémentation d'applications dans une architecture orientée service.
L'objectif premier est, selon eux, de faciliter la tâche des développeurs, en leur fournissant une couche d'abstraction, les dédouanant alors de contrainte de langage de programmation et de middleware spécifiques. (voir la partie 1.4).

1.2.3 SUITES PRÉVUES

Le résultat de notre travail sera utilisé dans le cadre d'un contrat de recherche entre le laboratoire I3S et France Telecom R&D notamment pour l'intégrer dans le logiciel de messagerie instantanée AMUI et procéder à la livraison.

Le déploiement ainsi que le packaging est prévu été 2008 lors d'un stage de 3 mois démarrant début juin.

1.2.4 NATURE DES PRESTATIONS DEMANDÉES

Etant donné le nombre important de technologies qui interviennent dans ce projet, toute l'équipe doit passer par une phase d'apprentissage de ces différentes spécifications. Nous avons suivi tous les tutoriels de base pour enfin pouvoir manipuler la plateforme Fractal et le pont Fractal-WS ainsi que ActiveBPEL.

Ensuite, nous devons étudier comment BPEL est intégré dans SCA afin de trouver la meilleure solution possible pour notre virtualisation.

Dans un second temps, nous devrons prendre en main les premiers codes d'essais pour relier le moteur d'orchestration Active BPEL aux composants Fractal. Puis il faudra développer incrémentalement et tester des fonctionnalités de virtualisation des orchestrations. Et enfin, nous intégrerons ces fonctionnalités dans le serveur de communications instantanées AMUI.

1.2.5 PARTIES CONCERNEES PAR LE DEROULEMENT DE PROJET ET SES RESULTATS

Le sujet a été proposé par Philippe Collet. Les personnes concernées par la réalisation du travail sont Amselem Jonathan, Bali Rami, Fayolle Samuel, Galea Nicolas.

Le résultat de notre travail sera intégré par l'équipe du I3S dans le logiciel de messagerie instantanée AMUI puis rendu à France Télécom.

1.2.6 CARACTÈRE CONFIDENTIEL

Les sources, la spécification, ainsi que les documents de conception ne pourront être exposés car ils seront utilisés dans le cadre d'un contrat de recherche entre le laboratoire I3S et France Telecom R&D. Seul le cahier des charges, la documentation, des schémas généraux de l'architecture, le rapport final ainsi que le site du projet seront publics.

1.3 ENVIRONNEMENT DU PRODUIT RECHERCHÉ

L'environnement du produit est consitué des technologies suivantes :

- AXIS, v1.4

- JAVA, JDK 1.6
- Fractal, implémentation Julia 2.5
- FWS, v1.0
- ActiveBPEL, v2.0

Le schéma suivant nous montre plus précisément comment se présentent nos interactions, et ce que nous avons à faire dans ce TER.

