République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Bejaia
Faculté des Sciences Exactes
Département d'Informatique

ECOLE DOCTORALE RESEAUX ET SYSTEMES DISTRIBUES

Mémoire de Magister

En Informatique

Option : Réseaux et Systèmes Distribués

Thème

Etude des modèles des « GRID Services » et implémentation d'une plateforme de développement/gridification basée sur les « Web services »

Présenté par

MEDJEK Faiza

Devant le jury composé de :

Président	KERKAR Moussa	Professeur	Université de Bejaïa
Examinateur	TARI Kamel	MCA	Université de Bejaïa
Examinateur	BOUFAIDA Mahmoud	Professeur	Université de Constantine
Rapporteur	BADACHE Nadjib	Professeur	Université de l'USTHB
Invitée	ELMAOUHAB Aouaouche	CR	CERIST, Alger

Promotion : 2008/2009

REMERCIEMENTS

Je remercie en premier notre grand Dieu pour m'avoir donné le courage et la volonté pour terminer ce modeste travail.

J'adresse tout d'abord mes remerciements à mes directeurs de recherche, pour avoir accepté de m'encadrer et m'avoir aidé à conduire ce travail jusqu'au bout : Monsieur Nadjib BADACHE, professeur, directeur du CERIST pour m'avoir d'abord proposé le sujet et donné l'occasion de travailler sur une thématique pleine de perspectives.

Je remercie Madame Aouaouche ELMAOUHAB, chargée de recherche au CERIST pour m'avoir orienté, guidé, corrigé mon travail et mis à ma disposition tous les moyens pour la finalité de ce travail, une riche documentation, soutenu et supporté dans des moments délicats, je lui suis profondément reconnaissante.

Je remercie: Mr. Moussa KERKAR, Mr. Mahmoud BOUFAIDA et Mr. Kamel TARI, pour avoir accepté de juger ce modeste travail.

Un grand MERCI à Mes Parents, Mon Mari, Mes sœurs et Mon frère pour m'avoir soutenu à fin que ce travail arrive à sa fin.

Je remercie ma belle-famille pour m'avoir encouragé.

Je remercie mes amis pour leur aide appréciable, leurs encouragements continus et leur soutien moral ininterrompu.

Je voudrais remercier vivement mes collègues du CERIST, pour m'avoir poussé à faire le magister, encouragé, motivé et surtout soutenu et aidé avant et durant la préparation de mon mémoire.

Et pour être sûr de n'oublier personne, que tous ceux, qui de près ou de loin, ont contribué par leurs conseils, leurs encouragements ou leur amitié, à l'aboutissement de ce modeste travail, trouvent ici l'expression de ma profonde reconnaissance.

RÉSUMÉ

Les applications de calcul scientifique qui nécessitent des ressources de calcul et de stockage importantes nécessitent des environnements de grilles de calcul. Ces environnements de grilles se basent sur des approches orientées services. Dans ce contexte, les infrastructures « web services », dans le domaine du calcul scientifique peuvent être utilisées pour le développement et la « gridification[1] » des applications.

Par ailleurs, les architectures orientées services qui utilisent les infrastructures des *Grid services* permettent les invocations des services à travers les réseaux et les exécutions parallèles selon les workflows[2] qui les définissent. L'orchestration des activités/processus désignés par les workflows est une condition essentielle.

Notre objectif est de résoudre le problème d'interaction (intégration) des différents environnements qui répondent aux besoins de gridification d'applications scientifiques (géo-spatiales, physiques, bioinformatique, etc.) basées sur la notion de workflows et de permettre l'accès transparent aux applications en faisant abstraction des différents langages de programmation. La finalité étant de masquer la complexité à utiliser la grille et de permettre le parallélisme des applications en cas de besoin et ce en exploitant la force des workflows.

Mots clés: Grille de calcul, Services web, Grid services, composition de services web, workflow, BPEL

[1]La gridification est de rendre les applications exécutables sur une grille de calcul.

[2]On appelle "**WorkFlow**" (traduisez littéralement "flux de travail") la modélisation et la gestion informatique de l'ensemble des tâches à accomplir et des différents acteurs impliqués dans la réalisation d'un **processus métier** (Business Process)**.**

ABSTRACT

Scientific computing applications that require powerful computing resources and large storage capacities require grid computing environments. These environments are based on service-oriented approaches. In this context, "web services" infrastructure can be used in scientific computing domain for applications development and "gridification". This approach requires a study of the two environments and their interaction.

Moreover, service-oriented architectures that use the Grid services infrastructure allow invocations of services across networks and parallel executions as workflows that define them. The orchestration of activities/processes covered by the workflow is essential. It is requested, as part of this work to study the development environments that best meet this constraint and validate this work by an implementation of GRID and Web services environments combined.

Our goal is to solve the problem of interaction (integration) of the different environments that respond to the scientific applications (geo-spatial, physical, bioinformatics, etc.) gridification needs based on the workflows concept and enable transparent access to those applications ignoring the different programming languages. The purpose is to hide the complexity to use the grid and allow the applications parallelization when needed and that by harnessing the power of workflow.

Keywords: Grid computing, Web services, grid services, web services composition, workflow, BPEL

ملخّص

تطبيقات الحوسبة العلمية التي تتطلب موارد حوسبة وتخزين كبيرة تتطلب بيئات الشبكات الحاسبة (ڤريد). هذه الشبكات تستند على منهجية الخدمات. في هذا السياق، يمكن استخدام البنية التحتية "خدمات ويب" في مجال الحوسبة العلمية لتطوير البرمجيات و "الڤريديفيكاسيون".

بالإضافة إلى ذلك ، الهندسة المتجهة نحو الخدمات و التي تستخدم البنية التحتية لخدمات ڤريد تسمح دعوات الخدمات عبر الشبكات، والتنفيذيات الموازية التي تحدد لهم استنادا على طريقة سير العمل. تنسيق الأنشطة/العمليات التي تعينها طريقة سير العمل أمر ضروري.

هدفنا هو حل مشكلة دمج و اندماج البيئات المختلفة التي تلبي احتياجات التطبيقات العلمية في شبكات ڤريد استنادا على مفهوم طريقة سير العمل وتمكين الوصول إلى هذه التطبيقات بكل شفافية و دون حسبان لغات البرمجة المختلفة. والغرض من ذلك هو إخفاء التعقيد المتعلّق باستخدام شبكات ڤريد، وتمكين التنفيذيات الموازية للتطبيقات في حالة الضرورة والاستفادة من قوة طريقة سير العمل.

الكلمات الرئيسية : الحوسبة الشبكية، خدمات الويب، خدمات الشبكة ، تركيب خدمات الويب ، طريقة سير العمل ، بي پي إي آل

Table des matières

Table des illustrations

Chapitre1: Les grilles de calcul

Chapitre 2: Les services web

Chapitre 3: Composition de services web & BPEL

Chapitre 4: Modèle de gridification proposé & prototype

Liste des tableaux

INTRODUCTION GÉNÉRALE

1. Introduction

La répartition des applications informatiques fait l'objet d'un fort développement depuis une vingtaine d'années. Cet essor s'est accentué avec les progrès technologiques des réseaux de télécommunication et des réseaux de l'informatique. Plusieurs technologies sont nées pour répondre à différentes contraintes liées à cette répartition mais d'autres types de problèmes dans les cadres académiques et industrielles ont ressurgis.

Au sein des projets scientifiques, les simulations ainsi que les applications prennent en entrée des centaines de giga-octets et génèrent des résultats de l'ordre de plusieurs dizaines de téraoctets. Les chercheurs ont donc besoin d'un instrument pouvant stocker et exploiter une masse d'information considérable. Face à cette demande croissante de puissance, les *grilles de calcul* apparaissent de plus en plus comme *la solution* de demain **[Smo 06]**. En effet, ces architectures permettent d'additionner les ressources matérielles pour former un ensemble offrant une capacité de stockage et de calcul virtuellement infinie.

Les caractéristiques d'une grille de calcul permettent l'accessibilité, la disponibilité, la fiabilité, le partage et la sécurité. Le principe d'externaliser/délocaliser le calcul permet de profiter d'une puissance de calcul exceptionnelle, tout en évitant les contraintes de place, de climatisation et donc d'énergie pour des entités de recherche ce qui garantit un gain énorme en temps et en argent.

D'un côté, les applications scientifiques modernes ont de plus en plus besoin d'être distribuées sur les grilles de calcul, de l'autre côté, il y a un intérêt croissant à utiliser des infrastructures de services web pour les calculs scientifiques parallèles et distribués. Cette technologie de services web émergente qui a été créée à l'origine pour avoir comme préoccupation première l'interopérabilité entre applications via le Web **[Pke 04]** représente un défi informatique mais aussi économique. Elle fait parler d'elle de plus en plus dans le domaine de grille de calcul et du calcul scientifique intensif.

La principale raison de l'adoption d'infrastructures de services web est que le calcul scientifique peut bénéficier de manière significative de l'investissement considérable que l'industrie informatique engage dans des méthodes, outils et middleware en faveur du développement des services web.

Aussi, les technologies des workflows sont en train de devenir l'approche dominante pour coordonner des groupes de services distribués. Elles sont appelées «Orchestration et Chorégraphie» **[Hsa 05]**, cette notion permet la modélisation des applications scientifiques.

2. Problématique

Les applications scientifiques exécutées avec les technologies distribuées modernes ont tendance à être calculées sur des ressources diverses et dispersées. Ces mêmes applications complexes peuvent être modifiées au niveau de leur structure par l'ajout, la suppression ou la modification d'une ou de plusieurs fonctionnalités ou procédures qui peuvent être exécutées en séquentiel ou en parallèle ce qui laisse les workflows devenir une méthode naturelle de les décrire. Cette méthode fonctionnelle de décomposition d'application permet de concevoir manuellement et entièrement qu'un certain niveau de complexité.

Notre problématique repose sur les points clés suivants :

1. Comment permettre l'interaction (intégration) des différents environnements qui répondent aux besoins de gridification d'applications scientifiques (géo-spatiales, physiques, bioinformatique, …) basée sur la notion de workflow?

2. Comment accéder aux applications de manière transparente tout en faisant abstraction des différents langages de programmation.

Les projets dans ce domaine sans divers mais chacun traite une application particulière, dans un contexte particulier. L'un des objectifs est de proposer un modèle de gridification dans lequel un grand nombre d'application peut bénéficier.

3. Contribution

Le travail présenté dans ce document s'inscrit dans le cadre du NGI « Nationnal Grid Initiative » qui vise à développer une infrastructure de grille de calcul nationale nommée DZ-grid qui rentre dans le cadre du projet régional Eumedgrid[3]. Ce travail est réalisé au sein de l'équipe de la Technologie des Systèmes Collaboratifs et Intégrés de la division Réseaux du Centre de Recherche sur l'information Scientifique et Technique. Il traite le développement et la gridification de workflow d'application scientifique.

Notre objectif est de masquer la complexité à utiliser une grille et de permettre le parallélisme des applications en cas de besoins, et ce par la gridification des applications

[3] Eumedgrid est une initiative qui vise à développer une infrastructure de grid pour la Recherche dans le bassin Méditerranéen.

scientifiques interfacées via des services web, composés à l'aide du langage BPEL[4] et s'exécutant dans un moteur d'orchestration tout en exploitant la puissance des workflows. Pour ce faire, nous avons fait une étude sur les environnements de grille de calcul, les technologies de services web et leur composition. Nous avons vu un ensemble de travaux relatifs à notre problématique. En fin nous sommes arrivés à proposer un modèle de gridification avec intégration des environnements adéquats, notamment, le middleware, le conteneur de services de la grille « *grid services* » et le moteur d'orchestration. Nous avons par ailleurs, modélisé, déployé et exécuté des *grid services* et un workflow à base de ces *grid services*.

Dans tout le mémoire le terme « *grid services* » représente la traduction de *« services de la grille »*

4. Organisation du document

Ce mémoire commence par une introduction générale qui sera suivie de deux parties présentant respectivement l'état de l'art et notre contribution et se termine avec une conclusion générale. Dans la première partie, l'état de l'art est présenté sous forme de trois chapitres. Le premier chapitre aborde une vision détaillée de la technologie grille: ses définitions, caractéristiques, domaines d'application, architecture, etc. Le deuxième chapitre est une présentation générale de la technologie des services web, son architecture et ses standards. Dans le troisième chapitre, nous parlerons des définitions et types de composition de services web, nous étudierons quelques langages de composition en se focalisant sur BPEL. La deuxième partie du mémoire sera consacrée à notre contribution qui consiste à gridifier des applications construites à base de services web et workflow. Elle est constituée du chapitre solution proposée dans lequel nous avons décrit les modules et les environnements utilisés pour atteindre nos objectifs ainsi que la mise en œuvre de la solution proposée. Enfin, ce mémoire se termine avec une conclusion générale et des perspectives.

[4] BPEL (Business Process Execution Langage) http://www-128.ibm.com/developerworks/library/specification/ws-bpel

CHAPITRE 1 : LES GRILLES DE CALCUL

1.1. Introduction

Le concept de « **grille de calcul** » ou en anglais (Grid **[Ifs 01]**) est apparu suite à la demande croissante des scientifiques de disposer de très grande puissance de calcul et capacité de stockage. Aussi, la collaboration des chercheurs et scientifiques nécessite l'accès et l'échange permanents d'importantes bases de données à travers Internet afin de corréler entre elles leurs informations, établir des statistiques, analyser des résultats, etc. De même, les chercheurs ont besoin d'effectuer des calculs importants pour modéliser une situation, etc. Ces traitements exigent la plupart du temps des moyens de calcul et de stockage considérables. De plus, ils correspondent à une utilisation intensive des ressources de calcul la plus part du temps pendant un laps de temps réduit.

Les centres de calcul ne suffiront plus d'ici quelques années pour ce type de traitement. Aussi l'investissement nécessaire pour ces calculs puissants ne peut être pris en charge par des entités isolées. Dans ce contexte, depuis quelques années une multitude de grilles émergent dans le monde scientifique mais également dans les sociétés commerciales pour être la solution permettant de répondre à la problématique du besoin exponentiel de puissance de calcul et de capacité de stockage.

Ce chapitre nous permet de faire une large revue d'ensemble sur le fonctionnement et les champs d'application des technologies de grilles.

1.2. Les grilles

Le terme « **grille de calcul** »**,** vient de l'analogie avec le réseau de distribution d'électricité en Anglais appelé ***electrical power grid***. En effet, on peut rapprocher la puissance de calcul actuelle avec ce qu'était l'électricité au début du XXe siècle **[Tpr 04] [Mqu 09]**. On maitrisait l'électricité et disposait de matériel pour l'exploiter mais chaque personne devait produire sa propre électricité et la consommer sur place. Ce n'est qu'avec le développement du réseau de distribution électrique que la vraie révolution a pu s'opérer en offrant, au plus grand nombre, la possibilité de recevoir et utiliser de l'électricité sans se soucier de la façon ou de l'endroit où elle a été produite. Actuellement, chaque ordinateur utilise la puissance de calcul qu'il a généré localement. L'idée de grilles de calcul est de mettre à disposition ses ressources en échange d'un accès aux ressources des autres utilisateurs.

Figure 1. 1 : Grille de calcul

Une **grille de calcul** est une infrastructure virtuelle car les relations entre les entités qui la composent n'existent pas sur le plan matériel mais d'un point de vue logique, elle est constituée d'un ensemble de ressources informatiques (tout élément qui permet l'exécution d'une tâche ou le stockage d'une donnée numérique : ordinateurs personnels, téléphones mobiles, calculatrices, …) potentiellement : **[Mba 02] [Lba 06] [Mqu 09]**

- **Partagées** : elles sont mises à la disposition des différents consommateurs de la grille et éventuellement pour différents usages applicatifs.
- **Distribuées** : elles sont situées dans des lieux géographiques différents.
- **Hétérogènes :** elles sont de toute nature, différentes par exemple par le système d'exploitation ou le système de gestion des fichiers.
- **Coordonnées** : elles sont organisées, connectées et gérées en fonction de besoins (objectifs) et contraintes (environnements). Ces dispositions sont souvent assurées par un ou plusieurs agents, qu'ils soient centralisés ou répartis.
- **Non contrôlées** (ou **autonomes**) : les ressources ne sont pas contrôlées par une unité commune. Contrairement à un cluster, les ressources sont hors de la portée d'un moniteur de contrôle.
- **Délocalisées** : les ressources peuvent appartenir à plusieurs sites, organisations, réseaux et se situer à différents endroits géographiques.

1.3. Classement et taxonomie

Les points de vue divergent sur les catégories de grilles selon leurs fonctionnalités. D'après **[Tpi 05]** et **[Mqu 09]** on distingue 3 catégories :

1.3.1. Grille d'information

Une grille d'information est une infrastructure générique qui facilite l'intégration de toute information n'importe où à travers des sources de données hétérogènes dans un environnement réseau. En particulier, une grille d'information constitue une virtualisation des sources de données pour permettre une intégration de toute information disponible. Ce type de grille permet de partager la connaissance. On a même considéré que le Web représente une application à succès du concept de Grille. Toutefois, la source de l'information n'est pas toujours connue (sources fiables?).

1.3.2. Grille de stockage

Ce principe de grille permet le partage de données (Musique, Vidéo, Données scientifiques, …) externalisées entre plusieurs nœuds vers plusieurs nœuds. Cette catégorie permet le stockage de données à grande échelle. Les cas les plus représentatifs de ce concept sont les réseaux d'échange Peer-To-Peer ("*P2P*") **[Gfe 03]**. Ce système permet l'accès à des données (fichiers, flux, etc.) via un réseau de sites (ou serveurs) qui contiennent et partagent un index. Les données sont référencées pour optimiser les recherches à travers un moteur de recherche. Les fichiers peuvent se trouver sur des nœuds différents du réseau, en différents points du globe. Il peut donc y avoir plusieurs copies d'un même fichier afin d'optimiser l'obtention de ce dernier. Parmi ces systèmes, on pourra notamment citer l'existence du réseau GNUtella[5] **[Gnu 10]** ou encore KaZaA[6] **[Kaz 10]**. Mais ce genre de principe est relativement vulnérable, de par le partage de fichiers ou documents illégaux. Un autre exemple de grille de stockage est le Projet S3[7] (Simple Storage Service) d'Amazon **[Sss 10]**.

D'après **[Yde 07]**, la grille de stockage est un nouveau modèle pour le déploiement et la gestion des ressources de stockage réparties sur plusieurs systèmes et réseaux, en utilisant efficacement les capacités de stockage disponibles.

1.3.3. Grille de calcul

Une grille de calcul est un dispositif logiciel qui offre aux utilisateurs des puissances quasi illimitées de calcul ou de stockage de données, grâce à un accès transparent et facile (une simple connexion à un réseau à très haut débit de type Internet) à un vaste ensemble de ressources informatiques distribuées sur une grande échelle. **[Grl 09]**.

[5] http://sourceforge.net/projects/rfc-gnutella/
[6] http://www.kazaa.com/
[7] http://aws.amazon.com/fr/s3/

Nous verrons dans ce qui suit (section 1.4) plus amples détails sur les grilles de calcul.

1.4. Grille de calcul

C'est sur la grille de calcul que s'exécutent les programmes. Deux éléments sont indispensables pour qu'une grille soit réalisable : des réseaux à très haut débit (ex : ADSL) sur de longues distances et une capacité à gérer la qualité de service. L'évolution des technologies de communication le rend aujourd'hui possible. Cette nouvelle donnée est susceptible de modifier en profondeur la problématique des moyens de calcul dans trois grands domaines : le calcul intensif, la visualisation et les grandes bases de données. Si les ressources en matériel existaient déjà en grande partie (centres de calcul, ordinateurs des laboratoires et réseaux), il restait à construire toute l'infrastructure logicielle sur laquelle reposera la grille et qui comprend, entre autres : l'identification et la sécurité, les logiciels d'arbitrage de ressources, de suivi des applications et de garantie de qualité, et l'interface utilisateur. **[Grl 09]**

Dans divers cas, il faut utiliser d'autres expressions que '**grille de calcul'**. Il est utile de distinguer les concepts suivants : **[Rbu 10]**

- **Calculateur ou supercalculateur :** unité informatique (machine) dédiée au calcul, dont la puissance est notamment caractérisée par l'adjonction de plusieurs processeurs ;
- **Réseau :** association physique ou logique d'unités informatiques qui *collaborent* (par exemple : elles partagent des données), qui évoluent de manière indépendante sur le plan fonctionnel et qui dépendent d'une unité centrale de contrôle (domaine) sur le plan administratif ;
- **Cluster :** regroupement d'unités informatiques qui *coopèrent* (la finalité est commune) et forment une seule unité informatique virtuelle sur les plans fonctionnel et administratif ;
- **Grille :** agrégat de réseaux autonomes (l'autonomie s'exprime au niveau de chaque réseau) et hétérogènes (l'hétérogénéité se manifeste aussi bien au sein d'un réseau qu'entre deux réseaux) d'unités informatiques ou apparentées.

Il n'existe pas de définitions très précises des grilles de calcul.

- ✓ **Plaszczak et Wellner [Ppl 05],** définissent la grille de calcul comme la technologie qui permet la virtualisation de ressource à la demande, et le partage entre plusieurs organisations.

- ✓ **IBM** dans **[Ibm 02]** définit le calcul en grille comme une méthode utilisant une panoplie de protocoles standards pour avoir l'accès aux applications, aux données, à la puissance de calcul, et à la capacité de stockage entre un vaste réseau de ressources informatiques à travers Internet.
- ✓ **Buyya** définit la grille comme un type de système parallèle et distribué qui permet le partage, la sélection, et l'agrégation dynamiques de ressources autonomes, géographiquement distribués lors de leur exécution en fonction de leur disponibilité, capacité, performance, coût, et des besoins des utilisateurs en terme de qualité de service. **[Rsv 05][Rbu 10]**
- ✓ Le **CERN** (*European Organization for Nuclear Research*) **[Cer 08]**, un des plus gros consommateurs de puissance de calcul à travers la technologie de grille de calcul, il la définit comme un service pour le partage de puissance informatique et de capacité de stockage de données à travers l'Internet. **[Gri 10]**
- ✓ **Ian Foster** dans **[Ifo 02]** présente trois critères de base pour distinguer une grille d'un autre système distribué:
  - Des ressources coordonnées dont leur administration n'est pas centralisée,
  - Des méthodes utilisées qui sont standardisées (en utilisant des normes, ouvertes, des protocoles et des interfaces à des fins générales),
  - Délivrer une Qualité de Service non négligeable (non triviale) (temps de réponse, disponibilité, sécurité, etc.).

1.5. Les différentes approches de grille de calcul

Une grille de calcul permet, de répartir un calcul qui a besoin de grandes capacités sur des postes distants. Cette répartition est gérée par plusieurs concepts dans la grille. Des auteurs comme **[Tpr 04], [Tpi 05]** et **[Mqu 09]** ont distingué les 3 approches suivantes :

1.5.1. Le Supercalculateur Virtuel (Virtual Supercomputing)

C'est l'association de plusieurs supercalculateurs répartis géographiquement (10-1000). Chaque nœud est une machine parallèle contrôlée par un gestionnaire de tâches (ex: batch). Cela offre une vision d'un supercalculateur virtuel. Citons l'ancienne version de Globus[8] **[Glo 09]** qui offre une boite à outils pour la construction de supercalculateurs virtuels à l'échelle d'Internet et fait exécuté ses applications sur des ressources distantes.

[8] Globus est un logiciel open source pour les grilles qui répond aux problèmes les plus difficiles dans le partage des ressources distribuées.

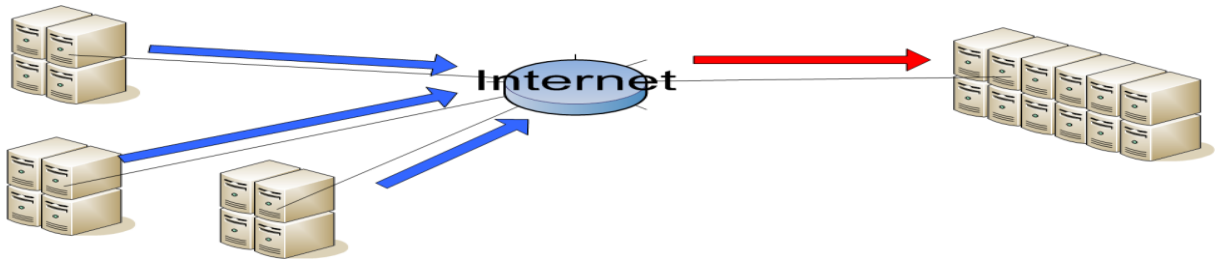

Figure 1. 2: Virtual Supercomputing

1.5.2. La grille pour PC (Internet Computing ou Desktop Grid)

C'est le concept le plus simple à mettre en œuvre et le plus répandu. C'est la combinaison d'un très grand nombre de PC (10000 -1000000). Cela permet d'exploiter le PC lorsque celui-ci est inutilisé. Les unités de traitement — CPU ou processeurs — sont généralement exploitées à moins de 10 % de leurs capacités réelles. Dans une enquête d'Omni ConsultingGroupQuickTime dans **[Tpr 04]**, ils citent les exemples d'Internet Computing suivants : SETI@home[9] (pour la recherche de signaux extra-terrestres, 62 Teraflop/s à comparer aux 36 Teraflop/s de l'ordinateur le plus puissant au monde au Japon !), Décrypthon[10] (pour établir la carte des 550 000 protéines du monde vivant) et RSA-155[11] (pour casser des codes cryptographiques)

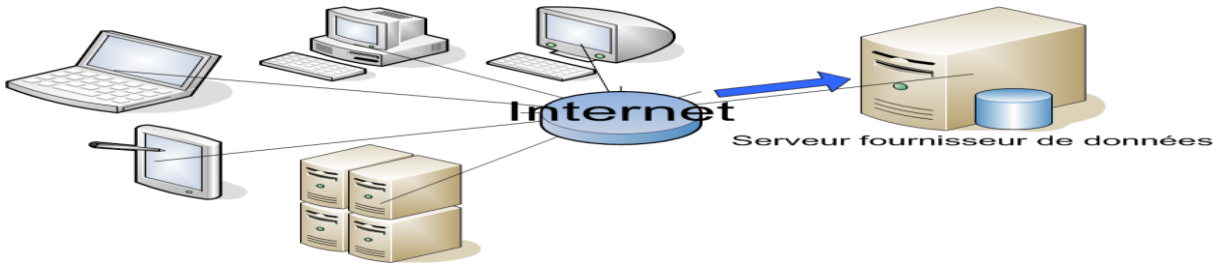

Figure 1. 3: Desktop Grid

1.5.3. Le Metacomputing

Le Metacomputing est le modèle client/serveur pour les grilles de calcul **[Tpr 04]**. C'est l'association de plusieurs machines proposant des applications. Le principe est d'acheter du service de calcul sur l'Internet. Ce service représente des applications préinstallées plus des calculateurs. Par exemple, plutôt que de demander l'énergie pour faire chauffer le café,

[9] http://setiathome.free.fr/
[10] http://www.decrypthon.fr/ewb_pages/h/historique.php
[11] http://www.rsa.com/rsalabs/node.asp?id=2098

acheter le café chaud… Citons par exemples Netsolve[12] (Univ. Tennessee), NINF[13] (Univ. Tsukuba), DIET[14] (ENS Lyon/INRIA).