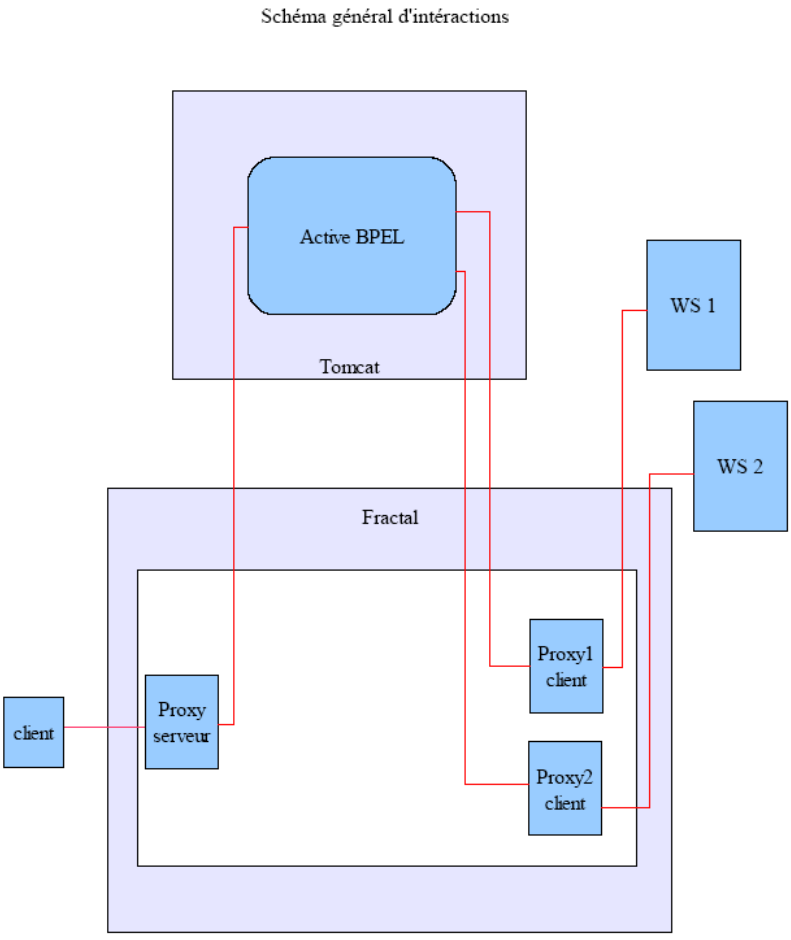


Figure 4

1.4 ETUDE PRELEMINAIRE DE BPEL ET SCA

1.4.1 BPEL

Le langage BPEL4WS, (Business Process Execution Language for Web Services) ou tout simplement BPEL, est une spécification d'IBM, Microsoft, et BEA. Elle remplace les précédentes spécifications XLANG de Microsoft, et WSFL (Web Services Flow Language) d'IBM.

Le modèle de procédé BPEL forme une couche au-dessus de WSDL(Web Service Definition Language). Il définit la coordination des interactions entre l'instance de procédé et ses Web Services partenaires. Les procédés dans BPEL exportent et importent les fonctionnalités en utilisant des interfaces de services web uniquement.

BPEL permet de modéliser des procédés exécutables : chacun spécifie l'ordre d'exécution des activités constituant le procédé, des partenaires impliqués dans le procédé, des messages échangés entre ces partenaires, et le traitement de fautes et d'exceptions spécifiant le comportement dans les cas d'erreurs ou d'exceptions. Le grand avantage de BPEL est la possibilité de décrire les interactions entre les logiques métiers des différentes entreprises à travers les services web. Les éléments du procédé BPEL sont : les liens de partenaires, les activités et les données.

1.4.1.1 LES LIENS DE PARTENAIRES

Un lien de partenaire (partnerLink) correspond au service avec lequel le procédé échange des informations. Le lien de partenaire représente la relation de conversation entre deux procédés partenaires. Chaque lien de partenaire est typé par un partnerLinkType, il est chargé de définir le rôle que joue chacun des deux partenaires dans une conversation.

1.4.1.2 LES ACTIVITES

Le procédé dans BPEL est constitué d'activités liées par un flot de contrôle. Ces activités peuvent être basiques ou structurées.

Les activités basiques sont :

L'invocation d'une opération dans un service web, l'attente d'un message d'une source externe, la réponse à une source externe, l'attente un certain temps, la copie des données d'une place à l'autre, le lancement d'une erreur d'exécution. La terminaison de l'instance de service en entier.

Les activités structurées sont composées d'autres activités basiques et structurées. Les types d'activités structurées sont : la définition d'un ordre d'exécution, l'acheminement conditionnel, les boucles, l'attente d'arrivée d'événements, l'acheminement parallèle, le regroupement des activités afin qu'elles soient traitées par le même gestionnaire d'erreur et l'invocation des activités de compensation par le gestionnaire d'erreur, pour défaire l'exécution déjà complétée d'un regroupement d'activité.

1.4.1.3 LES DONNEES

Le procédé dans BPEL a un état, cet état est maintenu par des variables contenant des données. Ces données sont combinées afin de contrôler le comportement du procédé. Elles sont utilisées dans les expressions et les opérations d'affectation. Les expressions permettent d'ajouter des conditions de transition ou de jointure au flot de contrôle. L'affectation permet de mettre à jour l'état du procédé, en copiant les données d'une variable à une autre ou en introduisant de nouvelles données en utilisant les expressions.

Dans BPEL il n'y a pas de flot de données, BPEL se sert des variables pour passer une donnée d'une activité à une autre, à l'aide de l'affectation (contenant les données).

- Process : regroupe une série d'activités de partenaires et d'éléments d'extensibilité.
- Partners : ils sont définis comme un ensemble de liens de partenaires (partnerLink).

Active BPEL est le moteur d'exécution des orchestrations . Il est considéré comme un Web service dans un serveur Web Tomcat.

1.4.2 SCA.

SCA (SERVICE COMPONENT ARCHITECTURE) EST UN ENSEMBLE DE SPECIFICATIONS VISANT A SIMPLIFIER LA CREATION ET LA COMPOSITION DE SERVICES, INDEPENDAMMENT DE LEUR IMPLEMENTATION, DANS LE CADRE D'ARCHITECTURES ORIENTEES SERVICE (SOA).

SCA est donc un **modèle de programmation pour la construction d'applications à base de composants suivants le paradigme SOA**. Ce modèle se base notamment sur l'idée qu'un service de niveau N se construit par assemblage / agrégation / orchestration de services de niveau N-1 ou N (et ce quelque soit la hiérarchisation choisie. Par exemple : Services organisationnels, Services métiers, Services, techniques).

A ce titre, SCA fournit deux niveaux de modèle :

- **Un modèle d'implémentation** : Construire des composants qui fournissent et consomment des services ;
- **Un modèle d'assemblage** : Construire une application métier à forte valeur ajoutée en liant entre eux un ensemble de composants.

Ainsi, SCA insiste sur une séparation forte entre l'implémentation des services et leur assemblage. Le modèle SCA se veut agnostique vis-à-vis :

- Des technologies d'implémentation des composants de service. Il inclut entre autre les technologies d'implémentation suivantes : Java, C++, BPEL, XQuery ou SQL.
- Des technologies d'exposition et d'invocation des composants de services (même si WSDL et les interfaces java sont mis en avant).

SCA permet de **décrire des services et leur assemblage indépendamment de toutes considérations techniques d'implémentation**.

1.4.2.1 COMPOSANTS ET SERVICES : LE MODELE D'IMPLEMENTATION

L'élément de base de SCA est le **composant** qui constitue l'unité élémentaire de construction.

Un composant est une instance configurée d'implémentation où une implémentation est un code source fournissant des fonctionnalités.

Ces fonctionnalités sont exposées en tant que **services** en vue de leur utilisation par d'autre composant. Les services sont décrits au travers d'interfaces qui constituent le contrat de service. Ces contrats sont implémentés par le composant.

Les paramètres des services sont des structures de données pour lesquelles SCA recommande fortement l'utilisation de la spécification SDO (Service Data Objects) dont il est également l'instigateur.