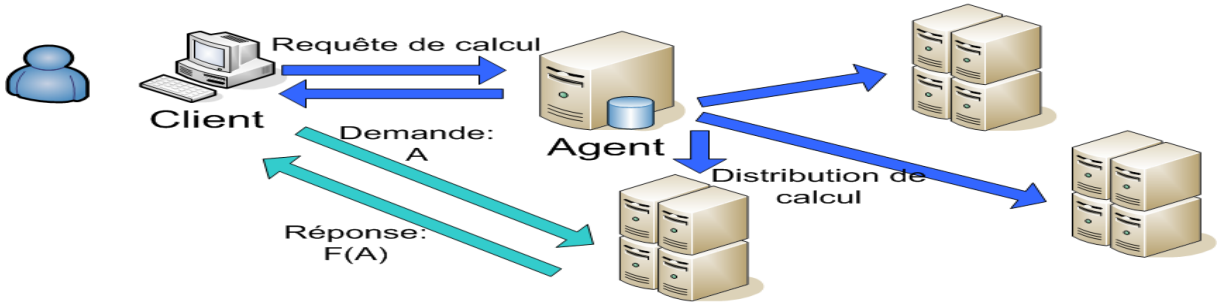

Figure 1. 4: Metacomputing

1.5.4. Synthèse sur les différentes approches

D'après **[Tpi 05],** chacune des approches énumérées ci-dessus rencontre des problèmes et des défis. Par exemple, l'ancienne version de **'Globus'** n'avait pas de protocole de communication ni de standard pour l'invocation, la notification, la propagation des erreurs, la sécurité, la terminaison, … et l'accès aux services se faisait au travers d'APIs en constante évolution. Pour le **'Desktop Grid'**, il y'a le problème d'élargissement du spectre d'application (de vrais applications parallèles, autoriser la communication), la sécurité (êtes-vous prêt à laisser exécuter n'importe quoi sur votre PC ?, comment communiquer avec le monde lorsqu'on est isolé ?, peut-on croire dans les résultats fournis par quelqu'un que l'on ne connaît pas ?), la parité (Cela marche si tout le monde joue le même jeu…) et le modèle de déploiement (essentiellement client/serveur, à terme, nécessité du P2P). Enfin pour le **'Metacomputing'** il doit relever les défit sur, le stockage des données (pour éviter les transferts multiples entre client et serveurs, la gestion des données distribuées et redistribution), la sécurité (dans les transferts et dans les calculs) ainsi que le modèle de déploiement (P2P).

Cela dit, ces approches ont permis de faire des expériences et de donner naissance à des standards émergents à l'initiative des chercheurs et soutenus fortement par quelques grands acteurs de l'informatique (exemples de standards : OGSI[15], WSRF[16]).

[12] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.38.8341&rep=rep1&type=pdf
[13] http://ninf.apgrid.org/
[14] http://graal.ens-lyon.fr/DIET/
[15] OGSI (Open Grid Service Infrastructure) http://www.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf
[16] WSRF (Web Services Resource Framework) http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

1.6. Les besoins d'une grille

En tenant compte des problématiques et défis des différentes approches, nous concluons que les besoins d'une grille sont les différents aspects de sécurité qui sont l'authentification, l'autorisation et l'accounting, la gestion des ressources qui comprend la découverte, la caractérisation et l'allocation, le niveau application qui concerne l'accès aux données distantes, le transfert rapide, la gestion des pannes et des fautes ainsi que les algorithmiques distribués et puis aussi, les aspects de monitoring, de garantie des performances et des évolutions des systèmes.

1.7. Domaines d'application des grilles

Le principe de grille de calcul est utilisé dans une multitude de projets, pour des objectifs différents. **[Kta 09]** dans son étude cite les applications suivantes :

1.7.1. Les sciences de la physique

Les sciences de la physique est un domaine en plein essor avec le développement de la technologie grille. Le projet LHC **[Lcg 09]** du CERN a été lancé en 2007 pour fournir un débit de données massives (de 10Gb/s à 100Gbit/s) où chaque événement de la physique peut être traité indépendamment, résultant de parallélisme de billions de voies. Ce grand défi ne peut être résolu par un seul ordinateur ou cluster. Par conséquent, les scientifiques essaient maintenant de trouver des moyens pour résoudre ce problème avec les nouvelles possibilités offertes par la grille de calcul. A cet effet, plusieurs projets de grille sont créés ou impliqués: LCG **[Lcg 09]**, EGEE **[Ege 09]**, DataGrid **[Dat 09]**, GridPP **[Grp 09]** et INFN **[Inf 09]**.

1.7.2. L'astronomie

L'astronomie a toujours été un domaine dynamique de la recherche étant donné que l'être humain est toujours curieux de l'espace. À partir d'images traitées de planètes à d'énormes quantités de données brutes, il ya une grande quantité de données basées sur l'astronomie disponibles sur Internet. Il en résulte des exigences de stockage importantes. Les projets de grille D-Grid **[Dgr 09]**, DutchGrid **[Dut 09]**, OSG **[Osg 09]**, ChinaGrid **[Chi 09]** et SDG **[Sdg 09]** sont tous impliqués dans ce domaine de recherche.

1.7.3. Le biomédical

La biologie de calcul a changé d'une science traditionnelle de calcul intensif à une science de haut débit, pilotée par les données. Beaucoup d'expérimentations et d'équipements de mesure sont directement liés aux ressources informatiques pour obtenir et traiter

rapidement les données. La technologie DataGrid **[Dat 09]** a également été élaborée pour recueillir et traiter d'énormes ensembles et bases de données en toute sécurité. Le projet de grille japonais nommé BioGRID **[Bio 09]** a pour objectif de faciliter le lien entre ces bases de données pour le traitement de données nécessitant des ressources de calcul ultra-rapides. En Europe, le projet EGEE **[Ege 09]** a offert des ressources de grille de calcul et de solutions de distribution de données, d'algorithmes, et de ressources de stockage pour la génomique. En fait, de nombreux pays ont établi des projets de grille de calcul spéciale pour la recherche biologique, en vue du développement pharmaceutique et l'élucidation biodynamique.

1.7.4. L'observation de la terre et climatologie

Les satellites d'observation terrestre retournent 100 giga-octets d'images de données par jour. L'équipement de stockage sur le terrain a déjà stocké plus de 1 000 000 giga octets de données. Ces données peuvent être utilisées pour analyser les profils d'ozone ou à détecter du pétrole. La grille permet à ces données d'être facilement partagées entre les différents producteurs et consommateurs. Par exemple, DataGrid **[Dat 09]** a réalisé le stockage de données réparties dans toute l'Europe.

1.7.5. Autres applications

En plus de ces applications clés mentionnées précédemment, de plus en plus de scientifiques et de chercheurs s'intéressent aux grilles de calcul. Des universitaires aux industrielles, la grille de calcul est de plus en plus utilisée par les entreprises et les sciences. Par exemple, BEinGRID **[Bei 09]** est un projet de grille qui se focalise sur les problèmes commerciaux. Ses principales applications se trouvent dans des expériences d'entreprises sur des grilles. Le projet Chinois SDG **[Sdg 09]** et le projet Japonais NAREGI **[Nar 09]** ont également effectué la recherche en nanosciences dans la grille. En résumé, le tableau ci-dessous présente quelques domaines d'application de certains projets de grille.

	Physique de haute énergie HEP	**Biomédicale**	**Astronomie**	**Observation terrestre et Climatologie**	**Nanotechnologie**	**Entreprise**
EGEE	Oui	Oui	Oui	Oui		
D-GRID	Oui		Oui			
BeInGrid						Oui
CrossGrid	Oui	Oui		Oui	Oui	
LCG	Oui					
DutchGrid	Oui		Oui	Oui		Oui
GridPP	Oui					
OSG	Oui	Oui	Oui		Oui	
DATAGrid	Oui	Oui		Oui		

Tableau 1 : Domaines d'application de certains projets de Grille

Les travaux de recherche sur les grilles sont principalement pour le développement de grille. De plus en plus d'ingénieurs et de scientifiques participent à ce domaine de recherche. Ils proviennent de disciplines différentes. Leur travail implique une grande quantité de calculs scientifiques, qui exigent une grande quantité de ressources de calcul et de produire des données à grande échelle. Nous avons énuméré quelques uns dans les domaines précédents. Pour plus amples informations, voir l'annexe A.

1.8. Architectures d'une grille (modèle en couche)

L'architecture détermine les principales couches d'une grille, les fonctionnalités de chaque couche et les interactions entre elles. Nous allons présenter dans ce qui suit l'architecture en couche selon Baker et al et Foster et al:

1.8.1. Architecture Selon Baker et al

L'architecture d'une grille est souvent décrite sous la forme d'une série de couches empilées, chacune remplissant un rôle spécifique.

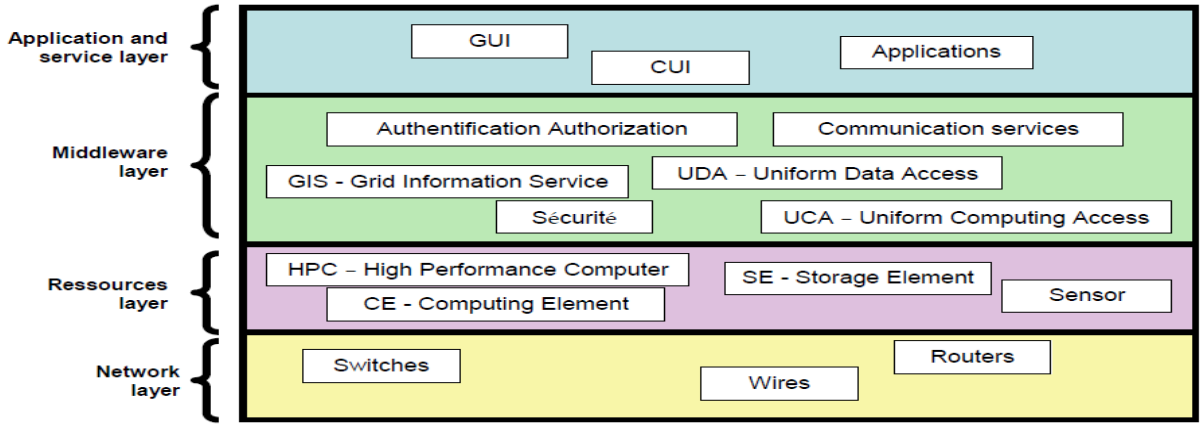

Figure 1. 5: Architecture en couches d'une grille de calcul [Mba 02]

Nous retrouvons dans les couches supérieures les fonctionnalités proches de l'utilisateur alors que les couches inférieures, de bas niveau, représentent les éléments constituant l'infrastructure hardware et réseau. (Figure 1.5)

1.8.1.1. La couche applicative

C'est la couche « visible » par les utilisateurs. Cette couche donne l'accès aux ressources par l'intermédiaire d'interfaces graphiques GUI (Graphical User Interface) ou en mode commande CUI (Command User Interface) ou totalement intégrées dans les logiciels spécifiques.

1.8.1.2. La couche middleware ou intergiciel

Elle symbolise l'ensemble des fonctionnalités permettant l'exploitation des ressources par les utilisateurs de la grille. C'est le cœur même de la grille. C'est elle qui gère les ressources, les réservations et les ordonnancements des différents jobs pris en charge. Cette couche offre un ensemble de services tels que la sécurité, l'authentification, le service d'information, de communication, d'ordonnancement, etc.

1.8.1.3. La couche ressource

Elle symbolise l'ensemble des ressources mises à la disposition de la grille, tels que les processeurs, les espaces de mémoire, les catalogues de données, les softwares modulaires ainsi que tout type d'appareils tels que les capteurs.

1.8.1.4. La couche réseau

Elle symbolise l'infrastructure de télécommunication permettant la connexion des différents éléments de la grille.

1.8.2. Architecture Selon Foster et al

Voici une autre représentation inspirée du modèle d'Ian Foster **[Ifs 01]** qui présente l'architecture de grille au regard de l'architecture IP.

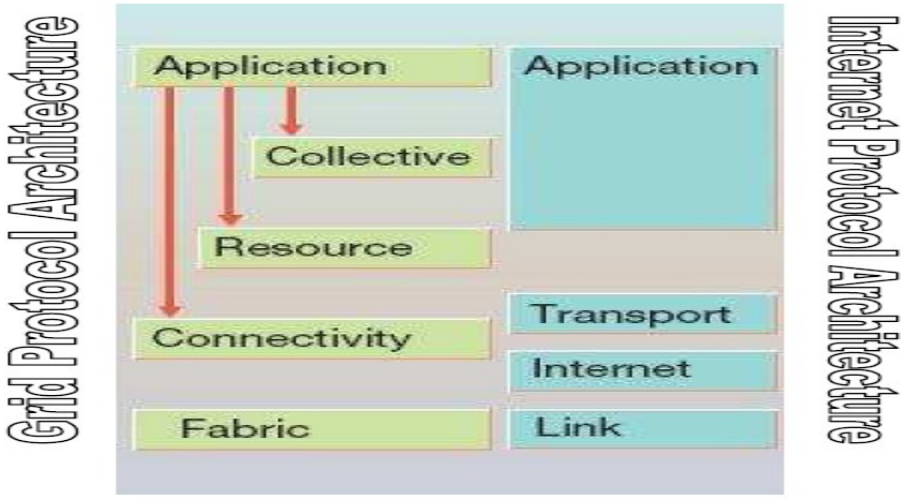

Figure 1. 6: Architecture en couches d'une grille de calcul — Foster [Ifs 01]

1.8.2.1. La couche applicative

Est située au niveau le plus élevé, symbolise l'ensemble des applications donnant accès aux ressources de la grille. C'est la couche visible par l'utilisateur.

1.8.2.2. La couche collective

Cette couche assure la gestion et l'interaction des ressources globales de la grille. Cette couche gère les interactions entre les différentes ressources de la grille.

1.8.2.3. La couche ressources

Elle assure les fonctions de contrôle et d'accessibilité aux ressources ainsi que le monitoring, l'accounting. La couche ressources accède et contrôle les ressources locales.

1.8.2.4. La couche connexion

Elle symbolise les protocoles nécessaires à l'authentification et l'accès pour les transactions souhaitant exploiter les ressources de la grille. Une notion essentielle dans les techniques de sécurité des grilles de calcul est celle de l'**organisation virtuelle**[17] qu'on verra plus loin dans la section 1.10.

1.8.2.5. La couche fabrique

Elle symbolise les interfaces vers les ressources locales telles que les disques pour le stockage des données, les CPU, les réseaux, les softwares modulaires ainsi que tout autre type de ressource mise à la disposition de la grille.

1.9. Middleware

Une **grille de calcul** intègre un **intergiciel**, en anglais **middleware**, qui permet le dialogue entre les différents composants d'une application répartie.

1.9.1. Définition

Le middleware est une couche logicielle intermédiaire entre les applications et le réseau, permettant le dialogue entre des applications hétérogènes. D'après **[Kta 09]**, dans le jargon des grilles, un middleware est une collection d'API, protocoles et logiciels qui permettent l'exploitation des ressources de la grille. Il offre une transparence de l'hétérogénéité et de la distribution du système.

1.9.2. Le middleware dans la couche protocolaire

Dans l'architecture d'une grille, comme nous l'avons vu dans les sections précédentes, la couche la plus basse est la couche Fabrique composée des ressources proprement dites, la

[17] Organisation Virtuelle

couche la plus haute est composée des applications utilisateurs. Le middleware se trouve entre ces deux couches comme le montre la figure 1.7 **[Kta 09]**.

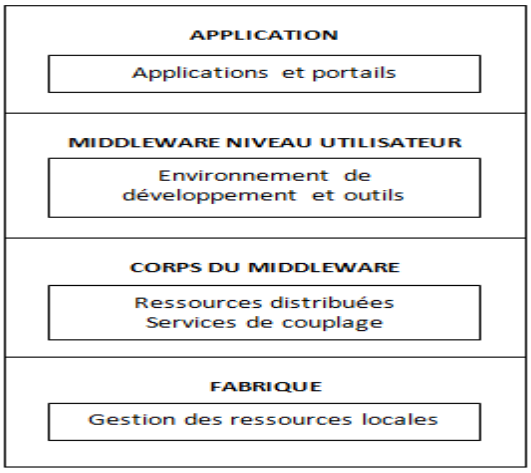

Figure 1. 7 Le middleware dans l'architecture de la grille

Dans cette architecture, le middleware est divisé en deux niveaux : le corps du middleware et le middleware niveau utilisateur. Le corps du middleware offre les services élémentaires tels que l'exécution des tâches, l'allocation des ressources, la sécurité, le service d'information et le transfert de fichier. Le niveau utilisateur comprend les environnements de développement, les outils de programmation, les applications de planification de tâche, etc.

Il existe plusieurs middlewares, les plus connus, sont : Globus **[Glo 09]** qui est très utilisé, Legion **[Leg 10]**, Unicore **[Uni 10]**, Condore **[Con 10]** et g-Lite **[Gli 09]** qui est de plus en plus adopté.

1.10. Organisation virtuelle

Les ressources d'une grille peuvent potentiellement être utilisées par une très large communauté d'utilisateurs. Il est donc nécessaire d'avoir une politique claire pour la gestion des droits associés aux ressources et à ceux qui peuvent les utiliser. Pour ce faire, on regroupe les utilisateurs en organisations virtuelles (*Virtual Organization* (VO)). Toutes les personnes appartenant à une même VO ont généralement des droits et besoins communs, typiquement parce qu'elles travaillent dans une même discipline.

Fellenstein et al définissent une organisation virtuelle comme étant: des entités logiques, limitées dans le temps, créées dynamiquement dans le but de résoudre un problème spécifique, et en fournissant et en allouant des ressources à la demande. **[Cfe 04]**

Foster et al ont introduit la notion d'organisation virtuelle comme suit : **"…Flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources"**. **[Ifs 01]**

L'organisation virtuelle prend en charge les aspects relatifs à la sécurité. Elle définit les conditions d'accès et la politique d'utilisation des ressources disponibles sur la grille tels que les cycles CPU, les capacités de stockage, les logiciels accessibles, les périphériques, etc.

Le succès futur des grilles de calcul dépendra en grande partie de la facilité à pouvoir créer et exploiter des organisations virtuelles. Il peut être intéressant pour différentes entités de recherche de réunir leurs ressources au sein d'une même entité virtuelle. Ceci permet d'augmenter le potentiel global du parc informatique et d'utiliser les ressources d'une manière plus rationnelle.

Les organisations virtuelles peuvent également être constituées juste le temps nécessaire à la résolution d'un problème spécifique. Des chercheurs ne disposant d'aucune ressource hardware ni software et ayant besoin d'une puissance de calcul importante peuvent contacter un fournisseur de services en grille et constituer une organisation virtuelle le temps nécessaire à la résolution de leur problème. Cette solution évite aux chercheurs d'investir dans une infrastructure informatique lourde et coûteuse et permet d'autre part à l'administrateur de la grille d'amortir une partie de son matériel hardware et software.

1.11. Fonctionnement d'une grille

Pour mieux comprendre le fonctionnement d'une grille, nous allons voir une brève description des différents éléments intervenants dans la prise en charge d'un job sur la grille EGEE basée sur le middleware g-Lite **[Ela 06] [Gli 09]** (Figure 1.8). Le chapitre 4 présente une vue détaillée sur les composants du middleware g-Lite.

Dans la figure 1.8, nous présentons brièvement quelques composants de la grille (gLite). (Pour plus de détail voir le chapitre 4)

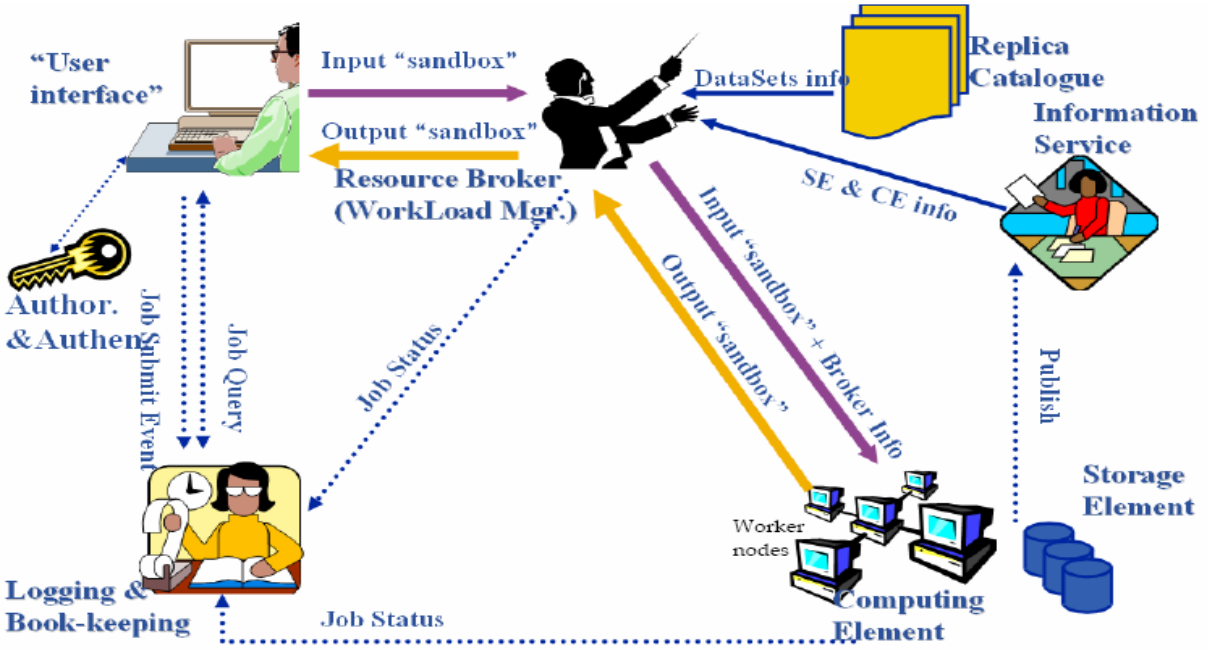

Figure 1. 8: Schéma d'exécution d'un job sur la grille — EGEE [Gli 09]

- **L'interface utilisateur (User Interface : UI) :** l'interface via laquelle l'utilisateur accède à la grille ;
- **Le système d'information (Information System : IS) :** le système qui collecte l'ensemble d'information relative aux ressources précisant les caractéristiques et l'état des éléments de calcul et de stockage (CE, SE) ;
- **Le système de gestion de la charge de travail (Workload Management System : WMS) :** est un ensemble de services ayant la responsabilité de trouver le meilleur élément de calcul disponible où soumettre les jobs utilisateur de manière transparente ;
- **Le gestionnaire de charge de travail (Workload Manager : WM) :** fait correspondre les besoins des utilisateurs avec les ressources disponibles sur la grille ;
- **L'élément de calcul (Computing Element : CE) :** représente le point d'accès unifié aux ressources de calcul, aux nœuds travailleurs (WN) qui seront utilisés par la grille pour l'exécution des jobs. Le CE se charge de la gestion des jobs qui lui sont attribués ;
- **L'élément de stockage (Storage Element : SE) :** organe de gestion du stockage d'information. Permet la gestion des fichiers dans la grille et offre les mécanismes pour les localiser facilement pour les utilisateurs et les jobs ;

- **Les nœuds travailleurs (Worker Nodes : WN) :** groupe de machines sur lesquelles les jobs vont être exécutés. C'est également sur les WNs que sont stockées les données transmises par le SE.

1.11.1. Schéma de prise en charge d'un job

Le cheminement d'un job exécuté sur une grille de calcul peut être schématisé comme suit : **[Glt 10] [Gli 09]**

1. Après avoir obtenu un certificat numérique d'une autorité de certification digne de confiance, en vous inscrivant dans une VO et d'obtenir un compte sur l'UI, l'utilisateur est prêt à utiliser la grille. Il se connecte à l'UI et crée un proxy qui lui permet d'interagir en toute sécurité avec le système ;
2. l'utilisateur soumet le job au WMS via l'UI. Le WMS recherche le/les CE(s) pouvant prendre en charge l'exécution du job en consultant l'IS. L'utilisateur transmet les fichiers d'entrée dans l'InputSandBox[18] ;
3. Le job ainsi que l'InputSandBox sont transférés au CE qui prend en charge le job dans sa queue (liste d'attente) ;
4. Le CE envoie le job sur un ou plusieurs WN (s) disponibles ;
5. Lorsque le job est terminé, les fichiers produits par celui-ci sont disponibles sur le LRMS[19] (Local Resource Management System). Le WMS est averti que le job s'est terminé ;
6. Le WMS récupère les fichiers de sortie dans l'OutputSandBox[20] ;
7. Le WMS envoie les résultats (l'OutputSandBox) à l'utilisateur via l'UI ;
8. L'utilisateur peut interroger à tout moment l'état de son job par l'intermédiaire du Logging and Bookkeeping Service (LB[21]). Le LB conserve une trace de l'exécution des jobs.

1.11.2. Les différents états d'un job soumis à la grille

Un job soumis à la grille peut prendre les états suivants :

[18] L'InputSendBox contient le chemin de l'exécutable et les paramètres de l'application. Il les transfère depuis l'UI au WNs (Chapitre 4)

[19] LRMS est le système de gestion de ressources locale (Chapitre 4)

[20] L'OutputSendBox contient les fichiers qui sont retournés par la grille, une fois l'exécution du job terminée. Il les transfère depuis les WNs à l'UI (Cahpitre 4)

[21] Le LB (Chapitre 4)

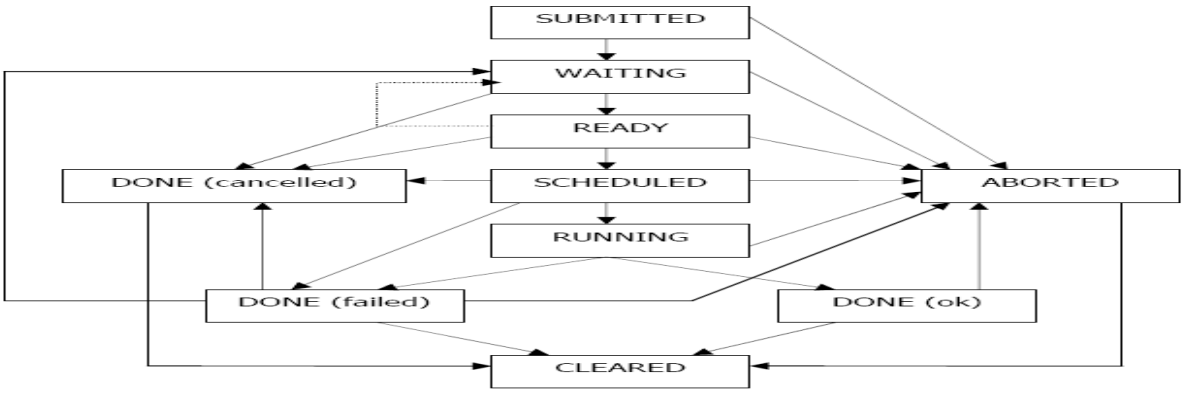

Figure 1. 9: La machine d'états d'un job sur la grille — EGEE [Hgj 08]

- **SUBMITTED :** le job a été soumis pour exécution par l'utilisateur mais il n'a pas encore été traité par le serveur WMProxy;
- **WAITING :** le job a été accepté par le serveur mais n'a pas encore été pris en charge par le WMS ;
- **READY :** le job a été assigné à un CE mais il n'a pas encore été transmis à celui-ci ;
- **SCHEDULED :** le job est en liste d'attente sur le CE ;
- **RUNNING :** le job est en cours d'exécution sur le WN ;
- **DONE :** l'exécution du job est terminée ;
- **ABORTED :** le job a été arrêté par le WMS (la durée d'exécution du job était trop longue, le certificat proxy est expiré, etc.) ;
- **CANCELED :** le job a été arrêté par l'utilisateur ;
- **CLEARED :** l'OutputSandbox a été transféré vers l'UI.

1.12. La normalisation dans la grille

Le problème rencontré dans la technologie des grilles est que chaque projet propose sa solution, ce qui rend l'utilisation de ces solutions ensemble une tâche très complexe. Pour remédier à ce problème, des standards ont été introduits dans la mise en œuvre d'une grille. Dans cette section, nous examinons certains de ces standards. **[Kta 09]**

1.12.1. Les services web

Les services web sont des composants software qui fournissent des fonctionnalités accessibles via des protocoles standardisés du Web (Chapitre 2). La combinaison de la technologie des grilles avec celle des services web a donné naissance aux *grid services*[22]. Par

[22] Traduction libre de Grid Services : Services web de la grille. Le Grid Service est un service Web avec des extensions pour le rendre apte pour les applications basées sur une grille

exemple, le *grid service*, définit par OGSA **[Ogs 08]**, est une extension des services web. Ainsi, les *grid service*s peuvent tirer parti des spécifications des services web.

Cette tendance de services web et grilles se confirme par plusieurs travaux, **Frascaria [Kfr 02]** en a cité les suivants:

- Sun qui va plus loin en proposant SG23E[23] avec SunOne et iPlanet dans une suite (Technical *Computer Portal*), rapprochant clairement les services web des grilles ;
- Microsoft qui a investi un million de dollars dans Globus **[Glo 09]** pour qu'il soit compatible avec Windows XP et sa plate-forme de développement de services web .NET ;
- En fait, sur ce secteur, l'un des plus importants travaux actuels reste l'Open Grid Services Architecture (OGSA) qui vise à combiner la technologie des services web à celle des grilles de calcul.

1.12.2. L'Open Grid Services Architecture (OGSA)

L'Open Grid Services Architecture (OGSA) **[Ogs 08]** définit un cadre fondé sur des services web pour l'installation d'une grille. Elle vise à uniformiser les services fournis par une grille comme l'exploration des ressources, gestion des ressources, sécurité, etc., à travers une interface standard de services Web. Elle définit également les caractéristiques qui ne sont pas forcément nécessaires à l'installation d'une grille, mais sont néanmoins souhaitables. L'OGSA repose sur les spécifications des services web et ajoute des fonctionnalités à ces derniers pour les rendre compatibles avec l'environnement de grille. La littérature d'OGSA parle de *grid services*, une extension aux services web adaptés aux exigences de la grille. **[Kta 09]**

1.12.3. L'Open Grid Services Infrastructure (OGSI)

L'OGSA **[Ogs 08]** décrit les fonctions qui sont nécessaires pour l'implémentation des services fournis par la grille, en tant que services web. Cependant, elle ne fournit pas les détails de l'implémentation. L'OGSI **[Stu 07]** prévoit une spécification formelle et technique nécessaire à l'implémentation des services de la grille. Elle fournit une description du Web Service Description Language (WSDL[24]) **[Ech 01]**, qui définit un *grid service*. L'OGSI prévoit également les mécanismes de création, de gestion et d'interaction entre les *grid services*.

[23] SG23E est le Sun Grid Engine Enterprise Edition
[24] WSDL est le langage de description des services web, basé sur le standard XML. Voir le chapitre 3

1.12.4. Le Web Services Resource Framework (WSRF)

La motivation derrière le développement du WSRF **[Wsr 07]** est de définir un cadre (framework) générique et ouvert pour la modélisation et l'accès aux ressources avec état (statful ressources) utilisant des services Web. Il définit les conventions pour la gestion d'état permettant aux applications de découvrir et d'interagir avec des services Web avec état (statful web services) de façon standard. Les Services Web standards ne possèdent pas une notion d'état. Les applications basées sur la grille ont besoin de la notion d'Etat, car elles exercent souvent une série de demandes pour lesquelles la production d'une opération doit dépendre du résultat des opérations précédentes. WSRF peut être utilisé pour développer ces *grid services* avec état. Le format d'échange de messages en WSRF est défini par le WSDL. Le WSRF est soutenu par diverses sociétés et le cahier des charges a été finalisé par le Comité de travail OASIS[25].

1.12.5. L'Open Grid Services Architecture-Data Access and Integration (OGSA-DAI)

L'OGSA-DAI **[Ogsd 07]** est un projet conçu par la Force opérationnelle de base de données britannique (UK Database Task Force). Le but de ce projet est de développer des middlewares pour fournir l'accès et l'intégration aux sources de données distribuées à l'aide d'une grille. Ce middleware apporte son support à diverses sources de données telles que les bases de données relationnelles et XML. Ces sources de données peuvent être interrogées, actualisées et transformées via le service web OGSA-DAI. Ces services Web peuvent être déployés dans une grille, rendant ainsi les sources de données grid-enabled. La requête au service Web OGSA-DAI pour accéder à une source de données est indépendante de la source de données desservie par le service Web. Les services web OGSA sont conformes aux spécifications WS-I[26] et WSRF, les deux spécifications les plus importantes pour les services Web.

1.13. Avantages et inconvénients de la grille

Comme toute technologie, les grilles de calcul ont des avantages et des inconvénients. Nous citons dans ce qui suit quelques uns.

[25] OASIS (Organization for the Advancement of Structured Information Standards) http://www.oasis-open.org/
[26] WS-I (Web Services Interopérability) http://www.ws-i.org/

1.13.1. Avantages

D'après ce qu'on a vu précédemment sur les grilles, nous pouvons sortir les avantages suivants :

- La technologie de grilles permet de résoudre des problèmes très grands et très complexes en un temps réduit ;
- Pas besoin d'acheter de grands serveurs pour les applications qui peuvent être divisés et exécutés sur les plus petits serveurs. Les résultats peuvent ensuite être concaténés et analysés une fois le job terminé ;
- Une utilisation beaucoup plus efficace des ressources existantes. Les jobs peuvent être exécutés sur des serveurs ou postes de travail inoccupés. Plusieurs de ces ressources restent inutilisables en particulier après les heures de travail. Des politiques peuvent être mises en place pour permettre aux jobs d'aller sur des serveurs qui sont peu chargés ou qui ont des caractéristiques appropriés de CPU/mémoire pour l'application à exécuter;
- Les environnements de grille sont beaucoup plus modulaires. Si l'un des serveurs/postes de travail au sein de la grille est en panne, il ya beaucoup d'autres ressources en mesure de prendre en charge la tache. Les jobs peuvent redémarrer automatiquement en cas de défaillance ;
- Les politiques sont gérées par le middleware de la grille;
- Ce modèle est évolutif. S'il ya besoin de plus de ressources de calcul? Il suffit de les brancher en installant le composant middleware nécessaire (CE, SE, WN, …). Ces ressources peuvent être supprimées aussi facilement ;
- Des mises à jour peuvent être faites sans interruption de service. Comme il ya beaucoup de ressources, certaines peuvent être mises hors ligne, tout en laissant suffisamment pour continuer le travail ;
- Les jobs peuvent être exécutés en parallèle en accélérant les performances. Les environnements de grille sont extrêmement bien adaptés à exécuter les jobs qui peuvent être divisés en plus petits morceaux et s'exécuter simultanément (en concurrence) sur plusieurs nœuds ;
- Les utilisateurs de la grille peuvent se situer dans des environnements complètement différents. En effet, il n'est pas nécessaire de posséder le même système d'exploitation, la même puissance de calcul, la même connexion au réseau, etc.

1.13.2. Inconvénients

Bien évidemment, il ne peut y avoir que des avantages. Il existe aussi des inconvénients :

- Les middlewares de grille et les standards sont encore en évolution ;
- La courbe d'apprentissage pour commencer à travailler avec cette technologie ;
- La soumission de jobs interactifs n'est pas encore supportée, mais peut être détournée par la décomposition du job interactif en sous-jobs et l'implémentation des interactions par un langage riche et expressif tel que BPEL ;
- Il peut y avoir besoin d'une rapide interconnexion entre les ressources de calcul (Gigabit comme un minimum) ;
- Certaines applications peuvent avoir besoin d'être modifiées pour tirer pleinement parti du nouveau modèle (réécriture du code) ;
- Il peut y avoir besoin de licence sur plusieurs nœuds ;
- Les enjeux politiques liés à partager les ressources. De nombreux groupes sont réticents à partager les ressources, même si le profit est commun ;
- Un des gros problèmes de ce système est l'impossibilité de prévoir à l'avance les ressources réelles qui seront présentes et disponibles à un instant T. On ne peut donc savoir exactement combien de temps le calcul va prendre.

1.14. Conclusion

Les grilles informatiques sont des technologies qui ne sont pas récentes, mais elles suscitent de plus en plus l'engouement à travers le monde. En effet, elles représentent une fabuleuse alternative au calcul intensif classique, réalisé par des clusters.

Toutefois, cette approche reste encore dans des phases de prototypage, bien que certains projets, comme la Globus Alliance, essaient de régler ce problème. En effet, chaque projet développé sur ce principe n'a suivi aucun processus de normalisation. Le véritable problème de la technologie pour percer est donc de renforcer leur position sur le marché. Le fait d'assurer un suivi constant à cette technologie permettra, dans les années à venir, de pallier la faiblesse des outils de redistribution de la puissance. Il faudra, pour être normalisée, que cette technologie devienne un standard dans l'industrie, chose qui tend à se développer encore de manière très sporadique.

Certaines entreprises ont vite compris l'intérêt de ce concept. En effet, certaines sociétés commencent à entrevoir des possibilités de standardisation, tels que Oracle avec sa base

10G[27] (**G** pour **Grille**), IBM avec sa World Community Grid[28]. N'oublions pas la notion de services web qui prend de plus en plus place dans la standardisation des grilles.

Dans les prochains chapitres, nous allons voir en détail ce que recouvre la notion de « Services Web » et « Orchestration de services web » ainsi que les technologies nécessaires à leur développement et déploiement.

[27] Large Scale DataWarhouses On Grid: Oracle Database 10g and HP ProLiant Servers http://delivery.acm.org/10.1145/1090000/1083713/p1055-poess.pdf?key1=1083713&key2=3922315031&coll=DL&dl=ACM&ip=193.194.81.47&CFID=22365587&CFTOKEN=97665972

[28] Le World Community Grid a pour mission de créer la plus vaste grille de calcul distribué au monde afin d'aborder des projets qui bénéficieront à l'humanité entière. http://www.worldcommunitygrid.org/

CHAPITRE 2 : LES SERVICES WEB

2.1. Introduction

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 Services Web définissent une manière standard d'interagir avec des applications distantes en utilisant les technologies du Web.

Dans cette partie, nous revenons sur la genèse de cette technologie et nous allons essayer, par la suite, de décrire les solutions apportées par les services web pour repousser les limitations de l'informatique repartie.

2.2. Naissance des Services web

Les services web ont pour objectif de repousser voire de supprimer toutes les limitations des technologies de mise en œuvre de l'informatique répartie et surtout d'apporter une réponse au problème d'interopérabilité des systèmes informatiques. Ils s'inscrivent donc dans la continuité des initiatives CORBA **[Cor 10]** de l'OMG "Object Managment Group", EJB **[Ejb 10]**/RMI **[Rmi 10]** de Sun, DCOM **[Dth 97]** de Microsoft,…, en apportant toutefois une réponse plus simple et s'appuyant sur des technologies et des standards reconnus et maintenant acceptés par tous **[Mvi 04].**

Les services web sont nés aussi de cette volonté de définir une infrastructure technique favorisant l'intégration d'applications inter- et intra-entreprise s'appuyant sur les technologies issues du monde de l'Internet afin d'en tirer le meilleur profit.

La particularité des services web réside dans le fait qu'ils utilisent la technologie Internet comme infrastructure pour la communication entre les composants logiciels (les services web) et ceci en mettant en place un cadre de travail basé sur un ensemble de standards. Cette technologie de services web, initiée par IBM et Microsoft, puis en partie normalisée par le W3C est maintenant acceptée par l'ensemble des acteurs de l'industrie informatique sans exception. C'est surtout ce point qui fait des services web une technologie révolutionnaire.

Rôle de XML[29] dans l'infrastructure Services Web

Ayant fait le choix des protocoles du Web comme support de communication entre les composants, la syntaxe et la sémantique d'un service sont fondées sur XML qui représente un facteur important pour contourner les barrières techniques. En effet, il apporte à l'architecture l'extensibilité et la neutralité vis à vis des plates-formes et des langages de développement. De plus, grâce à sa structuration, XML permet la distinction entre les données des applications et les données des protocoles permettant ainsi une correspondance facile entre les différents protocoles. L'interopérabilité entre les systèmes hétérogènes demande des mécanismes puissants de correspondance et de gestion des types de données des messages entre les fournisseurs et les clients. C'est une tâche où les schémas de type de données XML s'avèrent bien adaptés. **[Tme 04].**

Figure 2. 1: Rôle de XML dans le processus de standardisation

2.3. Architecture Orientée Service (SOA)

2.3.1. Principe de l'architecture

L'une des dernières tendances en matière de logiciels est le concept d'Architecture Orientée Service (SOA). Ces architectures SOA, comme leur nom l'indique, ne représentent pas une technologie mais une façon de concevoir et de déployer des applications. Plus précisément, il s'agit de structurer des projets selon une approche basée sur le principe de "services" et non plus, comme par le passé, sur la base d'applications. Plutôt que de privilégier une architecture applicative basée sur des contraintes techniques, l'architecture SOA propose de découper les fonctionnalités d'une application en services métier réutilisables dans d'autres applications **[Pdu 04]**.

Le concept de « service » est un concept clé de l'architecture orientée service. Il consiste en une fonction ou fonctionnalité bien définie. C'est aussi un composant autonome qui ne dépend d'aucun contexte ou service externe.

L'architecture des services web comme l'architecture du Web sont des instances de l'architectures orientées services (SOA : Service Oriented Architecture). Elle propose une

[29] XML : eXtensible Markup Language

perspective globale sur le développement, la gestion et le fonctionnement des services web **[Dkb 03]**.

L'architecture SOA est un modèle abstrait qui définit un système par un ensemble d'agents logiciels distribués qui fonctionnent ensemble afin de réaliser une fonctionnalité globale préalablement établie **[Khe 01]**. Les agents dans un système distribué n'opèrent pas dans un même environnement de traitement et sont obligés de communiquer par échange de messages afin de solliciter des services dans le but d'accomplir leur résultat souhaité. Le modèle d'architecture SOA représente un modèle d'architecture compatible avec la nature de la problématique et les objectifs visés par le projet services web **[Tme 04]**.

Le choix d'une architecture SOA entre dans la perspective de transformer le Web en une énorme plate-forme de composants faiblement couplés et automatiquement intégrables. C'est un modèle qui promet interopérabilité, flexibilité et réutilisabilité.

2.3.2. Architecture d'une application repartie selon SOA

2.3.2.1. Architecture classique (3 couches)

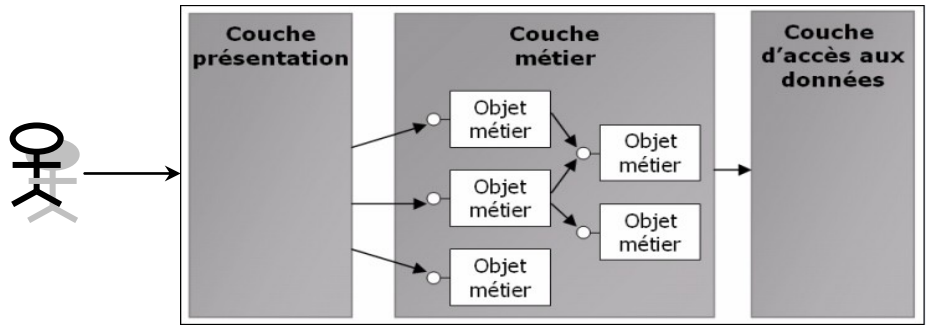

Figure 2. 2: Architecture trois couches classiques

- **Couche présentation :** Interface permettant l'interaction entre l'utilisateur et l'application, l'interface permet de manipuler et afficher les données en manipulant simplement les objets de la *couche métier*.
- **Couche métier :** cette couche contient toutes les règles de fonctionnement relatives à un métier. Par exemple pour une banque, la *couche métier* implémentera, entre autre, toutes les règles relatives à la gestion de comptes bancaires et de la clientèle.
- **La couche données :** cette couche modifie, récupère et assure la sécurité et l'intégrité des données.

Le code client présent dans la couche présentation est très lié aux objets de la *couche métier*, ce qui a pour conséquence que la plupart des modifications effectuées dans la couche métier ont des répercussions dans la couche présentation et requiert un nombre d'appels

important entre les deux couches. Ce nombre d'appels peut rapidement devenir problématique lors d'application répartie sur plusieurs sites distants.

2.3.2.2. Architecture orientée service

La figure 2.3 montre que la notion de service permet d'élever l'abstraction par rapport à la logique métier et ainsi augmente la possibilité de réutilisation et d'évolutivité des objets métier car *la couche présentation* n'est plus directement liée à la logique métier. Dans une architecture orientée service, la *couche présentation* ne manipule plus directement les objets métiers, mais passe par des services. Les objets métiers se trouvent dans des bibliothèques de classes directement chargées dans le même processus que les services, le coût des appels aux objets métiers est alors très faible.

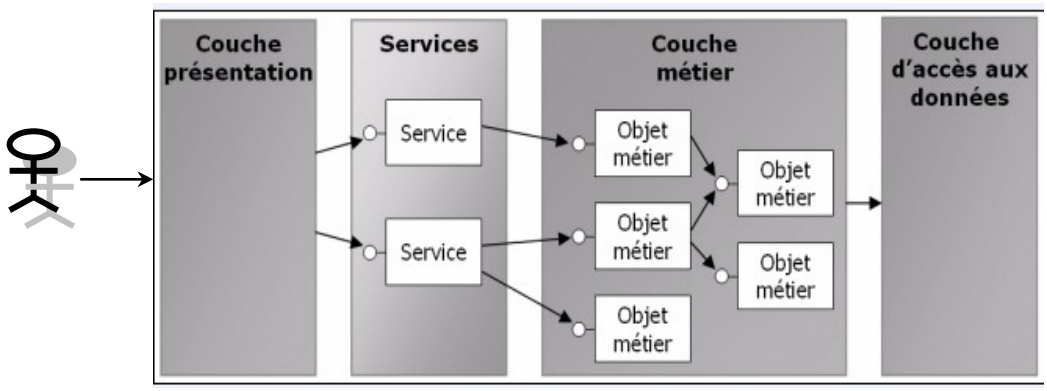

Figure 2. 3: Architecture orientée services

Les services agissent comme des boites noires faisant abstraction de la complexité du modèle objet, présentant un ensemble de fonctionnalités restreintes et permettant de réduire les échanges entre les couches. En résumé, lors du développement d'applications sous forme de services, la logique métier et la capacité d'isoler la *couche métier* de la *couche présentation* sont une priorité **[Pdu 04]**.

2.3.3. Composants d'une SOA

Une architecture de services est constituée de trois composants (voir figure 2.4). Le premier est le prestataire de services (le service réel). Vient ensuite le demandeur du service, autrement dit le composant qui accède au service. Enfin, l'agence de services fournit des services de découverte et d'enregistrement.

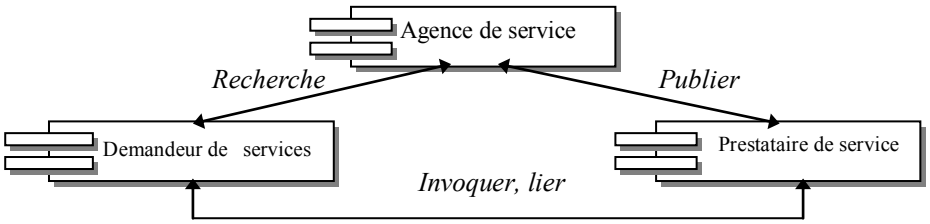

Figure 2. 4: Les éléments d'un service Web

- **Prestataire de services :** C'est la source de fonctionnalité des services. Un prestataire publie un contrat d'interface utilisé par les demandeurs pour accéder au service. Le contrat définit ce que fait le service et comment y accéder.
- **Demandeur de services :** C'est le client d'un service. Il utilise l'agence pour découvrir quels services sont disponibles. Une fois un service localisé, le demandeur extrait le contrat d'interface correspondant de l'agence. Le client utilise le contrat d'interface pour comprendre comment (méthodes disponibles) et où (adresse) accéder au service.
- **Agence de services :** C'est un registre des services disponibles. Chaque prestataire publie son contrat d'interface à l'agence avec les informations à utiliser pour localiser le service. Le demandeur recherche les services appropriés dans l'agence et extrait le contrat d'interface correspondant.

La présentation des composants SOA ci-dessus nous permet d'affirmer que les services Web implémentent l'architecture orientée service. Le prestataire de services est le fournisseur du service web, le demandeur de service est le client et l'agence de services est l'annuaire UDDI.

2.4. La technologie des Services Web

Le paradigme des services Web repose sur une architecture par composants qui utilise des protocoles Internet comme infrastructure pour gérer la communication entre les composants. Elle offre un modèle à base de composants (les services Web) en ligne encapsulant une application logique derrière une interface interactive uniforme et standardisée. Le terme «services Web» est utilisé pour désigner aussi bien le paradigme que l'idée d'un composant accessible à partir du Web.

2.4.1. Définitions

La définition des Services Web divise clairement la vision académique et industrielle. Certains restreignent les composants Services Web à ceux qu'utilisent les standards de la

technologie services Web tandis que d'autres proposent une définition plus générale. Pour cela, plusieurs définitions on été attribuées aux Services Web **[Tme 04]**.

- ✓ Pour **Duvanel [Pdu 04]**, "Les services Web sont des applications auto-descriptives, modulaires et faiblement couplées qui fournissent un modèle simple de programmation et de déploiement d'applications, basé sur des normes, et s'exécutant au travers de l'infrastructure Internet. Les services Web réalisent des fonctions allant des simples requêtes aux processus métiers sophistiqués."
- ✓ Pour **Vialette [Mvi 04]**, "Un Service Web est un composant implémenté dans n'importe quel langage, déployé sur n'importe quelle plate-forme et enveloppé dans une couche de standards dérivés du XML. Il doit pouvoir être découvert et invoqué dynamiquement par d'autres services."
- ✓ Pour **Curbera et al [Fcw 01]**, "Un Service Web est une application accessible à partir du Web. Il utilise les protocoles Internet pour communiquer et utilise un langage standard pour décrire son interface."