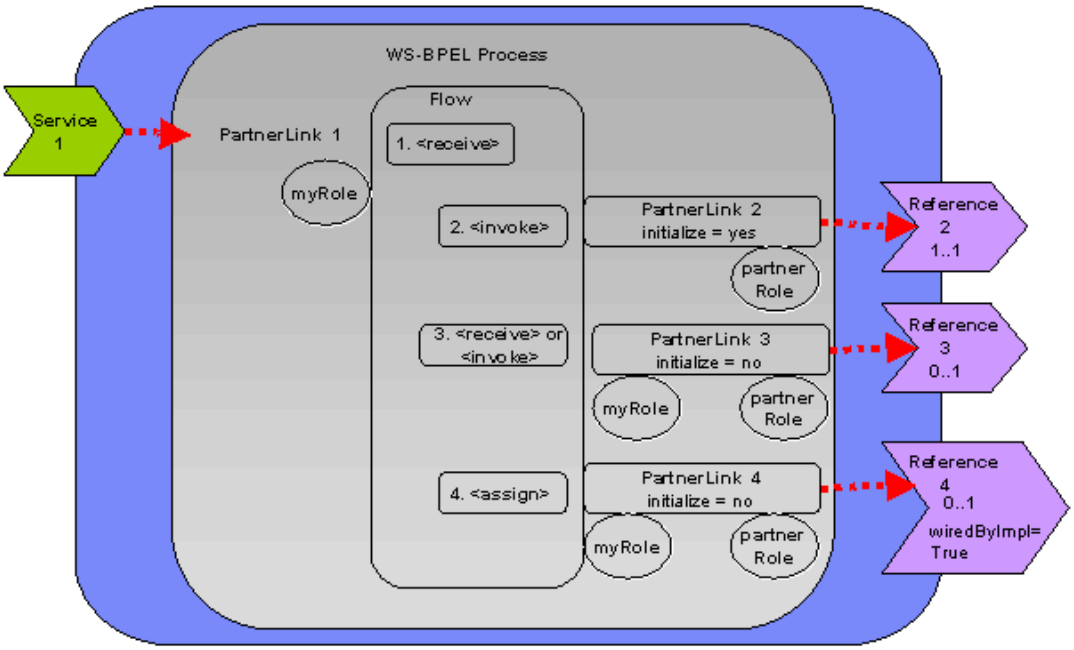


Figure 5

L'**implémentation** peut s'appuyer sur des services fournis par d'autres composants dont elle dépend. Ces dépendances sont appelées **références**. Elles sont associées à des services qui peuvent être soit exposés par d'autre composants SCA, soit exposés par des systèmes tiers. L'implémentation peut être paramétrable au travers de **propriétés** qui influencent le comportement d'une fonctionnalité. C'est le composant qui configure l'implémentation en fournissant des valeurs à ses propriétés et en liant les références aux services fournis par d'autres composants.
Le système de références associé à celui des interfaces permet de réaliser un couplage lâche (ou faible) entre les composants : Un composant consommateur de services ne connaît des composants

fournisseurs de services sur lesquelles il s'appuie que les interfaces (contrat de service) des services qu'il consomme.

1.4.2.2 COMPOSITION ET DOMAINES : LE MODELE D'ASSEMBLAGE

Le deuxième élément définit par SCA est le **composé** qui est un assemblage de composants, services, références, propriétés et des liens qui existent entre ces éléments.

Un composé n'est donc rien d'autre qu'un composant de plus haut niveau que ceux qui le compose (Il fournit des services, dépends de références et a des propriétés). Un composé peut donc à son tour être référencé par d'autres composants et utilisé au sein d'autres composés.

L'utilisation première du composé peut être "détournée" pour regrouper un ensemble d'éléments non nécessairement liés mais qui constitue un ensemble fonctionnel cohérent.

Au plus haut niveau, les composés sont déployés dans des **domaines** SCA qui regroupe l'ensemble des services pour un système fonctionnel.

2 FONCTIONNALITÉS

Nous allons représenter les orchestrations BPEL s'exécutant sur un serveur ActiveBPEL par des composants Fractal. Afin de surveiller les interactions de l'orchestration avec les différents Web-Services impliqués, nous allons :

- Créer des proxy client, à l'aide de Fractal2WS, qui vont envoyer les requêtes sur les orchestrations vers le moteur ActiveBPEL.
- Rediriger les interactions entre le moteur d'exécution et les Web Services requis vers des proxy serveur, créés à l'aide de WS2Fractal, représentant ces Web-Services comme des composants Fractal.
- Permettre de choisir quelles liaisons avec des Web Services nous voulons surveiller et donc quelles liaisons nous allons représenter par des proxy serveur.

Il existe plusieurs niveaux de représentation possible pour les orchestrations :

- Un composant par fichier BPEL et donc par type d'orchestration .(Figure 6)
- Un composant par exécution de l'orchestration BPEL. (Figure 7)

Enfin, nous allons nous appuyer sur les fonctionnalités d'ActiveBPEL pour consulter si possible l'état de l'exécution de l'orchestration.