2.4.2. Infrastructure de base des Services Web

L'architecture de référence des services web (figure 2.5) s'articule autour des trois rôles suivants :

- **Le fournisseur de service :** correspond au propriétaire du service. D'un point de vue technique, il est constitué par la plate-forme d'accueil du service.
- **Le client :** correspond au demandeur de service. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service. L'application cliente peut être un navigateur web ou elle-même un service web.
- **L'annuaire des services :** correspond à un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

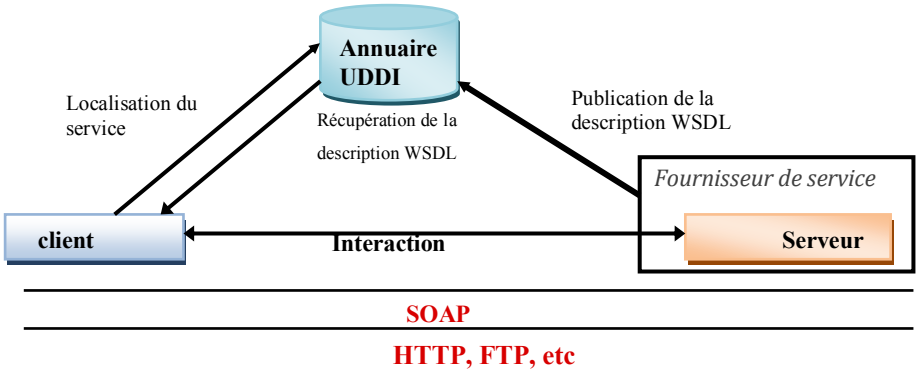

Figure 2. 5: Architecture de base des services Web

La définition des services web ne serait pas complète si on évoquerait pas ses principaux standards : SOAP (Simple Object Access protocol) **[Nmi 03]**, WSDL (Web Service Description Language) **[Ech 01]** et UDDI (Universal Description, Discovery and Integration) **[Udd 03]** mis en place par Microsoft, IBM, Sun…Comme on l'avait dit, le XML (eXtensible Markup Language) est à la base de tous ces standards. Le fait que les Services web utilisent ce méta-langage de structuration de données leurs procurent l'avantage d'être non-propriétaire et réellement multi-plateforme. Il est donc recommandé de posséder un minimum de bases (XML, DTD, XML Schema, XSL) afin de mettre en place des Services web réellement optimisés. **[Mvi 04].**

2.4.2.1. SOAP (Simple Object Access protocol)

2.4.2.1.1. Présentation

SOAP **[Nmi 03]** est un message basé sur le protocole de communication, qui peut être utilisé par les deux parties qui communiquent sur Internet. Les messages SOAP sont basés sur XML et sont donc indépendants de la plateforme. Ils forment la base de la pile du protocole de services Web. Ils sont transmis par HTTP. Donc, contrairement aux autres technologies comme le RPC[30] ou CORBA **[Cor 10]**, ces messages SOAP peuvent traverser un pare-feu. Les messages SOAP sont appropriés lorsque des petits messages sont envoyés. Lorsque la taille du message augmente, les frais généraux qui y sont associés augmentent aussi et par conséquent l'efficacité de la communication diminue.

Un message SOAP en cours de transit est composé de deux parties indépendantes :

- **Une structure XML qui constitue le message SOAP :** le message SOAP correspond à un document XML défini par une application cliente pour une application service suivant une structure particulière.

[30] RPC : Pemote Procedure Call

- **Une entête du protocole de transport :** le message SOAP est encapsulé dans un protocole de transport (http) en vue d'être livré à l'application destination. La figure 2.6 décrit la création, le transit et la consommation d'un message SOAP. La spécification SOAP est fondée sur trois parties : une syntaxe de structuration de message utilisant des techniques d'encodage, un modèle d'échange de messages et enfin un ensemble de modèles de transport basé sur les protocoles de transport applicatifs.

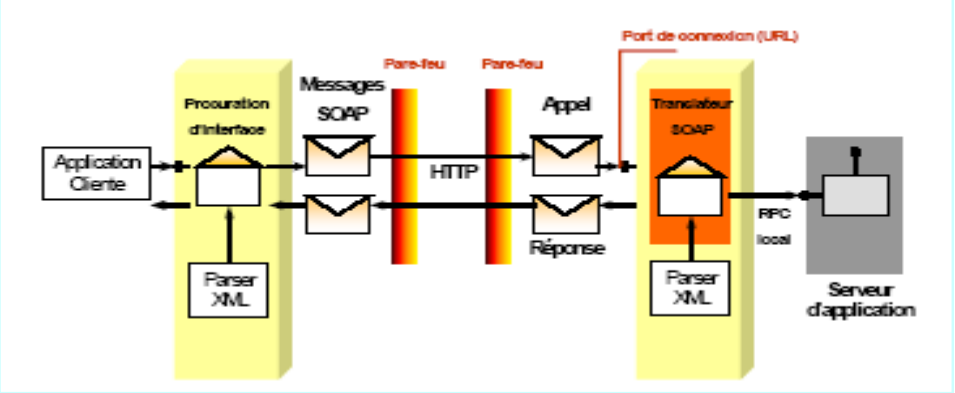

Figure 2. 6: Un échange type de message SOAP

2.4.2.1.2. Structure d'un message SOAP

Un message SOAP est un document XML - *Envelope* -, constitué de deux parties : un *Header* optionnel et un *Body* obligatoire.

```
<SOAP:Envelope xmlns:SOAP=
      "http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP:Header>
      <!– content of header goes here –>
   </SOAP:Header>
   <SOAP:Body>
      <!– content of body goes here –>
   </SOAP:Body>
</SOAP:Envelope>
```

Figure 2. 7: Structure d'un message SOAP

- **L'Envelope** est le premier élément du document XML, représentant le message. C'est l'élément racine du message. L'Envelope sert également d'introduction, de point de chargement des règles d'encodage *SOAP*. Ces règles sont utilisées pour indiquer la manière d'interpréter le message *SOAP*.

- **Le Header** sert à ajouter des caractéristiques supplémentaires au message *SOAP* (entrées non-applicatives : identifiant session, transaction, etc.). Le but principal d'un tel élément est de fournir des extensions au protocole, n'ayant pas directement de lien avec telle ou telle méthode spécifiée, mais apportant plutôt des informations contextuelles comme l'identifiant de transactions et/ou des informations relatives à la sécurité.

- **Le Body** est le conteneur de l'information délivré par le message (Nom d'une procédure, valeurs des paramètres, valeur de retour, code d'erreur). Le *Body SOAP* fournit un mécanisme simple pour l'échange d'informations obligatoires destinées au receveur final du message. Les utilisations typiques du *Body SOAP* sont :
  - Contenir des entrées applicatives : RPC.
  - Permettre l'encodage des entrées : déclaration d'objet, de valeur.
  - Reporter les erreurs (message retour).
- **Les erreurs SOAP :** Le fait d'utiliser SOAP ne garantit pas que les requêtes soient toujours couronnées de succès. Les choses peuvent mal se passer à différents moments du processus de traitement. Le serveur renvoie alors une réponse spécifiant un message d'erreur au sein d'un élément **<Fault>**, fils direct de <Body>:

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV=«http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV: Body>
<Fault>
<Faultcode>Server</faultcode>
<Faultstring>service:'urn: helloworld' unknown<falutstring>
<Faultactor>/soap/servlet/rpcrouter</faultactor>
</Fault>
</SOAP-ENV: Body>
</SOAP-ENV: Envelope>
```

Figure 2. 8: Exemple d'erreur SOAP

Actuellement, il existe des extensions de SOAP sur les protocoles de transport HTTP, SMTP ou encore FTP. Le transport sur HTTP c'est le protocole le plus utilisé pour le transport des messages SOAP. Le protocole HTTP constitue un excellent transport en raison de sa popularité sur le Web. Il existe deux règles à respecter pour pouvoir utiliser SOAP via HTTP :

1. Le mécanisme d'envoi doit utiliser la méthode HTTP POST standard.
2. La destination de la requête HTTP doit être adressée à une URI donnée sur le serveur Web.

Le bloc de données de la requête HTTP est composé du message SOAP lui-même et d'un attribut *SOAPAction*. Ce dernier est destiné à indiquer au serveur que le message HTTP transporte un message SOAP. La partie du code suivante représente un message SOAP en cours de transit sur HTTP. Le message est une requête (formatée RPC) pour une opération d'addition de deux entiers.

```
Content-Type :text/xml
Content-Length : ###
SOAPAction : "urn :mathTest""
<soap :Envelope xmlns :soap=" http ://schemas.xmlsoap.org/soap/envelope"
soap :encodinStyle=" http ://schemas.xmlsoap.org/soap/encoding">
<soap :body>
<math :add xmlns :math=" http ://exemple.com/math"/>
<math :op1> 25 </math :op1>
<math :op2>100</math :op2>
</math :add>
</soap :body>
</soap :Envelope>
```

Figure 2. 9: Message SOAP en cours de transit sur HTTP

Le protocole SOAP n'est pas concerné par la nature du programme cible ou le traitement des messages, il ne définit pas un environnement d'exécution mais plutôt un modèle de liaison qui assure l'interopérabilité au niveau applicatif entre des applications hétérogènes. La vision des services Web s'est concrétisée avec l'apparition de SOAP, en permettant à deux applications appartenant à des environnements technologiques hétérogènes d'échanger des données structurées.

2.4.2.2. WSDL (Web Service Description Language)

2.4.2.2.1. Présentation

Langage né des efforts d'IBM et de Microsoft, WSDL **[Ech 01] [Dsa 08]** est une note du W3C en mars 2001, dérivée de XML et qui décrit l'interface d'utilisation d'un service Web (méthodes et propriétés des composants de l'application), ses liaisons de protocoles (un service peut proposer une communication via SOAP sur un protocole de transport, via HTTP GET ou POST, ou via MIME[31]), et ses détails de déploiement. Le fichier WSDL peut être généré automatiquement par les boites à outils existant sur le marché.

2.4.2.2.2. Structure du WSDL

A partir de la description de WSDL pour un service Web (Figure 2.10), nous pouvons connaître les entrées, les sorties, les types des entrées et des sorties, l'ordre des entrées et des sorties, et comment le service Web devrait être appelé **[Fcu 02].** Comme le montre la figure 2.10, une description WSDL est un document XML dont l'élément racine est une *définition*. Chaque document définit un service comme une collection de points finals ou *ports*. Chaque *port* est associé à une liaison spécifique qui définit la manière avec laquelle les messages

[31] MIME : **Multipurpose Internet Mail Extensions** est un standard internet qui étend le format de données des courriels pour supporter des textes, des contenus non textuels, des contenus multiples, et des informations d'en-tête en d'autres codages que l'ASCII.

seront échangés. Chaque liaison établit une correspondance entre un protocole et un *type de port* **[Mel04].**

Un document WSDL décrit un service Web en utilisant les éléments majeurs suivants **[Dsa 08]**:

(a) portType - L'ensemble des opérations effectuées par le service web. Chaque opération est définie par un ensemble de messages d'entrée et de sortie.
(b) Message - Il représente les messages utilisés par le service Web. C'est une abstraction des données transmises.
(c) types - Il se réfère aux types de données définis pour décrire l'échange de messages.
(d) binding - Il précise le protocole de communication utilisé par le service Web.
(E) Port - Il définit l'adresse de liaison pour le service Web.
(f) service - Il est utilisé pour l'agrégation d'un ensemble de ports connexes.

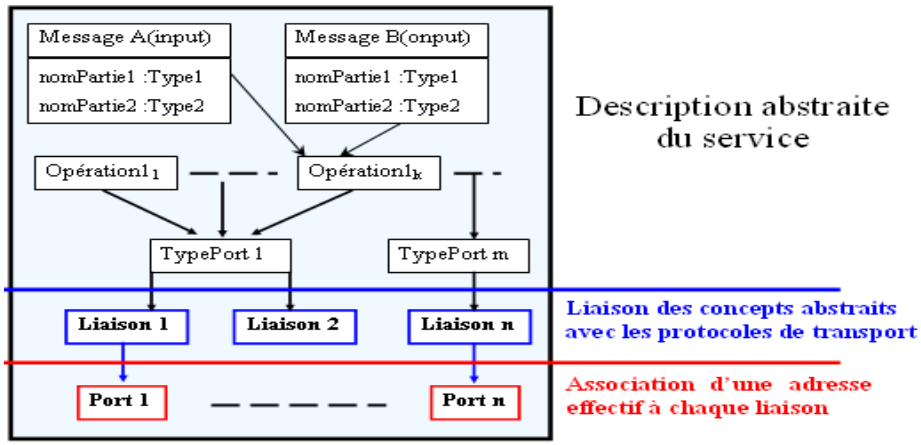

Figure 2. 10: Les éléments constitutifs du langage WSDL

2.4.2.3. UDDI (Universal Description Discovery and Integration)

2.4.2.3.1. Présentation

Défini initialement par Ariba, IBM et Microsoft, UDDI **[Udd 03]** est un protocole d'annuaire permettant de trouver le service Web recherché (de façon manuelle et automatique), mais aussi d'en annoncer la disponibilité.

Un annuaire UDDI est constitué de pages blanches (nom de l'entreprise, adresse, contacts), de pages jaunes (services classés par catégories industrielles) et de pages vertes (information d'implémentation des services Web proposés). Toute information saisie dans un annuaire est répliquée sur l'ensemble des annuaires. Concrètement, ces annuaires sont des fichiers XML hébergés par des entreprises appelées opérateurs UDDI ou nœuds d'opérateurs. **[Fcu 02]** Grâce à UDDI, les entreprises peuvent enregistrer des données les concernant, des renseignements sur les services qu'elles offrent et des informations techniques sur le mode

d'accès à ces services. L'UDDI normalise une solution d'annuaire distribué des services web, permettant à la fois la publication et l'exploration

2.4.2.3.2. Structure du registre UDDI :

En fait, le schéma sommaire de la structure du registre UDDI décrit la hiérarchie suivante **[Bli 03]** (Figure 2.11):

- ***businessEntity* :** Cette structure décrit l'entreprise, une ou plusieurs catégories auxquelles elle appartient (il existe plusieurs taxonomies), une description et l'adresse électronique des contacts qu'elle met à la disposition des clients pour assurer une assistance ou fournir des renseignements. Elle est en relation avec une ou plusieurs définitions des services ("businessService") fournis par l'entreprise. Les champs "categoryBag" et "identifierBag" permettent de distinguer les entreprises selon certains critères et permettent de faciliter la recherche dans le registre UDDI.
- ***businessService*** : Cette structure est décrite au moyen d'une entrée "serviceKey" identifiant le service publié par une entité d'affaires "businessEntity". Elle fait référence à une catégorie de service pour permettre une recherche en fonction d'un type de service particulier. Enfin elle pointe vers une liste de liens ("bindingTemplate") décrivant les différents accès au service.
- ***bindingTemplate*** : Une entreprise peut proposer ses services par différents moyens de communication ou protocoles. Pour chacune des méthodes, elle présente un enregistrement de type bindingTemplate. Ces derniers pointent vers une structures "tModel" et vers un point d'accès au service sur le web via http, FTP, SMTP…
- ***tModel*** : Cette dernière structure permet de mettre en relation les trois précédentes avec des classifications préétablies. C'est un index utilisé pour classer les "businessEntity", les "businessService" et les "bindingTemplate". Cet index permet de rechercher les services en rapport avec des schémas de classification (taxonomies) et pointe vers un URL dans lequel se trouve une description ou des descriptions particulières concernant un service.

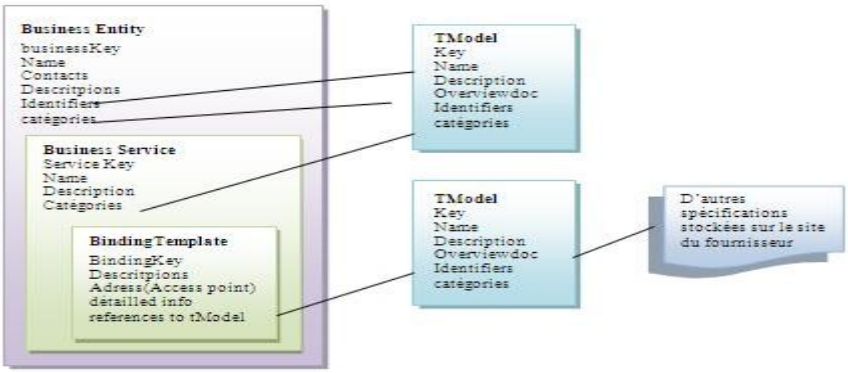

Figure 2. 11: Structure générale du registre UDDI

2.4.3. Fonctionnement des services Web

Pour expliquer le fonctionnement des services web (Figure 2.12), il convient de distinguer plusieurs étapes**:**

- **Recherche dans un annuaire UDDI :** le client cherche un service particulier, il s'adresse à un annuaire qui va lui fournir la liste des prestataires habilités à satisfaire sa demande. L'annuaire UDDI peut être comparé à un moteur de recherche sauf que les documents sont remplacés par des services.
- **Recherche de l'interface du composant à contacter :** une fois la réponse reçue (en XML) de l'annuaire, le client va chercher à communiquer via une interface. Cette interface décrite en langage WSDL fournit la description du service sélectionné.
- **Invocation du service :** le client doit passer maintenant les paramètres attendus par le service et assurer la communication avec le serveur.

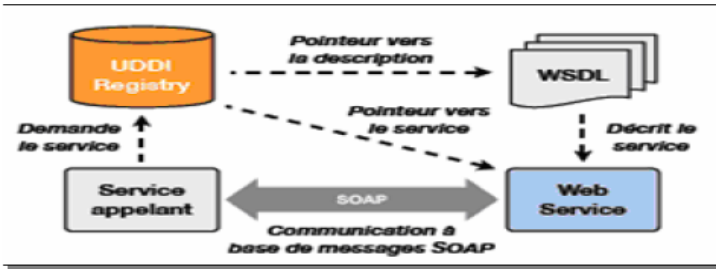

Figure 2. 12: Fonctionnement des services Web

2.5. Infrastructure étendue des Services web

L'infrastructure de base autour des standards SOAP, UDDI et WSDL répond aux problèmes d'interopérabilité et d'intégration technique des applications. Cependant, cette infrastructure n'est pas suffisante pour permettre une utilisation effective des services web dans les domaines dont les exigences vont au-delà de la capacité d'interactions simples via des protocoles standards.

Le W3C travaille activement à l'élaboration d'une architecture étendue standard. **[Dbo 04]** Cette architecture fourni un support pour la composition, la gestion des fonctionnalités, le suivie et le contrôle des services web pour une bonne exécution et coordination du service composite afin de garantir une certaine qualité de service.

L'architecture étendue est constituée de plusieurs couches se superposant les unes sur les autres, d'où le nom de **pile des services web [Dbo 04]**. La figure 2.13 décrit un exemple d'une telle pile. Chaque couche de la pile s'appuyant sur un standard particulier. On retrouve, au-dessus de la couche de transport, les trois couches formant l'infrastructure de base décrite précédemment. Ces couches s'appuient sur les standards SOAP, WSDL et UDDI.

On peut mettre en relief trois types de couches distinctes, décrites ci-dessous : **[Clo 04]**

- L'infrastructure de base (Discovery, Description, Exchange): C'est les fondements techniques sur lesquels est établis l'architecture de référence. Nous distinguons : les échanges de message (établis principalement par SOAP), la description de service (WSDL) et la recherche de services que les organisations souhaitent utiliser (via le registre UDDI).

- La couche Business Process: permet la composition des services Web. Elle établit la représentation d'un processus métier comme un ensemble de services web et décrit l'utilisation des différents services web composant ce processus.

- Les couches transversales (Security, Transactions, Administration, QoS) : Ces couches supportent les différents standards, intervenant dans l'ensemble du processus d'un service Web. Parmi ces dernières la gestion de la sécurité, l'administration, et la qualité de services web.

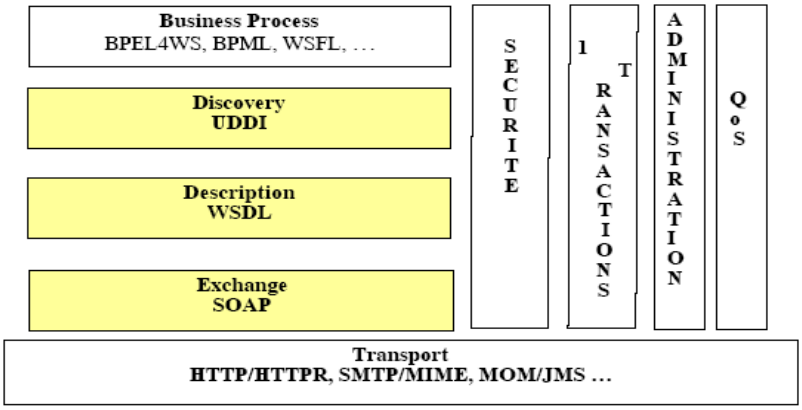

Figure 2. 13: Pile des services web

Des travaux tentent d'intégrer le web sémantique dans ces couches transversales, en ajoutant une couche verticale représentant le web sémantique et étant utilisable par les quatre couches horizontales représentant les standards. **[Dbo 04]**

2.6. Quelques outils et plateformes de développement des services web

a) Plates-formes commerciales : les logiciels de déploiement des services web sont des produits d'éditeurs de logiciels reconnus : Microsoft, IBM et Sun. Nous avons restreint les éditeurs au plus importants car le concept de service web atteint aujourd'hui de nombreux acteurs du monde informatique.

- Visual Studio .NET[32] de Microsoft
- Web Sphere Studio Application Developper[33] d'IBM
- Sun ONE developper Studio[34] de Sun

b) Plates-formes publiques : l'utilisation des services web étant en plein expansion l'éditeur SUN et la communauté de logiciels Apache mettent à la disposition d'utilisateurs potentiels des outils publics dont les plus importants sont :

- Java WSDP[35] (Web Service developper Pack) de Sun
- APACHE SOAP[36] de la famille APACHE
- AXIS[37] : APACHE eXtensible Intercation System
- Eclipse[38] avec Apache et Axis
- Netbeans[39] avec son serveur Glass Fish

Tous ces produits permettant le déploiement de l'intégralité des technologies de base des services web. De plus il existe différents éditeurs de services web, notamment Eclipse et Netbeans

2.7. Grid Services Vs Web services

Comme nous l'avons vu dans la section 1.12, les *grid services* représentent une extension des services web **[Ogs 08][Kta 09]**. Les services Web ont été choisis comme technologie sous-jacente pour les *grid services*, parce qu'ils sont des standards ouverts et ne nécessitent pas de plates-formes ou langages d'implémentation particuliers **[Kll 06]**. Cela dis la définition des *grid services* diffère d'un article à un autre. Y en a qui les définissent comme étant :

- Les services web définis, déployés et exécutés en utilisant les infrastructures de grille de calcul orientées services comme GT3.x, GT4.x, etc. **[Wem 05]**

[32] http://msdn.microsoft.com/en-us/library/fx6bk1f4%28v=vs.71%29.aspx
[33] http://www-01.ibm.com/software/integration/wsadie/
[34] http://developers.sun.com/jsenterprise/
[35] http://www.oracle.com/technetwork/java/webservicespack-jsp-140788.html
[36] http://ws.apache.org/soap/
[37] http://axis.apache.org/axis/
[38] http://www.eclipse.org/
[39] http://www.netbeans.org/

- Les ressources de calcul et de stockage qui sont exposées comme un ensemble extensible de services réseau et qui peuvent être regroupées pour créer des applications de fonctionnalité supérieure. **[Oez 07]**
- Les services basés sur la spécification WSRF. Ou les services web avec état (Stateful web services). **[Abo 10]**, **[Tdo 07]** et d'autres.
- Les services de la grille comme l'accès à la grille, la soumission des jobs, la gestion des données, … **[Ela 06]**

2.8. Conclusion

Le paradigme émergeant des services Web représente une nouvelle avancée dans l'ère de l'Internet. Les services Web s'inscrivent dans la continuité des technologies de mise en œuvre de l'informatique repartie. Leur atout majeur est de permettre l'interopérabilité entre les différentes applications en utilisant Internet comme infrastructure pour la communication. Ils sont caractérisés par leurs indépendances des plateformes et des systèmes d'exploitation, ce qui a impliqué leurs adoptions par les différentes organisations commerciales et industrielles.

L'approche "services web" est adoptée massivement comme la réponse appropriée aux problématiques d'échanges de données et d'intégration d'applications. Le développement et l'adoption des technologies associées aux services web permettent aux entreprises d'implanter de nouvelles applications en composant des services existants. Cette composition permet de combiner des services web élémentaires afin d'obtenir des services plus élaborés en se basant sur l'architecture étendue des services web.

La composition de services web représente une étape fondatrice de l'évolution de ce paradigme. La compréhension des propriétés fondamentales de la composition des services est nécessaire afin de tirer profit de ce paradigme.

Dans le chapitre suivant, nous allons nous intéresser particulièrement à la problématique de composition des services web et les différents formalismes et approches proposées dans ce cadre, en mettant en évidence l'impact de BPEL dans le domaine de composition de *grid services* pour la gridification des applications scientifiques.

CHAPITRE 3 : COMPOSITION DES SERVICES WEB & BPEL

3.1. Introduction

L'objectif de la notion de service est de promouvoir un accès simple et rapide aux fonctionnalités mises à disposition par les organisations. Aussi, la vision des services en tant que composants logiciels indépendants met en exergue les possibilités de coordination, ou composition, de plusieurs services pour fournir des fonctionnalités avancées. Afin de faciliter l'utilisation et la composition des services, les applications doivent réaliser des interactions faiblement couplées, c'est-à-dire qu'elles doivent être capables de fournir et utiliser des services tout en restant indépendantes les unes des autres, et sans divulguer leur fonctionnement interne. Cette exigence est satisfaite par l'adoption de protocoles et langages standardisés qui fournissent un accès uniforme aux services et à leurs descriptions.

Dans ce chapitre, nous présentons les définitions et les types de composition de services Web présents dans la littérature. Ensuite, nous étudions quelques langages de composition de services Web notamment le langage BPEL.

3.2. Définitions et types de composition de services web

Workflow est le terme utilisé pour définir la gestion des processus métier. Le workflow est un traitement automatisé qui organise et contrôle les tâches individuelles, les ressources et les règles nécessaires pour compléter un processus métier clairement défini. Les processus métiers sont un ensemble de procédures au sein d'une organisation, une séquence d'activités réalisées par différentes personnes (ou entités) vérifiant des *règles métiers* : les conditions d'enchaînement des processus élémentaires **[Dle 02]**.

La communauté industrielle suppose que les services web peuvent être intégrés comme des processus Workflow. IBM et Microsoft développent le concept de « Workflow de service web » **[Dle 02]** et propose chacun des outils logiciels supportant leurs nouveaux langages de gestion et d'orchestration des services web au sein d'un processus complexe. Dans le contexte des services web, un workflow définit comment les services participants (partenaires) travaillent ensemble dans un même processus pour exécuter une tâche précise. Il est appelé aussi par certains auteurs dans **[Hsa 05]** «Orchestration et Chorégraphie». La spécification d'un Workflow doit comprendre la description explicite de deux aspects complémentaires : «flux de contrôle» et «flux de données». Le flux de contrôle définit le séquencement des différentes activités dans un processus. Le flux de données définit comment les informations ou les données s'écoulent (circulent) entre les activités d'un même processus.

3.2.1. Définitions

La composition des services web en un processus connue sous le nom d'**Orchestration** a pour objectif de déterminer une combinaison de services en fonction d'une requête d'un client. Du coté du client, cette composition semblera un unique service. La composition sera transparente au client, même si cette composition sera la combinaison de plusieurs services web. Par ailleurs, une composition peut être composée de services atomiques ou composés (composites). Les services atomiques sont caractérisés pour avoir une unique fonctionnalité. D'un autre côté, les services composites peuvent regrouper plusieurs fonctionnalités, c'est-à-dire plusieurs services atomiques. En conséquence, une composition peut être soit composée de services composés, soit de services atomiques mélangés. Les services atomiques externalisés sont appelés les *services participants* ou *services partenaires* **[Nte 07].** Ces derniers peuvent appartenir à des organisations différentes, c'est pour cela, certains les appellent des services *externalisés (outsourced services)* **[Nte 07]**.

En d'autre terme, étant donnée une spécification de haut niveau des objectifs d'une tâche particulière, la composition de services implique la capacité de découvrir, de sélectionner, de composer et de faire interopérer des services existants en vue de créer un service composite offrant une nouvelle fonctionnalité satisfaisant l'objectif visé. **[Pke 04]**

Suite à l'étude de différents travaux sur la composition de services Web (**[Gal 04]**, **[Bbe 05]**, **[Dcl 06]**, **[Bsr 03]**, **[Jya 04]**) la définition de **[Bbe 05]** paraît la plus générale. Dans **[Bbe 05]**, les auteurs considèrent la composition de services Web comme étant un moyen efficace pour créer, exécuter, et maintenir des services qui dépendent d'autres services.

3.2.2. Quelques sources de complexité

La composition des Services web reste une tâche hautement complexe et pose un certain nombre de défis. Sa complexité provient généralement des sources suivantes :

- L'augmentation dramatique du nombre des services web sur le web rend très difficile la recherche et la sélection des services web pouvant répondre à un besoin donné. Cette caractéristique pose particulièrement le problème de scalabilité ;
- Les services web évoluent dans un environnement hautement dynamique et volatil, ils sont crées et mis à jour de façon irrégulière. Ainsi, une approche de composition doit tenir compte de ce comportement et doit détecter toutes les modifications survenues au moment de l'exécution et prendre des décisions avec toutes récentes informations. Cette

caractéristique pose le problème de disponibilité des services participants au moment de l'exécution du service composite ;

- Les services web sont d'habitude développés par différentes organisations qui utilisent différents modèles conceptuels pour décrire les caractéristiques des services web. Cela nécessite une technique d'interopérabilité sémantique pour vérifier leur compatibilité de composition et d'interaction en général **[Jra 04]**.

La modularité constitue une caractéristique importante des services Web, par conséquent, les services Web composites doivent être considérés récursivement comme des services Web et doivent garder les mêmes caractéristiques que les services Web basiques (WSDL) à savoir auto-descriptifs, interopérables, et facilement intégrables **[Tme 04]**.

3.2.3. Types de composition de services web

La plupart des travaux portant sur la composition de services Web reconnaissent deux types de composition : l'*orchestration* et la *chorégraphie* de services. **[Hsa 05]**

3.2.3.1. Orchestration

L'Orchestration décrit comment les services Web peuvent interagir les uns avec les autres au niveau du message, y compris la logique métier et l'ordre d'exécution des interactions. Ces interactions peuvent traverser des applications et/ou des organisations, et aboutir à un modèle de processus longlived (longue durée de vie), transactionnel et multi-étapes. **[Cph 03]**

Benattalah et al définissent l'orchestration comme un processus exécutable. Ils ajoutent que l'orchestration est un ensemble d'actions à réaliser par l'intermédiaire de services web. **[Bbe 05]**

L'orchestration peut être vue comme une composition ascendante : les services Web utilisés dans la composition existent au préalable et sont appelés selon un enchaînement prédéfini afin de réaliser un processus précis. La Figure 3.1 illustre l'orchestration. La requête du client (logiciel ou humain) est transmise au moteur d'exécution (*Moteur*). Ce dernier, d'après le processus préalablement défini, appelle les services Web (ici, *SW1*, *SW2*, *SW3* et *SW4*) selon l'ordre d'exécution.

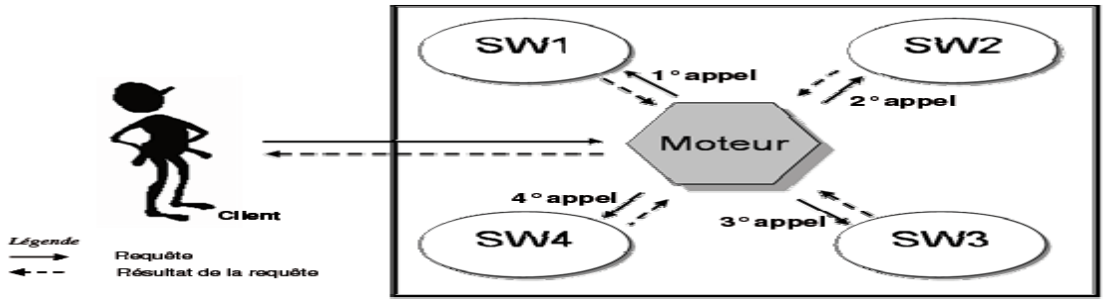

Figure 3. 1: Vue générale de l'orchestration

En d'autres termes, l'orchestration de services Web exige de définir l'enchaînement des services Web selon un canevas prédéfini, et de les exécuter selon un script d'orchestration. Ces derniers (canevas et script) décrivent les interactions entre services Web en identifiant les messages, et en spécifiant la logique et les séquences d'invocation. Le module exécutant le script d'orchestration de services Web est appelé un moteur d'orchestration. Ce moteur d'orchestration est une entité logicielle qui joue le rôle d'intermédiaire entre les services en les appelants suivant le script d'orchestration.

3.2.3.2. Chorégraphie

La Chorégraphie trace (piste) la séquence de messages qui peuvent impliquer de nombreuses parties et de multiples sources, y compris les clients, fournisseurs et partenaires. La chorégraphie est généralement associée à l'échange de messages publics qui se produit entre les services Web multiples, plutôt que d'un processus métier spécifique qui est exécuté par un seul parti. **[Cph 03]**

La chorégraphie est aussi appelée composition dynamique **[Cpe 03]**. En effet, l'exécution n'est pas régie de manière statique comme dans une composition de type orchestration. Dans une chorégraphie, à chaque pas de l'exécution (*i.e.* à chaque étape de la composition), un service Web choisit le service Web qui lui succède et implémente ainsi une partie de la chorégraphie. La composition de type chorégraphie n'est pas connue, ni décrite à l'avance.

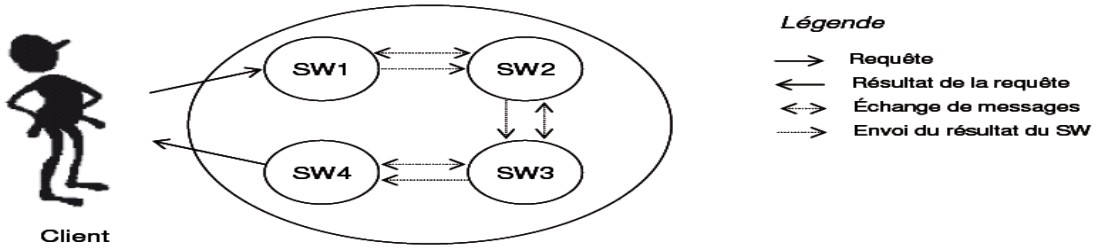

Figure 3. 2: Vue générale de l'exécution d'une composition de services Web de type chorégraphie

La Figure 3.2 donne une vue générale d'une composition de services Web de type chorégraphie. Le client (logiciel ou humain) établit une requête qui est satisfaite par l'exécution automatique de quatre services Web (SW1, SW2, SW3 et SW4). La requête de l'utilisateur est transmise au premier service Web (SW1) qui est exécuté. Le SW1 découvre ensuite le service Web lui succédant. Le processus de découverte repose, selon les cas, soit sur une recherche dans un registre local ou public, soit sur une découverte globale sur le Web à l'aide d'ontologies. Une fois le service découvert, les deux services (SW1 et SW2) échangent des messages afin de vérifier si leur communication est viable dans le cadre de la requête. Si les échanges de messages sont concluants, le résultat de l'action du SW1 est transmis au SW2 qui l'utilise comme paramètre d'entrée. Le processus d'implémentation de la composition est identique pour chaque étape (SW2 et SW3). Le SW4 termine le processus et le résultat de son action est transmis au client.

3.3. Avantage de la composition des services web

Du point de vue de la composition des services Web pour exécuter des processus métier, l'orchestration a un avantage sur la chorégraphie. L'orchestration est un paradigme plus souple, bien que la ligne entre l'orchestration et la chorégraphie est en train de disparaître. L'Orchestration présente les avantages suivants:

- Nous savons exactement qui est responsable de l'exécution de l'ensemble du processus métier.
- Nous pouvons intégrer des services Web, même ceux qui ne sont pas conscients qu'ils font partie d'un processus métier.
- Nous pouvons également fournir des scénarios alternatifs en cas de panne.
- La logique de la composition de services web est modulaire et est séparé du reste de l'application. Cette logique peut être réutilisée par différentes applications.
- Toute modification de la logique métier peut être faite directement dans le module sans modifier les autres parties de l'application.

3.4. Langages de composition de services Web

De nombreux industriels (tels qu'IBM ou Microsoft) et consortium (tel que le W3C) travaillent afin de mettre en œuvre un langage de composition de services Web standard (tel que WSCI – *Web Service Choreography Integration*) **[Aar 02]**. Dans cette section, nous étudions les langages qui sont soit largement utilisés dans l'industrie (BPEL4WS **[Tan 03]**),

soit en cours de standardisation (WS-CDL **[Nka 05]** et OWL-S **[Dma 04]**). Nous allons mettre l'accent sur BPEL **[Tan 03]**.

3.4.1. WS-CDL (Web Service Choreography Description Language)

WS-CDL **[Nka 05]** est un langage issu des efforts de standardisation du groupe de travail du W3C portant sur la chorégraphie de services Web (*Web Services Choreography Working Group*). L'objectif de ce langage est de décrire les relations entre les services Web lors d'une composition de type chorégraphie. WS-CDL, à l'instar des standards de services Web, est basé sur XML. Il complète la description WSDL des services Web afin de décrire les interactions entre les participants (les autres services) de la composition. Cette description est englobé dans l'élément Package (Figure 3.3).

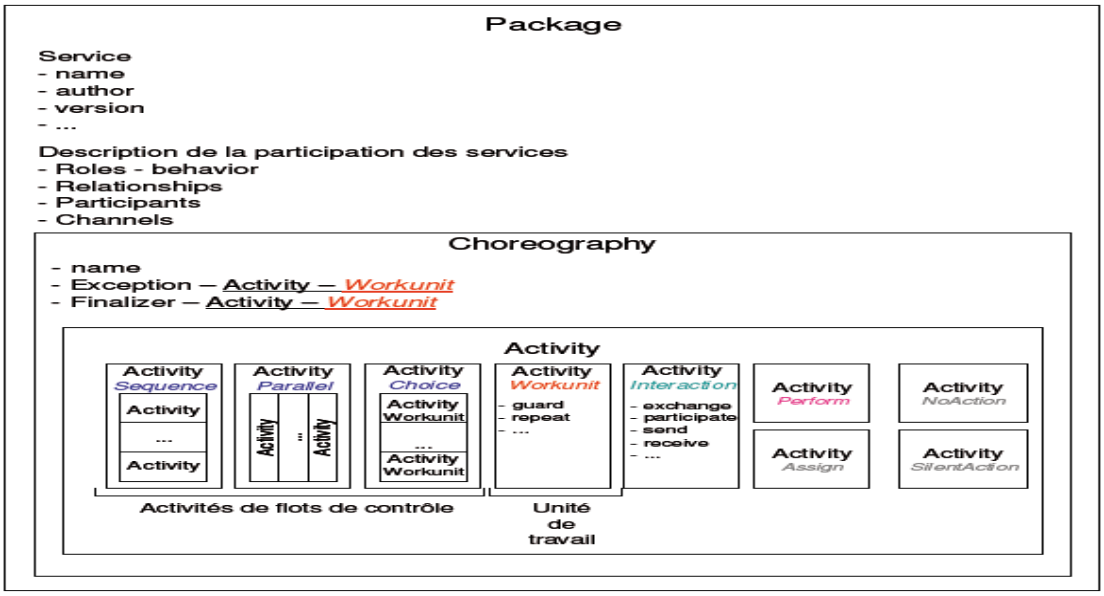

Figure 3. 3: Représentation du package de WS-CDL, d'après [Aba 05]

3.4.2. OWL-S (Web Ontology Language for Web Services)

OWL-S **[Dma 04]** est un langage de mise en œuvre de services Web sémantiques qui permet la description, la découverte, l'invocation et la composition de type chorégraphie des services Web. Une description OWL-S d'un service Web sémantique est composée de trois documents OWL : le profil du service (*Service Profile*), le modèle du service (*Service Model*) et l'accès au service (*Service Grounding*).

Le modèle du service présente le fonctionnement du service et définit les compositions de services Web (processus) dans lesquelles les services Web interviennent. Ces interventions sont décrites à l'aide du modèle de processus (*Process Model*). Le *Process Model* permet de décrire la tâche que propose le service web au sein de la composition de services. Le modèle

de processus est représenté par la classe *ProcessModel*. Cette dernière possède les propriétés suivantes : les paramètres d'entrée et de sortie, les participants au processus (les autres services web intervenant dans la composition), les pré-conditions et les actions du service décrit. Il existe trois types de processus : le processus atomique, simple et composé (Figure 3.4).

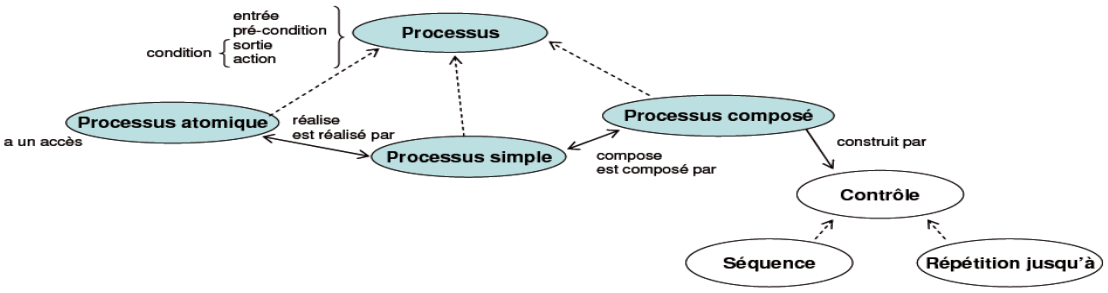

Figure 3. 4: Description des différents types de processus, d'après [Dma 04]

- **Le processus atomique (*Atomic Process*)**. Ce processus est directement invoqué par l'intermédiaire d'un accès.
- **Le processus simple (*Simple Process*)**. Le processus simple n'est pas directement invoqué. Il fournit une vue d'un processus atomique ou la représentation simplifiée d'un processus composé.
- **Le processus composé (*Composite Process*)**. Ce dernier est composé de processus simples. Il utilise des commandes de contrôle (*Control Construct*) afin de gérer l'invocation des processus qui le composent.

OWL-S permet de décrire les compositions de type chorégraphie. Puisqu'il utilise les langages RDF et OWL, il permet une description sémantique des services web intervenant dans la composition. L'inconvénient majeur d'OWL-S est que la description d'un service web est complexe étant donné qu'un service Web doit contenir autant de descriptions de processus que de compositions auxquelles il participe.

3.4.3. BPEL (Business Process Execution Language)

3.4.3.1. Définition

L'adoption générale des solutions d'automatisation des processus exige une base standard et un langage spécialisé pour la composition des services dans les processus métier qui offre la possibilité d'exprimer ces processus d'une manière standardisée, en utilisant un langage communément admis. BPEL **[Tan 03]** est un langage et est rapidement devenu la

norme dominante. L'objectif principal de BPEL est de normaliser les processus d'automatisation entre les services web.

Ces dernières années, il ya eu une acceptation croissante de BPEL comme un langage standard pour l'orchestration des services Web, l'une des implémentations les plus réussis et bien développée de SOA. BEA[40], IBM, SAP[41], Siebel Systems[42] et Microsoft ont uni leurs efforts afin de produire un langage de composition de services Web, conçu pour supporter les processus métier à travers les services Web **[Dle 02]**.

BPEL, connu aussi sous le nom de BPEL4WS (Business Process Execution Language for Web Services) ou WS-BPEL **[Tan 03]**, est issu de la fusion de deux langages : WSFL – *Web Service Flow Language* **[Fle 01]** d'IBM fondé sur la notion de graphes orientés et XLANG **[Sth 01]** de Microsoft qui est un langage structuré en blocs. BPEL **[Tan 03]** est basé sur XML et sur les *workflows*. Sa première version (BPEL4WS 1.0) a été réalisée en Août 2002. Ce langage distingue les processus abstraits des processus exécutables comme le montre la figure 3.5.

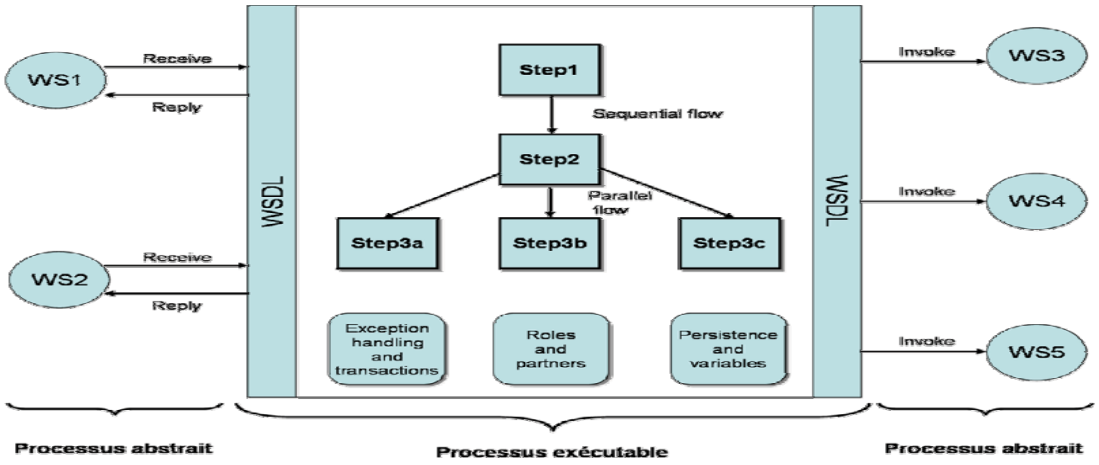

Figure 3. 5: Le flot de processus avec BPEL, d'après [Cpe 03]

- **Le processus abstrait**. Ce processus spécifie les messages échangés entre les différents participants (*services web composants*) sans indiquer le comportement de chacun d'eux. **Wynen** dans **[Fwy 03]** parle de *Business Protocol*, c'est-à-dire la spécification du comportement des partenaires par rapport aux messages échangés, sans rendre public le comportement interne. Ce processus abstrait peut être relié à une composition de type chorégraphie. Les services web communiquent alors à l'aide d'échanges de messages (partie gauche de la Figure 3.5).

[40] http://www.oracle.com/us/corporate/Acquisitions/bea/index.html
[41] http://www.sdn.sap.com/irj/sdn/index?rid=/webcontent/uuid/b4313088-0901-0010-3d97-9566d764c212
[42] http://www.oracle.com/us/products/applications/siebel/index.html

- **Le processus exécutable**. Ce processus permet de spécifier l'ordre d'exécution des activités, le partenaire concerné, les messages échangés entre ces partenaires, et les mécanismes des erreurs et des exceptions. En d'autres termes, il s'agit du moteur de l'orchestration donnant une représentation indépendante des interactions entre les partenaires.

La Figure 3.5 illustre la mise en œuvre des deux types de processus (exécutable et abstrait) par BPEL. La définition du processus métier est donnée par le processus exécutable. Les processus abstraits gèrent les invocations entre les différents services web permettant l'exécution de la composition de services définie dans le processus exécutable.

Trois éléments permettent à BPEL de gérer le flot de processus dans le processus exécutable : les transactions (*Exception handling and transactions*), les partenaires (*Roles and Partners*) et les espaces de stockage (*Persistance and Containers*) **[Clv 08]** (Figure 3.5).

a) **Les transactions** : sont utilisées dans BPEL pour gérer les erreurs et les appels d'autres services si le service appelé est indisponible ou défaillant.
b) **Les partenaires** : sont différents services Web invoqués dans le processus. Ils ont chacun un rôle spécifique dans un processus donné. Chaque partenaire est décrit par son nom, son rôle (en tant que service indépendant), et son rôle dans le processus. Le fait de décrire différents niveaux de rôle permet à chaque partenaire d'avoir une vie indépendante des compositions dans lesquelles il intervient.
c) **Les espaces de stockage** : permettent la transmission des données. Le flot de processus BPEL permet que ces données soient cohérentes à travers les messages échangés entre les services Web. Un message peut être un message d'appel (*invoke*), de réponse (*reply*) ou d'attente (*receive*).

La structure des différentes activités peut être, soit séquentielle (*Sequential flow*), soit parallèle (*Parallel flow*). Si l'activité est parallèle, alors plusieurs services peuvent être invoqués en même temps.

BPEL est le premier langage de composition de services web adopté par la communauté des services web. Ceci est principalement dû au fait que ce langage possède une grande expressivité dans la définition des processus.

Le développement d'un processus BPEL nécessite une bonne compréhension de WSDL et d'autres technologies connexes. BPEL introduit les extensions WSDL, qui nous permettent de spécifier avec précision les relations entre les différents services web dans les processus.

Ces relations sont appelées ***Partner Links***. La figure 3.6 montre un processus BEPL et ses relations avec des services web (partner links) **[Mbj 06]**:

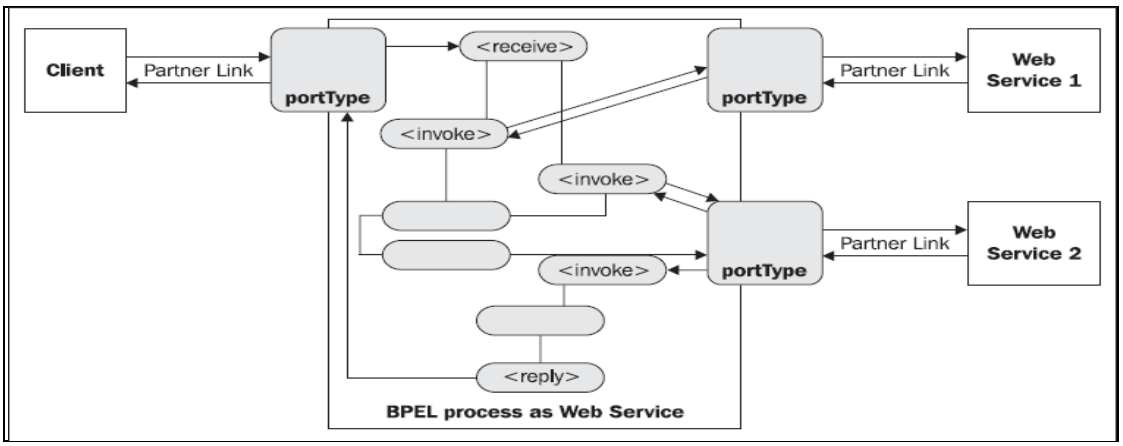

Figure 3. 6: Processus Exécutable BPEL et ses Partners Links [Mbj 06]

3.4.3.2. Caractéristiques de BPEL

Avec BPEL, nous pouvons définir les processus métier simples et complexes. Dans une certaine mesure BPEL est similaire aux langages de programmation traditionnels mais il est spécialisé axé sur la définition des processus métier. Par conséquent, d'une part, il offre des constructions tels que des boucles, des branches, variables, affectations, etc., ce qui rend la définition des processus relativement simple. D'autre part, il est moins complexe que les langages de programmation traditionnels, ce qui simplifie l'apprentissage.

Les constructions les plus importantes de BPEL sont liées à l'invocation de services Web. Avec BPEL, nous pouvons:

- Décrire la logique des processus métier à travers la composition de services ;