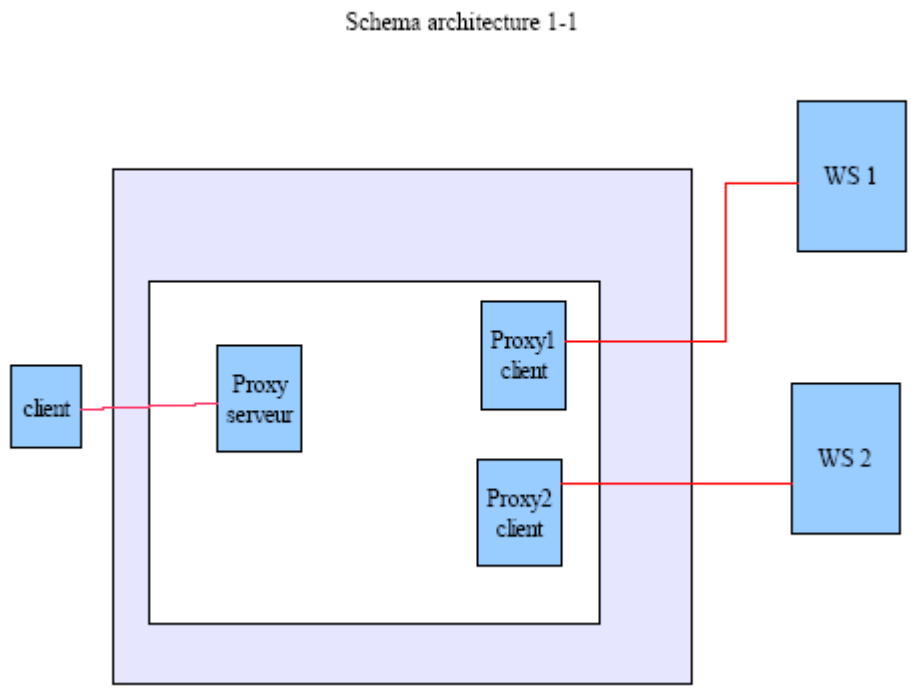


Figure 6

Schema architecture 1-N

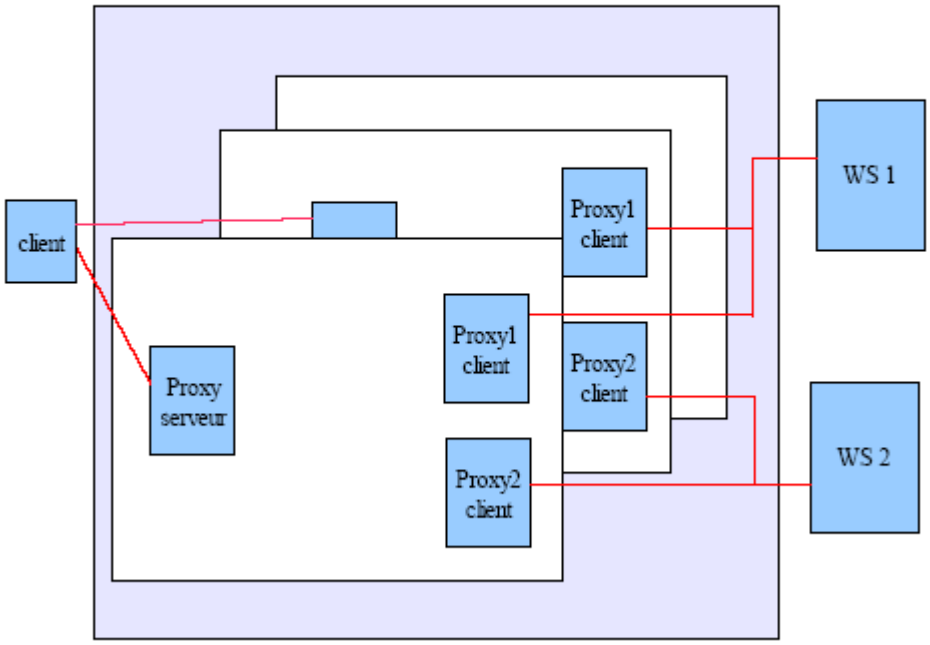
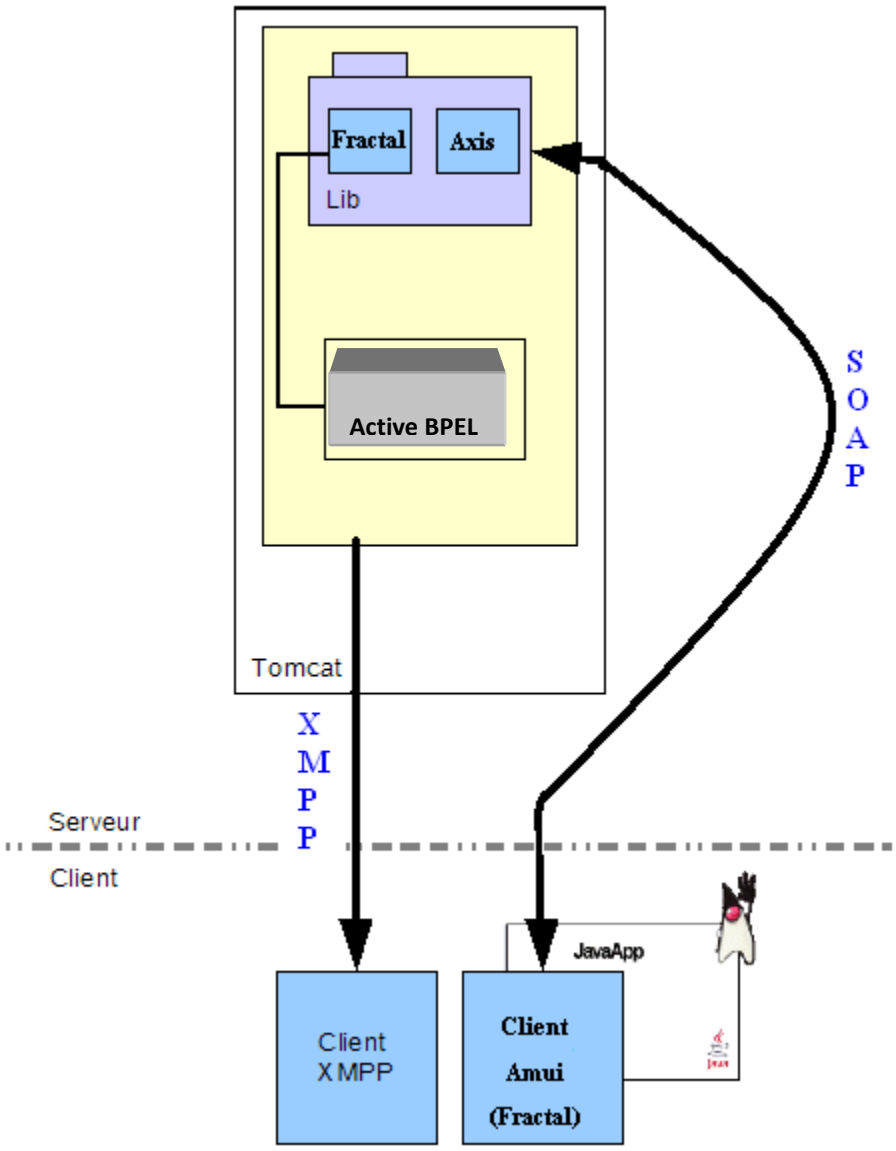