- Composer de grands processus métier à partir de petits processus et services ;
- Gérer les invocations d'opération synchrones et asynchrones (souvent de longue durée) sur les services, et gérer les rappels (callback) qui se produisent à des moments plus tard ;
- Invoquer les opérations de service en séquence ou en parallèle ;
- Compenser avec sélectivité les activités complétées en cas d'échec. BPEL fournit un vocabulaire riche pour le traitement des erreurs, car c'est très important qu'un processus métier robuste puisse réagir à des échecs de manière intelligente ;
- Maintenir de multiples activités transactionnelles de longue durée, qui sont également interruptible ;
- Reprendre les activités interrompues ou échouées pour ne pas refaire tout le travail déjà fait ;
- Acheminer les messages entrants aux processus et activités appropriés ;
- Corréler les requêtes à l'intérieur et à travers des processus métier ;

- Planifier les activités basées sur le temps d'exécution et définir leur ordre d'exécution ;
- Exécuter les activités en parallèle et définir la synchronisation entre les différents flux qui s'exécutent en parallèle ;
- Structurer les processus dans plusieurs étendues (champs d'application) ;
- Gérer les événements liés au message et aux temps. BPEL fournit un appui pour les processus de longue durée et de rémunération.

3.4.3.3. Relation de BPEL avec d'autres langages

BPEL n'est pas le seul langage pour la gestion des processus métier et de la modélisation. Avant de commencer à discuter des aspects techniques de BPEL nous allons donner un aperçu de la relation de BPEL avec d'autres langages. Récemment, plusieurs langages ont été proposés, y compris: **[Mbj 06]**

- XLANG et la nouvelle version XLANG/s de Microsoft.
- BPML (Business Process Modeling Language) de BPMI.org, le Business Process Management Initiative
- WSFL (Web Services Flow Language) d'IBM
- WSCL (Web Services Conversation Language) de HP, soumis au W3C
- BPSS (Business Process Specification Schema), s'inscrit dans le cadre d'ebXML
- WSCI (Web Services Choreography Interface), co-développé par Sun, SAP, BEA, et Intalio[43] et soumis au W3C
- WS-CDL (Web Service Choreography Description Language), au moment de la rédaction d'un projet du W3C (section 3.4.1)

La figure 3.7 présente la chronologie des langages mentionnés :

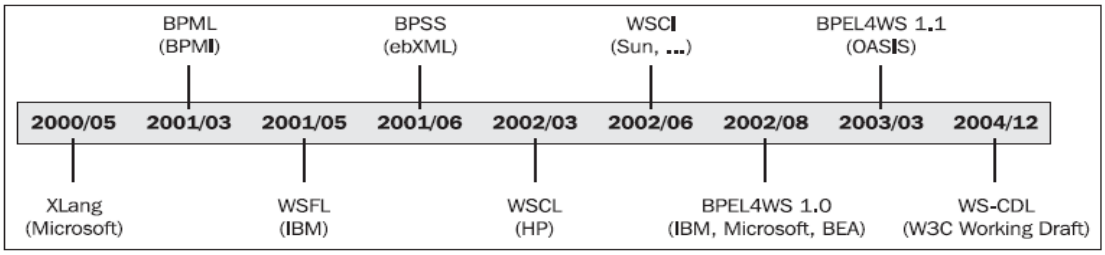

Figure 3. 7: Chronologie des langages dans l'ordre de leur apparition [Mbj 06]

3.4.3.3.1. ebXML BPSS

ebXML (Electronic Business XML) **[Jic 01]** est un Framework qui fournit un ensemble de technologies, BPSS (Business Process Specification Schema) étant l'un d'entre eux. ebXML a été élaboré à l'initiative d'OASIS et UN/CEFACT[44].

[43] http://www.intalio.com
[44] http://www.unece.org/cefact/

BPSS couvre le même domaine que BPEL. L'approche BPSS par rapport à la spécification processus suit le modèle chorégraphie et est donc comparable au processus abstrait BPEL. En plus de préciser la logique du processus, BPSS précise également les détails du protocole de communication. BPSS n'est pas une alternative directe aux BPEL, il est utilisé dans des environnements où ebXML est appliquée.

3.4.3.3.2. BPML

BPML (Business Process Markup Language) a été développé par BPMI.org (Business Process Management Initiative) **[Bpi 05]**. Intalio a joué un rôle important, et a été l'initiateur de BPML. BPML est un méta-langage pour la modélisation des processus métier et fournit un modèle d'exécution abstrait pour décrire les collaborations et les transactions. Il définit un modèle formel pour l'expression des processus abstraits et exécutable, et prend en charge la gestion des données, la conformité, la gestion des exceptions et les opérations sémantiques.

La comparaison de BPML avec BPEL montre que les deux partagent des bases similaires dans les services web et exploitent d'autres spécifications techniques des services web, notamment WS-Security, WS-Coordination et WS-Transactions. Cependant, BPML supporte la modélisation de processus métiers plus complexes grâce à son soutien pour la sémantique avancée tels que les processus imbriqués et les opérations compensées complexes. BPML peut donc être considérée comme un sur-ensemble du langage BPEL. Les extensions de BPEL des règles métier, la gestion des tâches, les interactions humaines, etc., sont définis dans BPXL (Business Process eXtension Layers). Le fait que BPEL et BPML partage les mêmes idiomes et ont une syntaxe similaire peut être une base pour une convergence future possible.

3.4.3.3.3. WSCI

WSCI (Web Services Choreography Interface) version 1.0 **[Aar 02]** a été développé par Sun, BEA, SAP, et Intalio. WSCI est un langage pour décrire les flux de messages échangés par des services Web dans le cadre d'un processus. Il nous permet de décrire le comportement observable d'un service Web dans un échange de messages. WSCI décrit également l'échange de message collectif entre les services web interactifs, offrant une vue globale et orientée message d'un processus impliquant de multiples services web.

En WSCI, l'échange de message est décrit du point de vue de chaque service web. C.-à-d., WSCI décrit le comportement observable des services web. Toutefois, WSCI ne traite pas

la définition du processus entraînant l'échange de messages. Il n'aborde pas non plus la définition du comportement interne de chaque service web.

WSCI suit le modèle chorégraphie et ne traite pas la définition des processus opérationnels exécutables, il est comparé directement au processus abstrait de BPEL. WSCI n'a pas obtenu le soutien de l'industrie comparé à BPEL et la seule entreprise qui a fourni un support des outils est Sun avec l'éditeur SunONE WSCI. Le consensus de l'industrie semble appuyer BPEL.

3.4.3.3.4. WS-CDL

Comme nous l'avons vu dans la section 3.4.1, WS-CDL **[Nka 05]** est un langage permettant de spécifier la chorégraphie de collaboration avec les services. Comme WS-CDL est un langage complémentaire à BPEL nous ne pouvons pas faire une comparaison directe. Toutefois, WS-CDL diffère considérablement de BPEL. Avec WS-CDL, nous définissons les flux de messages échangés par tous les partenaires, tandis qu'avec BPEL, nous nous concentrons sur les flux de messages et le comportement d'un partenaire spécifique, c'est-sur le comportement interne d'un processus métier. La description WS-CDL des flux de message est effectuée à partir d'un point de vue général, tandis que BPEL spécifie l'échange de messages à partir du point de vue d'un partenaire spécifique. Un processus BPEL précise les activités qui sont exécutées. WS-CDL spécifie les règles réactives, qui sont utilisés par tous les participants d'une collaboration.

3.4.3.4. Les serveurs BPEL (BPEL Engine)

Plusieurs serveurs commerciaux et open source existent sur le marché. Dans l'Annexe B nous citons les plus importants. Nous présentons dans le tableau suivant un comparatif de quelques serveurs BPEL.

Produit	Vendeur	Edition	Date de Réalisation	Framework	Compatibilité	License
ActiveVOS	Active Endpoints	8.0	Septembre 2010	Servlet ou Java EE	BPMN 2.0; WS-BPEL; BPEL4People /WS-HumanTask; standards	Propriétaire
Apache ODE	ASF (donated by Intalio)	1.3.4 1.0.164	9 Juin 2010 7 Juin 2006	Apache Axis, JBI Java EE	BPEL4 WS 1.1, WS-BPEL 2.0	Apache
BizTalk Server	Microsoft	Biztalk 2010	2010	.NET	BPEL, BPMN, RFID, WSDL, ...	Propriétaire
iBolt Server	Magic Software Enterprises			Java EE	BPEL4WS	Propriétaire

jBPM	jBoss	3.3.1	12 Janvier 2009	Java EE	WS-BPEL	LGPL
Open ESB	Oracle Corporation	2.0	10 Février 2009	Java EE, JBI	WS-BPEL 2.0	Open Source, CDDL
Oracle BPEL Process Manager	Oracle Corporation	11g	Avril 2010	Java EE	WS-BPEL 2.0, BPMN	Propriétaire
OW2 Orchestra	OW2	4.7	2004-2009	Apache Axis Apache CXF OSGi Java EE	WS-BPEL 2.0	LGPL
Parasoft o	Parasoft BPEL Maestr	5.0.1	25 Février 2010	Servlet	WS-BPEL, BPEL4People / WS-HumanTask	Propriétaire
Petals BPEL Engine	Petals Link	1.0.1	08 Décembre 2009	Java EE	WS-BPEL 2.0, WSDL 1.1 and 2.0	LGPL
SAP Exchange Infrastructure	SAP AG	3.0			BPEL	Propriétaire
Virtuoso Universal Server	OpenLink Software	4.5	2006		UDDI, WS-BPEL, WS-*	GPL et Propriétaire
WebSphere Process Server	IBM	6.0.1.3	29 Séptembre 2006	Java EE	WS-BPEL	Propriétaire

Tableau 2: Quelques serveurs BPEL

3.5. BPEL et les Grilles de calcul

Les applications scientifiques exécutées avec les technologies modernes distribuées ont tendance à être calculées sur diverses ressources dispersées, et les workflows sont devenus une méthode naturelle pour les décrire. Plusieurs travaux se sont mis sur l'utilisation de BPEL comme le moyen de composition des services de grille (*grid services*).

Koon et al [Kll 06] expliquent comment BPEL peut être utilisé pour supporter l'orchestration des *grid services*. Les auteurs ont utilisé ActiveBPEL comme moteur d'orchestration et Globus Toolkit4 comme conteneur de *grid services*. Ils ont expliqué comment CRESS (Chisel Representation Employing Systematic Specification) a été étendu pour décrire la composition des *grid services*.

Ivan et al [Ija 08], décrivent un moteur de contrôle de workflow (WEEP: Workflow Enactment Engine Project) pour l'orchestration des *grid services* utilisant le langage WS-BPEL 2.0. Ils estiment que BPEL est le langage approprié pour l'implémentation de processus d'analyse scientifique qui se composent de plusieurs phases telles que l'extraction de données, la transformation de données, l'intégration de données, le data mining, l'évaluation et la visualisation des résultats, et qui peuvent être implémentés comme des *grid*

services autonomes ou composés. Ces services peuvent être combinés pour former un workflow complexe de traitement de données interactive.

Dans **[Oez 07]**, **Onyeka et al** montrent l'utilisation de BPEL pour intégrer, créer et gérer les ressources WS qui implémentent le modèle fabrique/instance. Ils expliquent comment BPEL peut être utilisé pour composer les *grid services* basés sur WSRF. Ils ont donné l'exemple de services qui ont été composés pour créer une application bioinformatique qui est utilisée pour détecter des motifs Helix-Turn-Helix dans des séquences de protéines.

Andrea et al proposent **dans [Abo 10]** des services Web spécialisés, qu'ils nomment BPEL Partners, qui prennent en charge la découverte des ressources adéquates, le monitoring de l'exécution de chaque tâche du workflow dans les ressources et l'agrégation des résultats de l'exécution. Ils ont implémenté des *grid services* basés sur WSRF pour la bioinformatique, aussi pour détecter des motifs Helix-Turn-Helix dans des séquences de protéines. Ces *grid services* ont été déployé sur Globus Toolkit 4 et le processus BPEL sur ActiveBPEL 3.0.

Dörnemann et al dans **[Tdo 07]** ont discuté les problèmes de composition de *Grid services* avec BPEL dans sa version 1.1 et particulièrement les services basés sur WSRF .Ils ont présenté des extensions de BPEL avec de nouvelles activités pour l'invocation des *grid services*. Pour cela ils ont développé dans d'autres travaux une application de designer BPEL basée sur Eclipse. Ils ont utilisé ActiveBPEL comme conteneur de processus.

Gobe et al dans **[Pjg 07]** ont fait une comparaison entre les services web OGC[45] et OGSA, ils ont abordé les problèmes de non-interopérabilité de ces services. Ils présentent une approche de développement de workflows de web services OGC (services web géo-spatiales) exécutable sur la grille. Ils ont utilisé ActiveBPEL et Globus Toolkit 4. Ils déduisent que les workflows basés sur BPEL peuvent supporter les services web géo-spatiales exécutables sur la grille[46].

L'objectif de **Tino et al** Dans **[Tfl 08]**, est de construire un moteur de workflow géo-spatiale qui supporte la combinaison des services Web classiques (basés sur XML), des services Web OGC standards et des services Web OGC exécutables sur la grille (basés sur WSRF) avec le langage BPEL. Le prototype implémenté utilise la solution de **[Pjg 07]**, il est basé sur ActiveBPEL et Globus Toolkit 4.

[45] OGC : Open Geospatial Consortium
[46] Exécutable sur la grille est la traduction libre de grid-enabled

Wolfgang et al dans **[Wem 05]** décrivent leur expérience dans l'orchestration des workflows scientifiques en utilisant BPEL. Ils montrent une étude de cas dans le domaine de la chimie théorique, pour orchestrer des *grid services* pour l'automatisation d'une application de prévision polymorphe. Ils expliquent comment BPEL est utilisé dans l'orchestration des *grid services* et que BPEL peut être utilisé de manière plus générale pour l'orchestration de *grid services*. Ils ont utilisé ActiveBPEL et OMII[47].

3.6. Conclusion

La composition de services Web est un ensemble d'interactions entre services Web répondant à un problème complexe, formalisé par un client (logiciel ou humain) à l'aide d'une requête. Ces interactions (définies comme des échanges de données gérés par un flot de contrôle), ont pour but de réaliser un processus préalablement défini par les concepteurs. Une composition de services peut être de deux types : orchestration ou chorégraphie.

De nombreuses organisations (industrielles, telles qu'IBM et Microsoft, de recherche, telles que le W3C) travaillent afin de standardiser leur langage de composition de services Web. Dans ce chapitre, nous avons étudié quelques langages de composition : BPEL4WS, WS-CDL et OWL-S.

BPEL est la technologie clé dans les environnements où les fonctionnalités sont déjà ou seront exposés via des services Web. Avec l'augmentation de l'utilisation des technologies de services Web, l'importance de BPEL va encore augmenter

Le besoin d'orchestration de Grid/Web services prend de plus en plus d'ampleur dans le domaine de grille de calcul pour les applications scientifiques. On trouve un nombre important de travaux dans ce contexte qui utilisent BPEL comme langage d'orchestration. Nous avons cité quelques uns et il y en a d'autres.

[47] OMII : Open Middleware Infrastructure Institute for Europe. http://www.omii-europe.org/

CHAPITRE 4 : MODÈLE DE GRIDIFICATION PROPOSÉ & PROTOTYPE

4.1. Introduction

De nombreux chercheurs en sciences collaborent et effectuent souvent des analyses sur de grands ensembles de données. Il y a de plus en plus de tentatives pour la création d'infrastructures pour améliorer les temps de calcul, l'accès aux données, le partage et la collaboration de manière distribuée dans les grilles de calcul. Nous avons vu dans les chapitres précédents l'importance des technologies de grille de calcul et des services web dans le domaine du calcul scientifique. Nous avons vu aussi que différents travaux se penchent sur la question d'orchestration de *grid services* en utilisant principalement GT4 qui supporte les normes OGSA et WSRF comme environnement de grille et ActiveBPEL comme moteur d'orchestration.

Nous avons vu dans la section 2.7 différents points de vue de la notion de *Grid services*. Dans notre approche, un ***grid service*** est un service web qui permet d'exécuter un service offert par la grille (soumission de job, récupération du statut du job, récupération des résultats, etc.) ou une partie d'une application scientifique qui peut être réutilisable.

Dans ce chapitre nous exposons notre modèle de gridification qui consiste en l'intégration des deux environnements de grille (glite) et de services web (axis2 et orchestra) pour automatiser l'implémentation d'applications scientifiques en utilisant les *grid services*.

4.2. Approche proposée

4.2.1. Vue d'ensemble du système proposé

Notre solution se base sur les différents travaux qu'on a vu dans notre étude, entre autre **[Kll 06]**, **[Ija 08]**, **[Oez 07]**, **[Aba 10]**, …, aussi sur les caractéristiques du langage BPEL qui lui ont permis d'émerger non seulement dans le domaine de l'entreprise comme l'indique son nom (Business Process : CRM, …) mais également dans les domaines scientifiques (Workflow scientifique). Pour atteindre nos objectifs d'intégration des technologies grilles et services web, nous avons utilisé l'***encapsulation***. Dans notre contexte, encapsuler les applications et les commandes de la grille, signifie les exposer à travers une interface standard (services web) et c'est ce qu'on a appelé "*Grid services*". Le but étant de :

- Faciliter l'accès aux opérations offertes par la grille ;
- Permettre un faible couplage ;
- Orchestrer les services obtenus pour constituer des workflows scientifiques plus complexes ;
- Réutilisation de morceaux d'applications dans d'autres applications ;

- Mise à jour flexible des applications ;
- Etc.

La figure 4.1 représente une vue d'ensemble du système à mettre en œuvre.

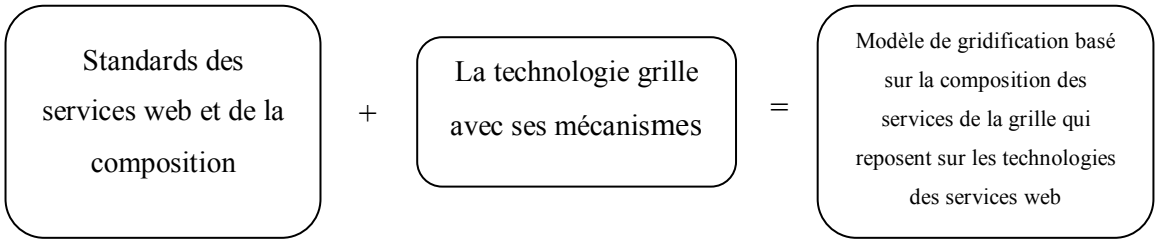

Figure 4. 1: Fondement de l'approche

4.2.2. Architecture du système proposé

L'architecture de notre approche est représentée sous la forme de quatre couches, comme indiqué dans la figure 4.2 :

- ✓ **Couche utilisateur** : C'est le portail par lequel l'utilisateur (le scientifique) formule sa requête après son authentification ;
- ✓ **Couche workflow :** C'est sur cette couche que les workflows scientifiques seront déployés et exécutés ;
- ✓ **Couche grid services :** C'est sur cette couche que les *grid services* seront déployés et exécutés. C'est à ce niveau que l'interaction avec la grille est effectuée ;
- ✓ **Couche gLite :** C'est sur cette couche que l'exécution des applications scientifiques se concrétise.

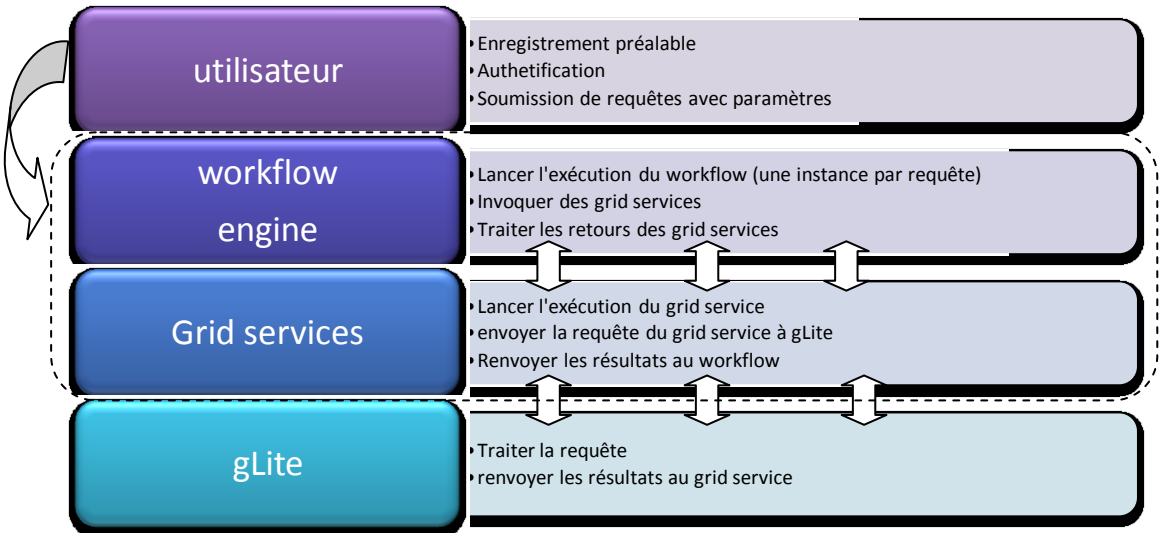

Figure 4. 2: Architecture en couche de l'approche proposée

Dans les sections suivantes nous verrons le fonctionnement interne de chaque couche en mettant en évidence l'interaction entre ces différentes couches.

4.2.2.1. La couche utilisateur

Cette couche est une interface à la grille. C'est un ensemble de Servlets qui permettent l'authentification, l'accès à l'espace de l'utilisateur, la création de fichier jdl (section 4.3.3), la modélisation et le déploiement du workflow, … La couche utilisateur représente l'utilisation graphique des *grid services*.

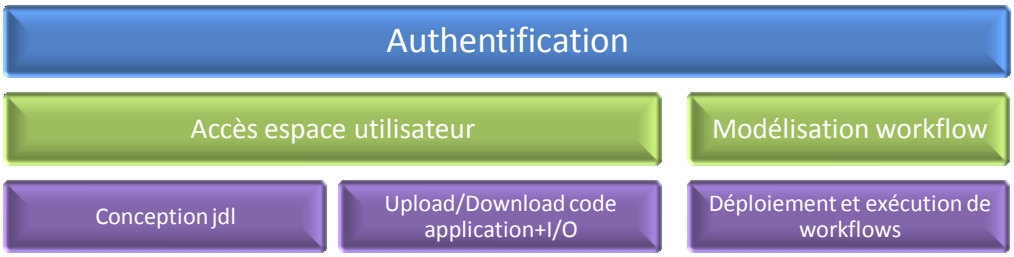

Figure 4. 3 : Composants de base du portail utilisateur

Selon les besoins de l'utilisateur, cette couche peut communiquer avec la couche workflow en envoyant des requêtes aux processus déployés ou avec la couche *grid services* en invoquant, par exemple, le *grid service* de suivi de l'exécution ou de récupération des résultats sans passer par la couche workflow. Les deux communications se font avec échange de messages SOAP en utilisant les interfaces WSDL du workflow ou des *grid services*. La communication avec la couche gLite se fait par l'intermédiaire des deux autres couches.

4.2.2.2. La couche workflow

Cette couche implémente le moteur d'orchestration. Ce dernier comporte principalement un conteneur de workflow et un conteneur des *grid services* à invoquer. Le moteur utilisé est "orchestra" (section 4.7.2.1.2). Il est intégré comme un composant de l'UI (section 1.11).

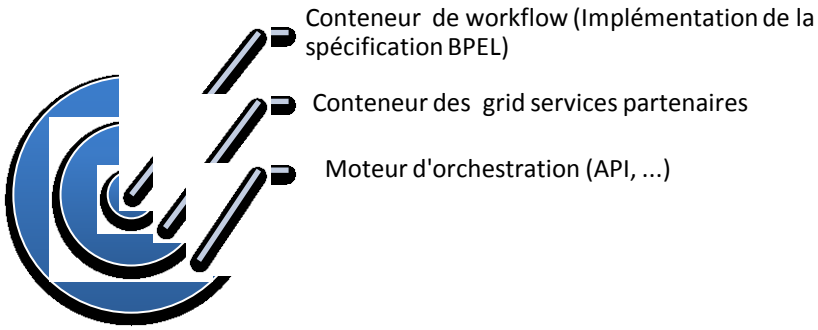

Figure 4. 4 : Représentation architecturale du moteur d'orchestration

Cette couche communique avec la couche *grid services* via les balises du langage d'orchestration.

4.2.2.3. La couche grid Services

C'est la couche qui permet la communication avec la grille en implémentant les mécanismes d'interaction. C'est ce composant qui jouera le rôle de l'utilisateur de la grille. La figure 4.5 montre les différents composants d'un *grid service*. Ce dernier est représenté par une interface et un noyau.

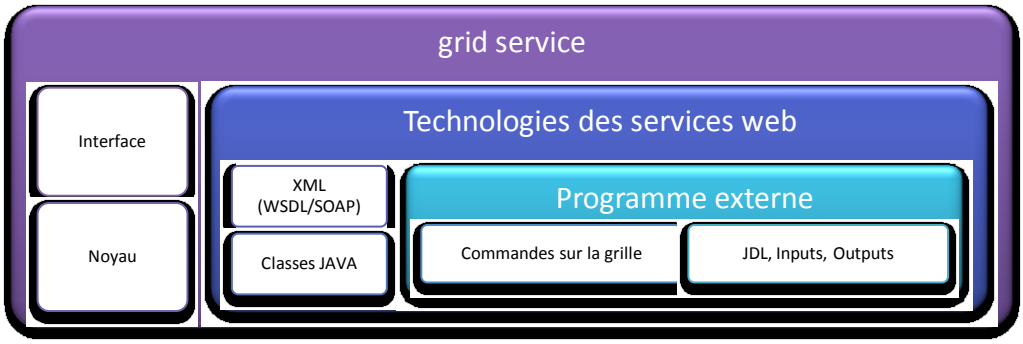

Figure 4. 5: Architecture interne d'un *grid service*

4.2.2.3.1. L'interface

L'interface est l'ensemble des technologies des services web qui permettent d'exposer le *grid service* lui-même comme un service web. Ce dernier est implémenté en utilisant la technologie java qui permet l'exécution de programmes externes (script batch, C, C++, Fortran, …).

4.2.2.3.2. Le noyau

Le noyau du *grid service* est constitué de programmes externes à exécuter par les classes java. Ces programmes exécutent l'une des commandes de la grille (section 4.6). Pour effectuer cette opération un programme a besoin de connaitre l'utilisateur, le mot de passe du proxy, le nom du fichier JDL, le chemin vers le jdl, les inputs et l'exécutable de l'application scientifique.

4.2.2.4. La couche gLite

Elle représente la couche OS (noyau) de l'architecture. C'est dans cette couche que les applications scientifiques sont exécutées, que les données et les ressources sont gérées, … Vue l'importance de cette couche dans la mise en place de notre architecture, nous détaillerons ses composants, ses mécanismes et l'interaction entre eux avant d'expliquer comment on a défini la communication entre cette couche et la couche *grid services*. **[Ela 06] [Gli 09] [Glt 10]**

4.3. Fonctionnement interne de gLite

Les services offerts dans gLite peuvent être regroupés en cinq ensembles de services de haut niveau : L'accès à la grille, la sécurité, le système d'information et de monitoring, le système de gestion de la charge de travail et la Gestion des données.

La figure4.6 présente ces différents services. Dans les documents représentants le middleware, ces services sont nommés *grid services* (les services de la grille)

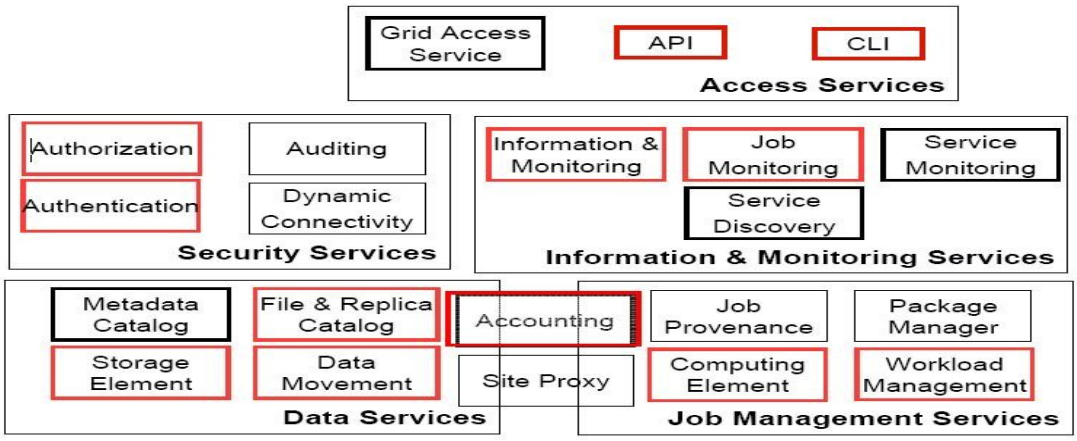

Figure 4. 6: Les services du middleware gLite

4.3.1. Service d'information (IS[48])

Ce service fournit des informations sur l'état des ressources, ce qui est crucial pour les opérations sur la grille. C'est aussi par ce service que les ressources sont découvertes et monitorées. Il existe deux mécanismes de service d'information dans gLite, notamment R-GMA[49] pour l'accounting, le monitoring et la publication d'informations au niveau utilisateur, et le MDS[50], qui est utilisé pour découvrir des ressources et de leur état de publication. Les fonctions du service d'information dans gLite sont présentées dans la figure 4.7.

Figure 4. 7: Fonctions du service d'information dans gLite.

[48] IS : Information Service
[49] RGMA : Relational Grid Monitoring Architecture (Architecture relationnelle de monitoring de grille)
[50] MDS : Globus Monitoring and Discovery Service (Service de découverte et de monitoring de globus)

4.3.1.1. R-GMA

Basé sur le GMA[51], le R-GMA est une base de données relationnelle qui facilite la recherche des informations de l'utilisateur. C'est une structure qui comprend trois composantes fondamentales, à savoir le registre, les producteurs et les consommateurs. Les producteurs publient les ressources qu'ils ont à offrir et les consommateurs recueillent les informations des ressources dont ils ont besoin à partir des tables dans la base. Dans le modèle R-GMA, la base est un monde virtuel dans le sens où il utilise un registre qui sert de médiateur entre les producteurs et les consommateurs plutôt que d'avoir une base de données réelle.

Le modèle R-GMA utilise la technologie des servlets pour l'interface de communication entre les consommateurs, les producteurs et le registre dans gLite. Par l'établissement d'un protocole de communication, ces différents acteurs sont capables de transmettre l'information de manière efficace entre les uns et les autres. L'utilisation de cette technologie permet également une adaptation facile et rapide aux autres systèmes de Services web. La figure 4.8 donne une représentation de base du modèle R-GMA entre les producteurs, les consommateurs et le registre.

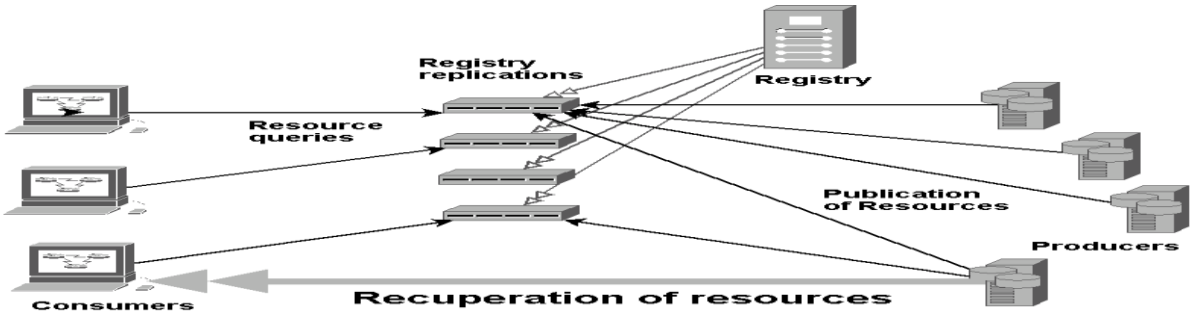

Figure 4. 8: La réplication des tables du Registre pour l'accès aux ressources dans de multiples producteurs.

4.3.1.2. MDS

4.3.1.2.1. L'Architecture MDS

Le MDS possède essentiellement une arborescence où l'information sur les ressources est recueillie à chaque niveau de l'arbre comme illustré dans la figure 4.9.

[51] GMA : Grid Monitoring Architecture (Architecture de monitoring de grille)

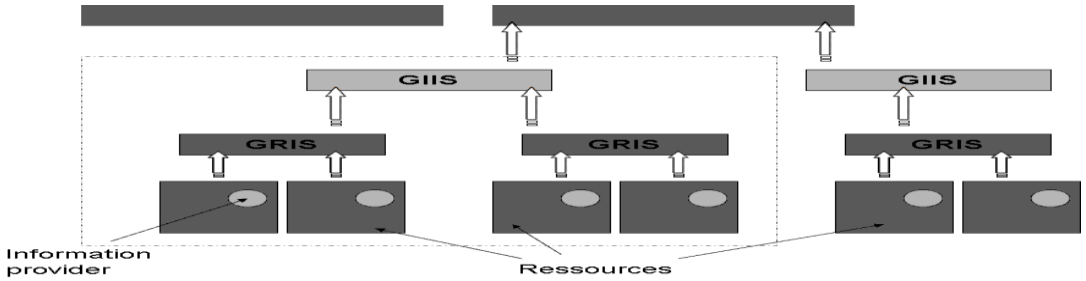

Figure 4. 9: Structure arborescente hiérarchique du système de monitoring et de la découverte.

4.3.1.2.2. Le serveur d'information de ressource de la grille (GRIS[52])

Nous pouvons voir dans la figure 4.9 que l'information sur les ressources individuelles au niveau fondamental est générée par le fournisseur d'information que l'on retrouve sur la plupart des CEs et SEs. Ces informations sont ensuite recueillies et diffusées par le GRIS, qui est essentiellement un serveur LDAP[53]. Le GRIS sur chaque CE ou SE gère toutes les informations pertinentes à un élément de ressource en particulier. Cette information est ensuite transmise au niveau suivant, qui est le serveur GISS.

4.2.1.2.3. Le Serveur d'information d'index de site de gille (GRIIS[54] ou GRISS)

Le GRISS est également un serveur LDAP responsable de la publication de toutes les informations des ressources sur le même site. Ces informations sont obtenues en rassemblant des informations de chaque GRIS sur le site.

4.2.1.2.4. Le Serveur d'index de base de données d'information de Berkeley (BDII[55])

Le BDII est le serveur LDAP au plus haut niveau de la structure MDS. Chaque serveur BDII est configuré pour monitorer un certain nombre de GRIIS où l'information des ressources sur un site particulier est transmise pour interrogation dans la grille. Comme les BDIIs sont au plus haut niveau dans le MDS, un BDII contient toutes les informations sur les ressources disponibles sur les sites qu'il monitore. Enfin, au plus haut niveau de tous les BDIIs dans une VO, il ya un BDII globale qui permet de relier les différents BDIIs dans la même VO.

Ceci est un serveur où toutes les informations utiles concernant la VO sont stockées. L'IS est un élément crucial dans la structure de gLite car il joue un rôle important dans le

[52] GRIS : Grid Resource Information Server
[53] LDAP : Lightweight Directory Access Protocol
[54] Serveur d'information d'index de site de gille est la traduction libre de GRISS ou GRIIS : Site Grid Index Information Server
[55] Serveur d'index de base de données d'information berkeley est la traduction libre de BDII : Berkeley Database Information Index server

monitoring et la découverte de ressources essentielles pour le fonctionnement normal de gLite.

4.3.2. Système de gestion de la charge de travail (WMS[56])

Le WMS est le principal composant de la structure gLite. Il gère la soumission et l'annulation des jobs depuis l'UI via le service réseau. C'est en général un service de médiation qui aide à planifier des tâches dans gLite de la manière la plus efficace, en sollicitant des informations importantes sur l'état des ressources disponibles à tout moment.

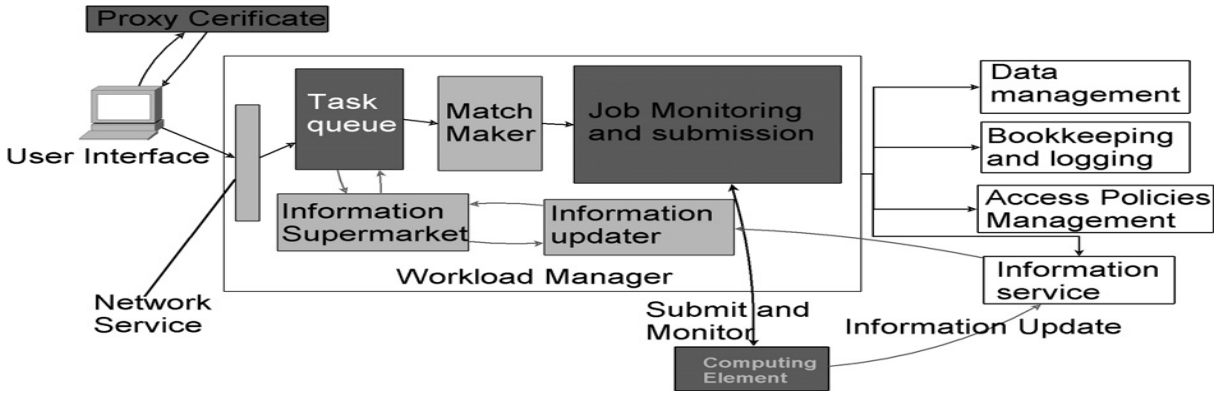

Figure 4. 10: Structure interne du WMS.

4.3.2.1. Gestionnaire de la charge de travail (WM[57])

Le WM est l'élément principal dans le WMS. Il comprend la file d'attente des taches *(task queue)*, l'agent de liaison *(matchmaker)*, le supermarché de l'information *(information supermarket)*, la mise à jour de d'informations *(information updater)* et le gestionnaire de jobs *(job handler)*. Son rôle principal est la prise en charge des requêtes de job au niveau utilisateur en procédant à l'allocation et le paquetage optimale de job avant la soumission aux éléments de ressource. Les requêtes de jobs sont écrites principalement dans un langage de description de job (JDL). Ce processus d'allocation et de paquetage est réalisé avec la contribution d'autres services de la grille, notamment, la gestion des données *(data management)*, les services de bookkeeping et d'information *(bookkeeping* and *information services)*. C'est par le WM que le statut des jobs soumis est récupéré de l'IS et ensuite relayé à l'UI.

4.3.2.2. La fille d'attente des taches

C'est le composant qui traite la manipulation d'une série de requêtes de jobs depuis l'UI via le service réseau. Le service réseau agit comme un portail pour accepter toutes les requêtes externes au

[56] WMS : Workload Management System
[57] WM : Workload Manager

WM. La fille d'attente effectue également l'authentification des utilisateurs pour l'utilisation des ressources de la grille via des certificats numériques du proxy. Une fois la requête du job (la soumission ou l'annulation du job) soumise au service réseau, la requête est interprétée dans le JDL avant de la passer à la fille d'attente. Une fois que cette dernière atteint la fille d'attente, elle est mise en file dans la liste des requêtes des jobs pour une éventuelle exécution par le *matchmaker*. Le passage des requêtes des jobs au *matchmaker* est un processus à double sens. Si aucune des ressources appropriées n'est disponible pour l'exécution, la requête est retournée à la fille d'attente pour une éventuelle expédition vers le *matchmaker* plus tard.

4.3.2.3. Le matchmaker

Le *matchmaker* est le composant dans le WM où les requêtes sont traitées pour correspondre à la ressource disponible la plus appropriée. Le *matchmaking* se fait en deux parties, à savoir le **traitement de la requête**, et la **sollicitation de renseignements** sur les ressources. D'une part, le traitement des requêtes est basé sur plusieurs critères, qui comprennent les politiques d'accès, les préférences professionnelles de l'utilisateur et la nature du job demandé. D'autre part, la sollicitation de renseignements sur les ressources est réalisée en interrogeant le **supermarché de l'information**, qui contient une mise à jour des informations concernant les ressources de la grille. Lorsque les deux parties du matchmaker sont prêtes, le processus de *matchmaking* peut être réalisé afin d'optimiser l'ordonnancement des requêtes à des ressources disponibles de la grille. La requête traitée passe à l'étape suivante pour la soumission des jobs aux éléments de ressource.

4.3.2.4. Le supermarché de l'information

Le supermarché de l'information est un cache où l'information globale des ressources relatives à la grille est stockée pour l'interrogation par le *matchmaker*. Les informations enregistrées dans le supermarché de l'information sont constamment mises à jour via **le responsable de la mise à jour de l'information**.

4.3.2.5. Le responsable de la mise à jour de l'information

Le responsable de la mise à jour de l'information est un composant où la mise à jour des informations sur les ressources dans le supermarché de l'information est faite. Cette mise à jour est effectuée par la lecture des informations du serveur BDII de haut niveau de la VO.

4.3.2.6. Le gestionnaire de Job

Lorsque le processus de *matchmaking* est fait, la requête du job est soumise aux éléments de ressources pour le traitement par le gestionnaire de job. Ce dernier se compose de quatre éléments de base qui sont responsables: du paquetage des jobs, le la soumission des jobs, de l'annulation et du monitoring des jobs. Le processus de paquetage de jobs est fait par l'adaptateur de jobs (*job adapter*) et le contrôleur de jobs (*job controller*). Dans l'adaptateur de jobs, le JDL est modifié pour une éventuelle soumission par le contrôleur de jobs. Le script *job wrapper* est également écrit dans l'adaptateur de jobs pour créer un environnement d'exécution approprié pour le CE. Une fois le JDL exécuté par l'adaptateur de jobs, il sera passé au contrôleur de jobs où le job sera soumis par CondorC[58]. Les jobs soumis sont ensuite suivis activement par le moniteur de log *(log monitor)* pour les actions appropriées pour répondre aux événements liés aux états des jobs.

4.3.3. Le langage de description de job (JDL)

Le JDL est un langage de haut niveau utilisé pour paramétrer un job avec des paramètres spécifiques à l'utilisateur. Le JDL est utilisé par le WMS pour comprendre la spécification du job afin de trouver le meilleur *matching* des ressources de la grille. Certains des principes de base du JDL sont exposés dans cette section.

4.3.3.1. Principes

Le fichier JDL est une description du job qui est essentiellement composé de lignes ayant un format semblable à : **Attribut = Expression**

Les expressions peuvent être de longueur variable, mais elles doivent se terminer par un point-virgule (**;**). Les commentaires peuvent être précédés par le caractère / / ou \ #. Pour des commentaires de plusieurs lignes, / * et * / peuvent être utilisés.

Dans les fichiers JDL, il existe des attributs obligatoires et non obligatoires, et il existe ceux qui sont propres à un type spécifique du JDL. Le tableau 3 présente quelques attributs.

Attribut	**Obligatoire ?**	**Signification**	**Exemple**
JobType	Oui (normal, parametric)	Spécifier le type du job (Normal, Parametric)	JobType = "Normal ";
Type	Oui (job, dag, collection)		

[58] CondorC : est le composant principal responsable d'effectuer les soumissions et les annulations des jobs atuels.

Executable	Oui	Spécifier l'exécutable	Executable = "test.sh";
Parameters	Oui pour le job paramétrique	Définir le nombre maximal de l'attribut \ **_PARAM** \ _	Parameters = (N) ;
ParameterStart	Oui pour le job paramétrique	Définir le nombre initial pour \ **_PARAM** \ _	ParameterStart = 1;
ParameterStep	Oui pour le job paramétrique	Définir l'incrément de _**PARAM** \ _.	ParameterStep = 1;
Arguments	Non	Les arguments pour l'exécutable	Arguments = "hello 10";
StdOutput	Oui	Spécifier les outputs standards	StdOutput = "std.out";
StdError	Oui	Spécifier les erreurs standards	StdError = "std.err"; Peut être le même que StdOutput
StdInput	Non	Spécifier les inputs standards	StdInput = "std.in";
InputSandbox	Non	Transférer les fichiers input depuis l'UI au WNs	InputSandbox = {"test.sh","std.in"};
OutputSandbox	Non	Transférer les fichiers output depuis les WNs jusqu 'à l'UI	OutputSandbox = {"std.out","std.err"};
Environment	Non	Elargir l'environnement	Environment = {"CMS_PATH=$HOME/cms","CMS_DB=$CMS_PATH/cmdb"};
Requirements	Non	Imposer des contraintes sur le CE	Requirements = other.GlueCEInfoLRMSType == "PBS";
Rank	Non	Appliquer un poids pour sélectionner le CE	Rank = other.GlueCEStateFreeCPUs;
RetryCount/ ShallowRetryCount	Non	L'auto-soumission du job jusqu'à ce qui soit soumis. Ils prennent par défauts les valeurs définies par **MaxRetryCount** et **MaxShallowRetryCount** dans le WMS	RetryCount = 0; ShallowRetryCount = 3;
DataAccessProtocol	Non	Définir le protocole utilisé pour le transfert des fichiers et la récupération des sorties des jobs	DataAccessProtocol = {" gsiftp ", "https "};
PerusalFileEnable	Non	Activer la lecture du job	PerusalFileEnable = true;
PerusalTimeInterval	Non	Spécifiez la fréquence en secondes de la copie des fichiers spécifiés sur la machine WMS,	PerusalTimeInterval = 30;

Tableau 3: Attributs d'un fichier JDL

4.3.3.2. Job Simple

Un job simple est de type "Normal". Pour exécuter un programme sur la grille, on peut écrire un fichier JDL de la manière suivante.

```
[
Type="Job";
JobType = "Normal ";
RetryCount = 0;
ShallowRetryCount = 3;
Executable = "compress \_hgdemo .sh";
```

```
Arguments = "argA argB";
InputSandbox = {"file :/// home/ compress_hgdemo.sh "};
StdOutput = "std .out ";
StdError = "std .err ";
OutputSandbox = {"std.out ", "std.err "," fileoutput "};
DataAccessProtocol = {" gsiftp ", "https "};
]
```

4.3.3.3. Job Paramétrique

Un job paramétrique est un job où un ou plusieurs de ses attributs sont paramétrés. Le monitoring et la gestion du job se fait toujours par un JobID unique, comme si le job était unique. L'attribut **Parameter** peut être soit un nombre ou une liste d'éléments (typiquement des chaînes, mais pas entourés de guillemets doubles). L'attribut InputSandbox (si présent) doit être cohérent avec les paramètres.

```
[
JobType = "Parametric";
Parameters = (N); (ou bien: Parameters = {EARTH,MOON,MARS}; sans ParameterStart et ParameterStep )
ParameterStart = 1;
ParameterStep = 1;
RetryCount = 0;
ShallowRetryCount = 3;
Executable = "compress \_hgdemo .sh";
Arguments = " argA_PARAM_ argB_PARAM_ ";
InputSandbox = {" file :/// home/compress_hgdemo.sh"};
StdOutput = "std .out ";
StdError = "std.err ";
OutputSandbox = {"std.out ", "std.err "," fileoutput "};
DataAccessProtocol = {" gsiftp ", "https "};
]
```

Les paramètres **ParameterStar**t et **ParameterStep** sont mis dans le JDL que si **Parameters** est un nombre. \ **_PARAM** \ _ sera une série de nombres ou de chaine de caractère selon le type de **Parameters**. Ceci est important car gLite utilise cette série comme paramètres aux arguments et exécutes les jobs.

4.3.3.4. Job DAG

Un job DAG est un ensemble de jobs où les inputs, les outputs ou l'exécution d'un ou de plusieurs jobs **dépondent** d'autres jobs. Les dépendances sont représentées par des graphes orientés acycliques (Directed Acyclic Graphs : DAG), où les nœuds sont des jobs, et les arcs identifient les dépendances. L'attribut "nodes" est le noyau du job DAG.

```
[
 Type = "dag";
 max_nodes_running = 4;
 nodes = [
  nodeA = [
```

```
    file ="nodes/nodeA.jdl" ;
  ];
  nodeB = [
        file ="nodes/nodeB.jdl" ;
  ];
  nodeC = [
        file ="nodes/nodeC.jdl" ;
  ];
  nodeD = [
        file ="nodes/nodeD.jdl";
  ];
    dependencies = {
        {nodeA, nodeB},
        {nodeA, nodeC},
        { {nodeB,nodeC}, nodeD }
  }
 ];
]
```

4.3.3.5. Job Collection

Un job collection est un ensemble de jobs **indépendants** que l'utilisateur veut soumettre et monitorer comme une unique requête. Les jobs d'une collection sont soumis comme des nœuds DAG **sans dépendances**. Tous les nœuds partagent le même **InputSandbox**. Le JDL est une liste de classes, qui décrit les sous-jobs.

```
[
  type = "collection";
  InputSandbox = {"date.sh"};
  RetryCount = 3;
  nodes = {
    [
      file ="jobs/job1.jdl" ;
    ],
    [
      [
  Executable = "/bin/sh";
  Arguments = "date.sh";
  StdOutput = "date.out";
  StdError =  "date.err";
  OutputSandbox ={"date.out", "date.err"};
      ]
    ],
    [
      file ="jobs/job3.jdl" ;
    ]
  };
]
```

4.3.3.6. Job MPI

L'exécution des jobs parallèles est un point essentiel pour l'informatique et les applications modernes. La librairie la plus utilisée pour supporter les jobs parallèles est la MPI[59]. A ce jour, les jobs parallèles peuvent êtres exécutés que seulement sur un même CE. Cependant, plusieurs projets sont impliqués dans les études concernant la possibilité d'exécuter des jobs parallèles sur des WNs appartenant à des CE différents.