Figure 7

3 CONTRAINTES NON FONCTIONNELLES

Du fait du grand nombre de technologies utilisées, les contraintes liées aux plates-formes matérielles sont importantes.

- Linux
- Sun JDK 6
- Tomcat 5.x
- ActiveBPEL 5.0



4 GESTION DU PROJET

4.1 PRIORITÉS

En fonction de l'étude du sujet et des différents éléments mis à notre disposition, nous avons pu définir les priorités suivantes :

Niveau 1 :

- Prendre le code d'expérimentation fourni précédemment et savoir comment le déployer à plus grande échelle et avec nos composants Fractal, afin de stabiliser ce code.
- Choisir entre plusieurs architectures Fractal candidates pour recevoir et orchestrer les services Web, à savoir :
    - -1 seul composant qui génère lui-même les composants pour chaque nouvelle demande de services.
    - -N composant correspondant aux demandes de services.
- Paramétrer la représentation des partner-link modélisant l'orchestration par des proxy.

Niveau 2 :

- Permettre de consulter l'état des orchestrations en cours d'exécution.

Niveau 3 :

- Intégration et validation du code créé dans le logiciel de communications instantanées AMUI.

4.2 LIMITES ET INTERFACES

Le présent projet est supposé réaliser des ponts entre orchestration de services et composants Fractal nous devrons donc utiliser un serveur ActiveBPEL.

4.2.1 LIMITES

- Nous n'implémentons pas l'exécution d'une orchestration mais celle-ci est faite par un moteur ActiveBPEL.
- Nous allons permettre de consulter et surveiller l'état d'exécution d'une orchestration mais pas de la contrôler.

4.2.2 INTERFACES

- Le composant Fractal représentant l'orchestration intercepte tous les messages reçus ou émis par le moteur d'orchestration.
- Les proxys permettant aux composants Fractal de communiquer avec des Web Services.
- Nous avons besoin d'un fichier BPEL qui décrit l'orchestration à virtualiser.

4.3 HYPOTHESES, DEPENDANCES, CONTRAINTES

Notre connaissance de l'état d'exécution d'une orchestration est fortement dépendante des possibilités d'introspection que nous offre ActiveBPEL. Si par exemple ActiveBPEL ne nous permet pas de suivre les évènements d'exécution, il ne nous sera pas possible d'introduire un mécanisme évènementiel de notification de l'avancement de l'exécution.

De plus nous devons apprendre, maîtriser et intégrer une quantité importante de technologies (BPEL, Fractal …).

4.4 GESTION DU RISQUE

La maîtrise de l'orchestration de service pourra prendre un temps considérable dû à un grand nombre de technologies annexes à apprendre, en particulier :

- WSDL, BPEL et Fractal.

Il est probable que le code intermédiaire visant à analyser le BPEL ne soit pas juste une redirection des services Web. Il nous faut pour cela savoir ce que nous recevons statiquement d'Active BPEL et connaître les notifications dynamiques.

Nous travaillons à partir d'un code de base fourni, donc il faudra un certain temps pour comprendre et stabiliser ce code.

Étant donné que le nombre de participants est limité et le temps fixé, il faudra se focaliser sur les tâches principales.

4.5 MOYENS DE CONTROLE

Pour réaliser dans les délais les objectifs fixés, nous devons mettre en place des moyens de contrôle :

- Faire valider par l'encadrant chaque étape de la réalisation.
- Intégration continue au lieu du développement incrémental : vérifier au fur et à mesure de l'avancement du travail à l'aide de tests.
- Vérifier que tout fonctionne bien à l'aide de la console d'administration.

4.6 METHODES ET OUTILS EMPLOYES

Pour la conception et documentation (privée/publique) : Java doc, Open Office, Word 2007.

Le suivi se fera au travers de réunions, de pages de suivi, et de mails.

Pour le développement nous utiliserons :

- IDE Eclipse
- JUnit pour les tests
- Outils associés aux technologies Fractal et Web services
- Subversion (en abrégé SVN) qui est un système de gestion de versions.

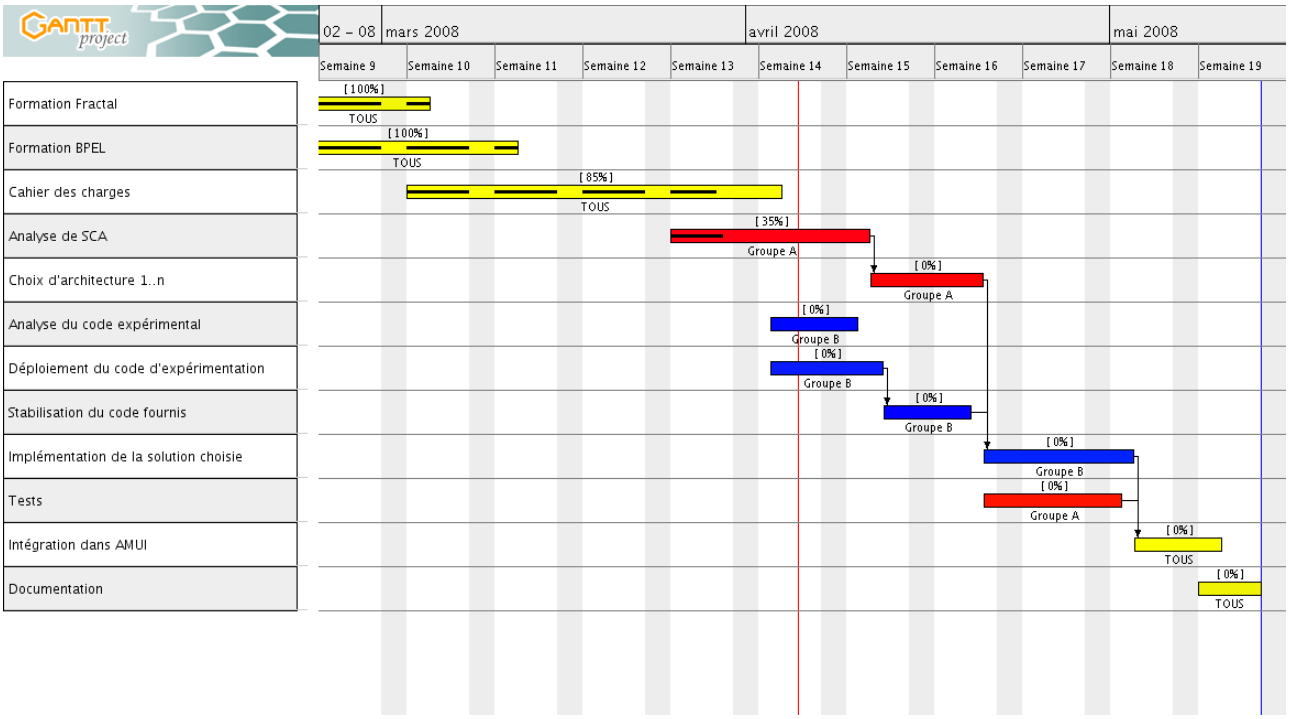
Pour la gestion de projet, nous utiliserons l'outil Gantt Project pour dessiner nos plannings.

4.7 PLANNING

Vu le grand nombre de technologies utilisées, nous avons commencé ensemble par les étudier en lisant les spécifications fournies par les éditeurs et en s'exerçant avec les tutoriaux disponibles, puis débuté l'écriture de ce rapport.
Ensuite nous nous somme séparés en deux groupes, A et B :

- Le premier groupe s'est chargé de l'étude de la manière dont BPEL est intégré dans SCA et d'une solution pour représenter dans Fractal chaque exécution d'une orchestration par un composant.
- Parallèlement, le second groupe analyse, le déploiement et la stabilisation du code fourni par Annabelle MERCIER, qui a fait les premières expérimentations.

A la fin de cette seconde phase, nous devons pouvoir représenter une orchestration par un seul composant Fractal composite contenant les proxys nécessaires.
Ensuite, l'équipe B devra implémenter la solution choisie au problème des composants par exécution d'une orchestration. L'équipe A procèdera en même temps à l'élaboration de tests pour cette partie.
Enfin , nous réaliserons ensemble l'intégration dans AMUI et la rédaction du rapport final.



5 ANNEXES :

5.1 DÉFINITIONS :

[1]Composant Fractal :

Le modèle de composants Fractal est un modèle général dédié à la construction, au déploiement et à l'administration (e.g. observation, contrôle, reconfiguration dynamique) de systèmes logiciels complexes, tels les intergiciels ou les systèmes d'exploitation.

[2]Orchestration :

Il s'agit de la partie qui décrit les règles de distribution des différentes tâches à exécuter aux différents Web Services. Autrement dit, le compositeur en train d'« orchestrer » distribue consciemment ses instruction selon les services qu'il souhaite obtenir.

[3]BPEL :

BPEL et une spécification visant à créer une infrastructure d'orchestration pour services Web.

[4]ActiveBPEL :

L'outil de conception Active BPEL open source et le premier environnement de développement intégré pour le déploiement, la construction, et les testes des applications basée sur la spécification BPEL.

[5]Web Services :

Un Service Web est un programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités exposées sur Internet ou sur un Intranet, par et pour des applications ou machines, sans intervention humaine, et en temps réel.

- **[6]WSDL : (**Web Services Description Language)

  Il s'agit d'une normalisation regroupant la description des éléments permettant de mettre en place l'accès à un service réseau.

  Le **WSDL** décrit une Interface publique d'accès à un service Web. C'est une description basée sur le XML qui indique « comment communiquer pour utiliser le service »; le Protocole de communication, et le format de messages requis pour communiquer avec ce service.

- **[7]SOAP : (**Simple Object Access Protocol**)**

  Il s'agit d'un protocole permettant la transmission de messages entre objets distants, ce qui veut dire qu'il autorise un objet à invoquer des méthodes d'objets physiquement situés sur un autre serveur.

6 BIBLIOGRAPHIE :

«Le système de composants Fractal» , *Intergiciel et Construction d'Applications Réparties,* T. Coupaye, V.Quéma, L. Seinturier, J.-B. Stefani, licence Creative commons , 2006

1 - « Universal Description Discovery and Integration » , http://uddi.xml.org/

2 - « **SOAP » ,** http://www.w3.org/2002/07/soap-translation/soap12-part0.html

3 - « **Web Services Description Language » , http://www.w3.org/TR/wsdl**

4 - « **Service Web** » **,** http://ws.apache.org/axis/

5 - « **Fractal Project** » **,** http://fractal.objectweb.org/

6 - « Service Component Architecture Home », http://www.osoa.org/display/Main/Service+Component+Architecture+Home