```
[
 Type = "Job";
#Obligatoire
 JobType = "MPICH";
# Le nombre de CPU qui seront utilisés
 NodeNumber = 2;
#L'exécutable doit avoir été compilé avec les avec les librairies mpicc
 Executable = "cpi";
 StdOutput = "cpi.out";
 StdError = "cpi.err";
 InputSandbox = {"cpi"};
 OutputSandbox = {"cpi.err","cpi.out"};
 RetryCount = 3;
]
```

4.3.4. L'élément de calcul (CE[60]):

Le CE peut être vu comme un cluster de ressources de la grille qui sert à exécuter les jobs. Chaque CE est composé de trois éléments de base, notamment le portail de la grille [61] (GG), le système de gestion des ressources locales[62] (LRMS) et les nœuds travailleurs[63] (WN). La figure 4.11 représente la structure d'un CE.

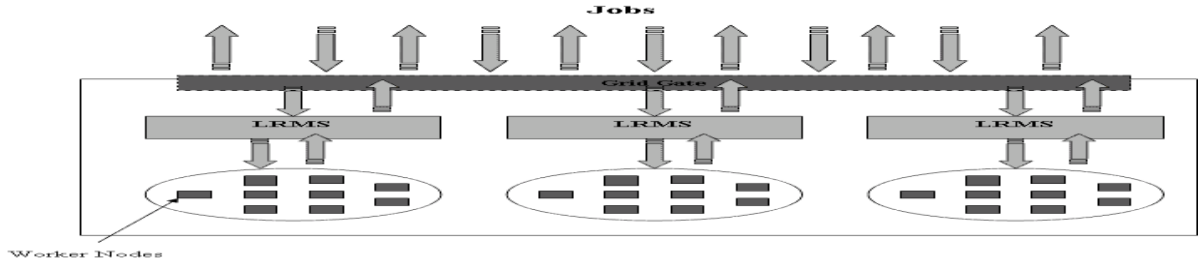

Figure 4. 11: Structure d'un CE

[59] MPI : Message Passing Interface
[60] CE : Computing Element
[61] Traduction libre de GG : Grid Gate
[62] Traduction libre de LRMS : local resource management system
[63] Traduction libre de WN : Worker Nodes

4.3.4.1. Le portail de la grille

La GG est l'interface entre le cluster et la grille. Elle réglemente l'entrée et la sortie du job dans le CE. La GG est chargée d'accepter les jobs et de les envoyer pour exécution sur les WNs via le LRMS.

4.3.4.2. L'allocation des ressources

C'est le R-GMA communément nommé LRMS, ce service gère l'allocation des ressources locales dans le cluster.

4.3.4.3. Les nœuds travailleurs

Les WNs sont généralement les processeurs où les jobs sont effectivement exécutés.

4.3.5. La gestion des données (DM[64])

La gestion des données dans gLite peut être regroupée en trois catégories: les éléments de stockage (SE[65]), les catalogues de fichiers locaux (FCL[66]) et le service de transfert de fichiers (FTS[67]). Ces trois composants sont essentiels pour accéder aux données dans gLite.

4.3.5.1. L'élément de stockage

Les SEs sont des composants dans gLite qui stockent des données pour une récupération éventuelle par l'utilisateur ou par d'autres applications. Dans un environnement de grille, les fichiers sont répliqués et stockés dans des endroits différents pour offrir de multiples possibilités d'accès rapide. Ainsi, afin d'assurer la cohérence et donc l'unicité des données, les fichiers sont limités aux propriétés de lecture seule. Dans cet environnement de type une écriture et plusieurs lectures (*write-once-read-many*), les fichiers sont écrits une fois lors de la création et ne peuvent être modifiés ultérieurement.

Pour définir un SE, nous devons connaître le gestionnaire des ressources de stockage (SRM[68]), les types de ressource de stockage (SRT[69]) et le protocole de transfert (TP[70]) utilisé par le SRM.

[64] DM : Data Management
[65] SE : Storage Elements
[66] LFC : Local File Catalog
[67] FTS : File Transfer Service
[68] SRM : Storage Resource Manager
[69] SRT : Storage Resource Types
[70] TP : Transfer Protocol

4.3.5.1.1. Le gestionnaire de ressource de stockage

Le SRM est une interface au middleware qui rend les opérations de gestion de données standards entre les SEs de différent type de ressources, transparentes à l'utilisateur. Ces opérations de gestion de données comprennent les transferts de fichiers, la réservation d'espace, le re-nommage de fichiers et la création de répertoire.

4.3.5.1.2. Le types de ressources de stockage

Il existe un certain nombre de SRT disponibles, mais ils peuvent être décomposés en deux catégories principales, à savoir stockage basé sur le disque (disk-based) et stockage basé sur la bande (fille d'arrivée) (tape-based). Pour les SEs relativement petits, l'implémentation du stockage basé sur le disque est employé conjointement avec le gestionnaire de zone de disque (*disk pool manager*), qui est le gestionnaire des ressources de stockage correspondant au stockage sur le disque (disk-based). Pour les plus grands SEs, le système de stockage de masse (*mass storage system MSS*) est implémenté avec CASTOR[71] en tant que gestionnaire des ressources de stockage. Pour les hybrides entre *disc pool storage* et *MSS*, il y a le gestionnaire des ressources de stockage *DCache*.

4.3.5.1.3. Le protocole de transfert

Basé sur le type de ressource des SEs, différents protocoles sont utilisés pour le transfert des fichiers dans et depuis le SE. Pour le transfert de fichier avec les SEs, le protocole GSIFTP[72] est utilisé pour faciliter le transfert rapide et sécuriser des données. Pour l'accès à distance aux fichiers sur les SEs, le protocole RFIO[73] est utilisé. Ainsi, basé sur le SRT dans chaque SE, il y a un SRM correspondant qui prend en charge les protocoles pour les différentes actions aux SEs.

4.3.5.2. Le catalogue de fichier local

Avant que les opérations de transfert et d'accès ne soient exécutées sur des fichiers dans les SEs, ils doivent être localisés et identifiés. Cette identification de fichiers dans les SEs se fait à travers l'utilisation d'identifiants différents, à savoir l'identifiant unique de la grille (GUID[74]), le nom logique du fichier (LFN[75]), l'URL de stockage (SURL[76]) et l'URL du transport (TURL[77]).

[71] CASTOR : CERN Advanced STORage Manager
[72] GSIFTP : grid security infrastructure file transfer protocol
[73] RFIO : remote file input/output protocol
[74] GUID : Grid Unique Identifier
[75] LFN : logical file name

4.3.5.2.1. L'identifiant unique de la grille

Le GUID est un identificateur qui identifie de manière unique les fichiers dans la grille. Il est associé à un fichier quand il est créé la première fois en utilisant une combinaison d'adresse MAC et d'horodatage (timestamp).

4.3.5.2.2. Le nom logique du fichier

Bien que le GUID identifie de façon unique un fichier dans la grille, il n'est pas couramment utilisé pour le localiser. Au lieu de cela, le LFN est utilisé. Il s'agit d'un identifiant qui utilise des chaînes de caractères lisibles par l'homme, il intuitif comme identifiant. Cela facilite l'interaction entre l'utilisateur et la grille. Le LFN d'un fichier n'est pas unique et un fichier peut avoir plusieurs LFNs pour des références multiples.

4.3.5.2.3. l'URL de stockage

La SURL d'un fichier ou le nom de fichier physique (PFN[78]), contient l'adresse physique du fichier répliqué dans la grille. Elle fournit des informations sur le SE ou ses SRMs et le chemin nécessaire pour localiser les fichiers sur le SE.

4.3.5.2.4. l'URL de transport

Tandis que la SURL fournit les informations sur la localisation du fichier dans la grille, la TURL fournit les informations d'accès et de recherche dans le fichier. Au premier regard, il semble redondant que la TURL contienne des informations identiques à la SURL. Cependant, la TURL contient également le protocole d'accès et l'information sur les ports physiques qui permettent la récupération physique du fichier concerné. Cette information est générée par les SRMs et peut changer au fil du temps. Ainsi, plusieurs TURLs existent pour un fichier donné et le nombre de TURL correspond au nombre de protocoles supportés par le SRM. En outre, le SRM peut tenir plusieurs fichiers en double pour réduire le goulot d'étranglement (bottleneck) dans l'accès aux fichiers.

Tandis que les GUIDs et LFNs sont utilisés pour l'identification des fichiers, les SURL et TURL fournissent les informations nécessaires pour accéder et récupérer les fichiers concernés.

Enfin, les GUIDs, LFNs, SURL et TURL sont enregistrés dans le LFC, où des traces entre les identificateurs de fichier sont stockées. Cela fournit une source de référence pour

[76] SURL : Storage URL
[77] TURL : Transport URL
[78] PFN : physical file name

localiser et accéder à des fichiers dans la grille. L'analogie avec une LFC serait le carnet d'adresses qui contient les coordonnées d'une liste de personnes. Chaque personne a un nom et une adresse résidentielle où l'on peut la localiser physiquement. Dans cette analogie, le nom serait comparable aux GUIDs et LFNs tandis que l'adresse du domicile est comparable aux SURL et TURL. Lorsque la grille veut obtenir des informations sur un fichier particulier, elle interroge le LFC pour obtenir les informations d'accès et de récupération du fichier concerné.

La figure 4.12 représente les différents aspects d'identification des fichiers dans la grille.

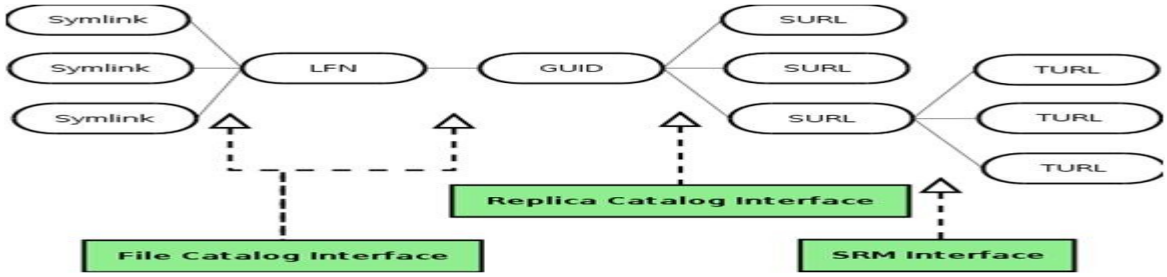

Figure 4. 12: Les accès aux fichiers dans la grille

4.3.5.3. Le service de transfert de fichier

Le FTS est un service de bas niveau qui permet le transfert de données entre les différents SEs. En général, le protocole de transfert utilisé entre SEs est le GSIFTP. En fait, le protocole nécessaire pour un transfert particulier, est établi sur la base des protocoles supportés par le SRM à la fois sur les SEs de source et de destination. Quelques-unes des terminologies utilisées dans le FTS sont introduites dans ce qui suit :

- Canal (**Channel**): il s'agit d'un chemin spécifique pour le transfert de fichiers.
- Fichier (**File**): c'est un ensemble de SURL de source et de destination.
- job de Transfert (**Transfer Job**): c'est un ensemble de fichiers avec des paramètres supplémentaires, notamment la cryptographie des données transférées.
- l'État du fichier (**File State**): c'est l'état d'un transfert de fichier individuel.
- l'État du Job (**Job State**): c'est l'état du job transféré basé sur les États des fichiers sous-jacent.

Une fois la soumission du job de transfert faite, il retourne un ID (Job ID), qui permet l'interrogation de l'état d'avancement ou l'annulation du Job. Le FTS supporte actuellement uniquement les paires SURL au lieu des paires GUID/LFN, qui sont plus intuitives.

4.4. Logging and BookKeeping

Le module LB permet le suivi des événements importants du job: le *matchmaking*, la soumission des jobs, l'annulation des jobs, etc. Cette information enregistrée est obtenue par

le WMS ou les CEs. Elle est ensuite transmise au composant LB le plus proche, nommé le *locallogger*, afin d'éviter les problèmes de réseau. Le *locallogger* a la responsabilité de transmettre cette information au composant LB principale nommé le serveur de Bookkeeping.

Dans le serveur de BookKeeping, l'information des événements liés au job (*job-event information*) est traitée et traduite en information du job de haut niveau (*higher-level job information*) (SUBMITTED, RUNNING, DONE, etc.). Ce serveur est affecté au job tout en long de sa durée de vie. La requête d'interrogation des événements du job peut être faite en utilisant l'UI.

Le module LB permet également de notifier les utilisateurs lorsque des modifications sont apportées notamment dans le job. Ceci est réalisé de manière systématique dans l'architecture de la grille où des URL Uniques pour identification (*unique URL job id*) sont donnés aux jobs. Cela coïncide avec l'URL du serveur de BookKeeping.

Le tracking (suivi) des événements du job est fait par un monitoring étroit du WMS où les événements du job sont transmis à partir des ressources de la grille.

4.5. Mécanisme de sécurité dans gLite

Dans cette section, nous discutons de l'utilisation et du rôle important des certificats qui servent de passeports dans la grille. Les *Certificats* sous l'infrastructure de sécurité de la grille (GSI[79]) sont des documents très importants qui garantissent l'authenticité et la sécurité des données transférées entre les composants de la grille. Elles permettent également l'inter-vérification de chaque composant individuel dans la grille. Dans le paragraphe suivant, nous discutons brièvement le rôle des proxys en matière de sécurité et de la délégation à distance des opérations de la grille.

Dans gLite, nous sommes souvent confrontés à la situation où un service distant agit au nom (pour le compte) d'un utilisateur, ce qui est appelée **délégation**. L'utilisateur doit autoriser le service distant avant que rien ne soit fait. Le processus d'autorisation est effectué en utilisant un certificat signé par la clé privée de l'utilisateur (certificat X.509 de l'utilisateur finale). Ce certificat de délégation est appelé «proxy». Avec le proxy, les jobs soumis par le service distant sont identifiés comme une délégation par l'utilisateur, et le proxy est interrogé avec le certificat utilisateur pour vérifier la délégation.

[79] GSI : Grid Security Infrastructure

Lorsque les utilisateurs ouvrent un compte, ils reçoivent un nom d'utilisateur et un mot de passe. En utilisant cette information, un utilisateur autorisé peut demander un certificat numérique X.509 d'une année qui est enregistré dans le répertoire « home » de l'utilisateur. Sur réception de ce certificat, les utilisateurs sont maintenant prêts à exécuter des jobs sur la grille en vertu de la GSI. Toutefois, avant de passer dans la grille, les utilisateurs doivent se connecter à cette grille. Cela se fait par la création de certificats de proxy temporaire à l'aide du certificat de l'utilisateur. Le certificat de proxy dure 12 heures. La durée de 12 heures est conçue pour réduire l'impact d'une éventuelle atteinte à la sécurité, par exemple, le vol d'un certificat de proxy.

Les ressources de gLite reçoivent également des certificats pour assurer leur authenticité et permettre leur identification. De cette manière, les composants de la grille peuvent être identifiés et authentifiés facilement.

L'identification et l'autorisation des utilisateurs par les ressources se fait par 2 mécanismes, notamment par le mécanisme *gridmap-file* et le mécanisme du service du membre de l'organisation virtuelle (VOMS[80]). Une fois l'accès autorisé aux ressources, l'utilisateur est en mesure d'effectuer des opérations de base sur la grille.

La figure 4.13 montre une vue d'ensemble des certificats dans le mécanisme de sécurité.

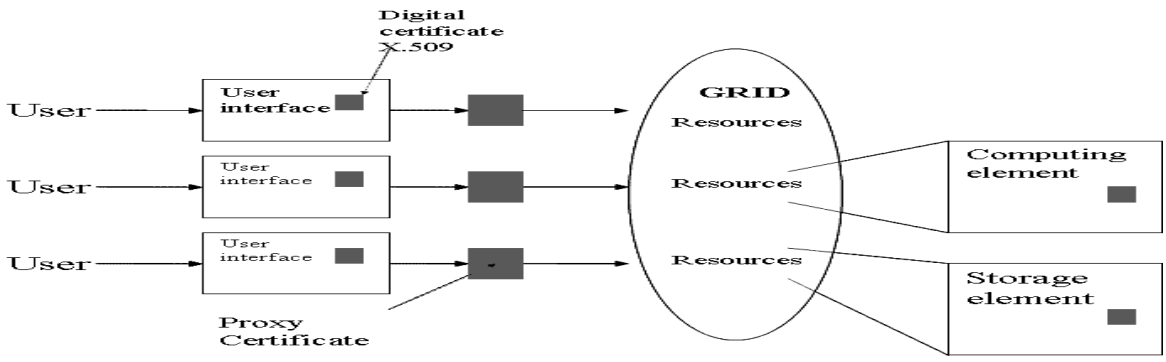

Figure 4. 13: Vue d'ensemble des certificats dans le mécanisme de sécurité dans gLite.

4.6. Utilisation de gLite

Dans cette section, nous donnons un aperçu des commandes fréquemment utilisées impliqués dans l'UI tout au long du cycle de vie d'un job dans gLite. Les principales phases pertinentes pour un utilisateur sont la soumission des jobs à gLite, la récupération des états du job et la collecte des sorties des jobs dans gLite.

[80] VOMS : Virtual Organization Membership Service

4.6.1. Initialisation

Avant la réalisation de toutes opérations en gLite, les utilisateurs doivent d'abord créer un proxy (similaire à un certificat) comme indiqué dans la section 4.5. Ce proxy a une durée variable (12h jusqu'à 1semaine) durant laquelle les utilisateurs peuvent effectuer diverses opérations pertinentes à gLite. Ce processus d'initialisation se fait par la commande **voms-proxy-init --voms <VO>** où **VO** est le nom de l'organisation virtuelle. Le mot de passe de la grille est exigé à l'utilisateur avant que le proxy ne soit attribué. La sortie est similaire à ce qui suit:

Enter GRID pass phrase:
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=DZ-eScience/CN=Medjek Faiza
Creating temporary proxy ..... Done
Contacting voms-02.pd.infn.it:15016 [/C=IT/O=INFN/OU=Host/L=Padova/CN=voms-02.pd.infn.it] "eumed" Done
Creating proxy ..... Done
Your proxy is valid until Sun May 1 02:40:23 2011

Cela fournit un niveau d'authentification des utilisateurs avant que les ressources de la grille ne leur soient rendues accessibles. Une fois que le proxy est attribué à utilisateur, ce dernier peut passer à la prochaine phase.

4.6.2. Soumission de job (Job Submission)

Cette phase est une partie très importante dans le cycle de vie du job dans gLite. C'est dans cette phase que les utilisateurs spécifient la nature du job (parallèle ou séquentiel) et d'autres paramètres en vue de le lancer. Ces spécifications sont étalonnées à l'aide du JDL (section 4.3.3).

Le WMS de gLite dispose de deux implémentations pour la gestion des jobs, notamment, le serveur réseau (NS[81]) et le proxy de gestion du job (WMProxy[82]). Les deux implémentations offrent des services similaires et les commandes en ligne sont aussi très similaires. Toutefois, WMProxy offre un bénéfice plus substantiel comme suit:

- Soumission d'ensembles de jobs
- Authentification plus rapide
- Matchmaking plus rapide
- Temps de réponse plus rapide pour les utilisateurs
- Débit supérieur du job

La soumission du job est faite en utilisant les commandes suivantes:

[81] NS : Network Server
[82] WMProxy : Work Management Proxy

Via WMproxy	**glite-wms-job-submit[-o joblist] –a jdlfile**
Via NS	glite-job-submit[-o joblist] –a jdlfile

Tableau 4: Commandes de Soumission de jobs en fonction des services

jdlfile est le fichier écrit avec JDL en précisant les paramètres et la configuration de ce job tandis que **joblist** est le fichier qui contient l'URL du job, équivalent à l'ID du job.

Si le job a été soumis avec succès, gLite retourne en sortie son ID, qui est l'URL par laquelle le statut du job peut être interrogé via l'UI. Un résultat typique obtenu après une soumission réussie est le suivant:

Connecting to the service https://infn-wms-01.ct.pi2s2.it:7443/glite_wms_wmproxy_server
================== glite-wms-job-submit Success ==================
The job has been successfully submitted to the WMProxy
Your job identifier is:
https://infn-lb-01.ct.pi2s2.it:9000/HU1eAJfY0pYkEsGIxVQyVQ
==

Si la commande **[-o joblist]** a été utilisé, l'identificateur du job (URL) sera stocké dans le fichier **joblist**. La liste des identificateurs du job dans le fichier **joblist** peut ensuite être récupérée dans les autres opérations du job en utilisant la commande **-i**.

4.6.3. L'état du job (Job Status)

Une fois le job soumis au WMS de gLite, l'état du job peut être obtenu à partir du journal de l'historique du job en interrogeant le LB. Le tableau 5 donne un certain nombre de commandes fréquemment utilisées pour la récupération de l'état du job.

Via WMproxy	**glite-wms-job-status ''jobURL''**
	glite-wms-job-status -i <filename>
Via NS	glite-job-status ''jobURL''
	glite-job-status -i <filename>

Tableau 5: Commandes de recherche de l'état de Job en fonction des services.

La commande **-i <nomfichier>** permet la récupération de l'état du job où les identifiants des jobs sont stockés dans le fichier avec le nom **<nomfichier>**.

Cela coïncide généralement avec le fichier utilisé avec la commande **-o** dans la soumission du job. Si l'état du job est récupéré correctement, le résultat est semblable au suivant:

glite-wms-job-status https://infn-lb-01.ct.pi2s2.it:9000/HU1eAJfY0pYkEsGIxVQyVQ
================== glite-wms-job-status Success ==================
BOOKKEEPING INFORMATION:
Status info for the Job: https://infn-lb-01.ct.pi2s2.it:9000/HU1eAJfY0pYkEsGIxVQyVQ
Current Status: Done (Success)
Logged Reason(s):
- job completed
- Job Terminated Successfully

Exit code: 0
Status Reason: Job Terminated Successfully
Destination: ce01.grid.arn.dz:8443/cream-pbs-eumed
Submitted: Sat Apr 30 15:00:48 2011 CET
==========

4.6.4. Annulation d'un job

Si les jobs soumis au WMS doivent être annulés, les commandes dans le tableau 6 sont utilisées.

Via WMproxy	**glite-wms-job-cancel <jobID>**
	glite-wms-job-cancel -i <filename>
Via NS	glite-job-cancel <jobID>
	glite-job-cancel -i <filename>

Tableau 6: Commandes d'annulation de job en fonction des services.

Semblablement à l'utilisation de l'état du job, la commande **-i** permet aux utilisateurs de choisir parmi une liste les jobs qu'ils veulent annuler. Si l'annulation de job est réussie, les utilisateurs obtiendront un résultat similaire comme suit:

Connecting to the service https://infn-wms-01.ct.pi2s2.it:7443/glite_wms_wmproxy_server
========== glite-wms-job-cancel Success ==========
The cancellation request has been successfully submitted for
the following job(s): -
https://infn-lb-01.ct.pi2s2.it:9000/HU1eAJfY0pYkEsGIxVQyVQ
==========

4.6.5. Collecte de résultats pour un job

Une fois l'état du job atteint le statut Done, les résultats du job peuvent être récupérés.

Via WMproxy	**glite-wms-job-output <jobID>**
	glite-wms-job-output -i <filename>
Via NS	glite-job-output <jobID>
	glite-job-output -i <filename>

Tableau 7: Commandes de récupération de job en fonction des services.

Similaire à ce qui précède, la commande **-i** permet aux utilisateurs de choisir parmi une liste des jobs parmi lesquels le résultat est récupéré. Si le résultat du job est récupéré avec succès, une sortie semblable à la suivante peut être obtenue:

Connecting to the service https://infn-wms-01.ct.pi2s2.it:7443/glite_wms_wmproxy_server
==========
JOB GET OUTPUT OUTCOME
Output sandbox files for the job:
https://infn-lb-01.ct.pi2s2.it:9000/HU1eAJfY0pYkEsGIxVQyVQ
have been successfully retrieved and stored in the directory:
/tmp/HU1eAJfY0pYkEsGIxVQyVQ
==========

Une fois le répertoire de sortie du job est connue, les résultats d'un job peuvent être consultées et les sorties peuvent être récupérées par l'utilisateur. Les répertoires de sortie du

job sont temporaires et sont effacées après un certain temps en fonction de l'administrateur du WMS.

La figure 4.14 présente un récapitulatif du cycle de vie d'un job, depuis sa soumission via l'UI jusqu'à la récupération des résultats, ainsi que les différents états qu'il prend pendant son parcours.

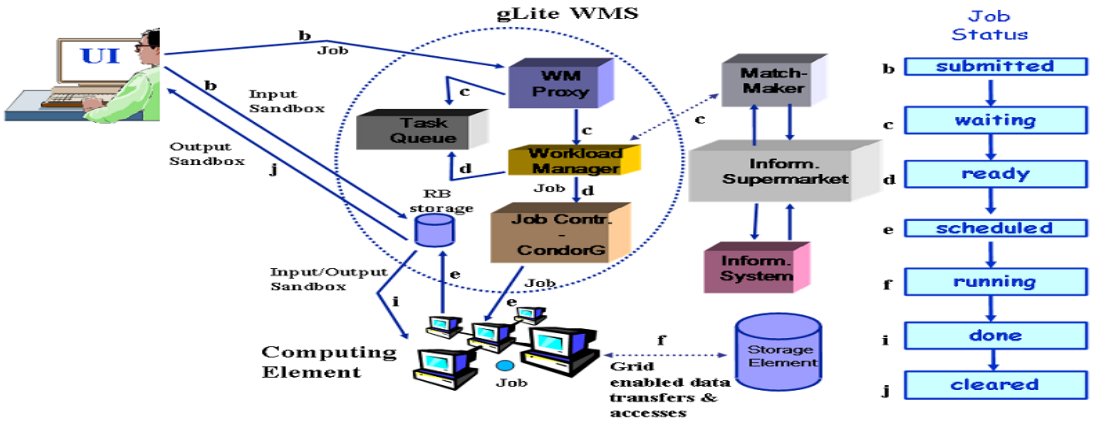

Figure 4. 14: Cycle de vie d'un job dans le middleware g-Lite

4.7. Modélisation

4.7.1. Modélisation de l'approche

4.7.1.1. Contraintes et solutions

Nous avons vu les architectures des différentes couches de l'approche proposée. Pour modéliser notre approche et mettre en place le mécanisme d'interaction entre les différentes couches, nous nous sommes basés sur notre étude du chapitre 1(section 1.11) et du chapitre 4 (section 4.3, 4.5 et 4.6) qui dit que:

a. L'utilisateur doit disposer d'un certificat en passant par les autorités de certificats (section 1.11, section 4.5) ;
b. Il doit s'enregistrer dans l'UI via l'administrateur de la grille avec dépôt de certificat (section 1.11, section 4.5) ;
c. Il doit ensuite se connecter à la grille via l'UI (exemple : avec ssh) (section section 1.11, 4.5);
d. Il doit aussi être authentifié dans la grille pour qu'il puisse solliciter ses différents composants (Avec son certificat de proxy) (section 4.5).

Pour pouvoir réaliser notre idée et répondre aux contraintes citées ci-dessus, nous avons choisis de :

1. Créer un groupe d'utilisateurs dans le système qu'on a nommé « GridServiceGroup », dans lequel on ajoute les utilisateurs qui veulent accéder aux *grid services*. Ce groupe jouera le rôle d'un utilisateur virtuel qui est reconnu et certifié dans le système ;
2. Installer et configurer le conteneur de services web, Axis, dans l'UI. L'installation sur le UI est du au fait que l'utilisateur doit être connecté à la grille via l'UI ;
3. Mettre le conteneur dans le groupe « GridServiceGroup », ce qui fait que les utilisateurs qui sont dans le groupe peuvent utiliser les *grid services* déployés (qui à leur tour appartiennent à ce groupe) en communiquant entre autres leur identifiant et mot de passe.

La figure 4.15 présente l'architecture globale du système pour exécuter une requête du client.

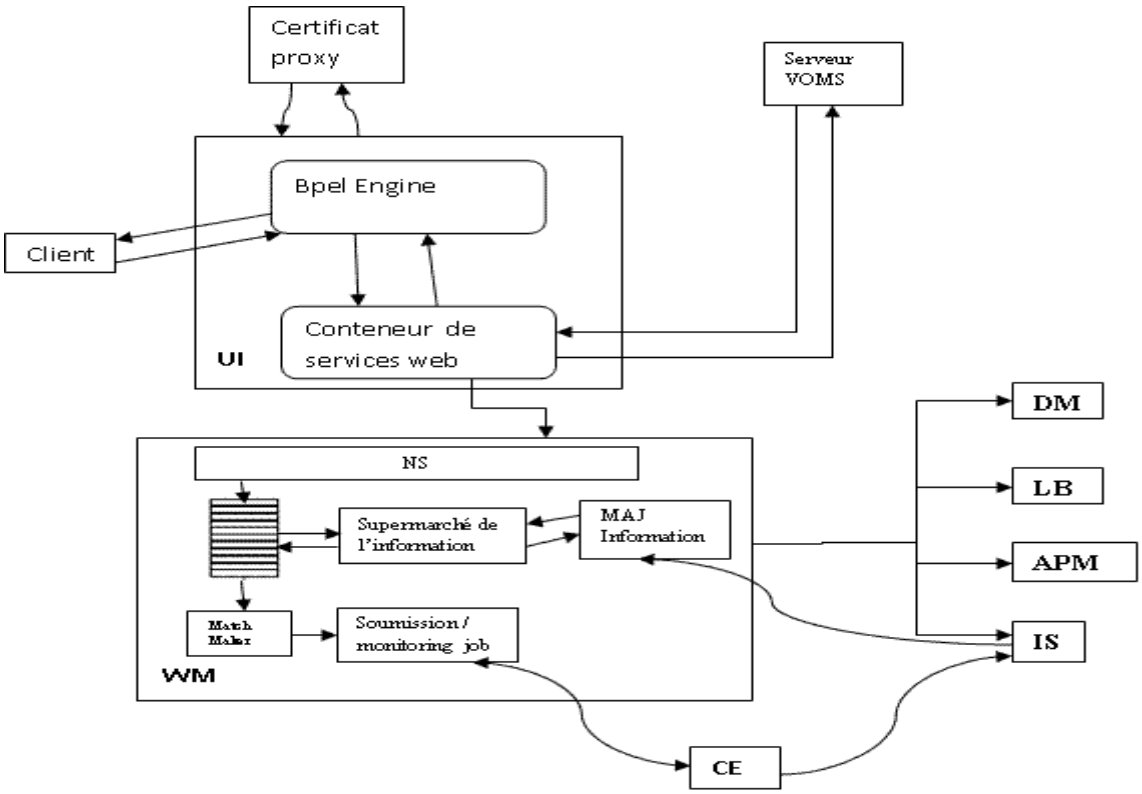

Figure 4. 15: Architecture Globale

4.7.1.2. Mécanisme de communication dans le grid service

Comme nous l'avions vu dans la section (4.2.2.3), la couche *grid services* se compose de l'interface et du noyau. Nous allons présenter dans la figure 4.16, le mécanisme de communication entre ces composants.

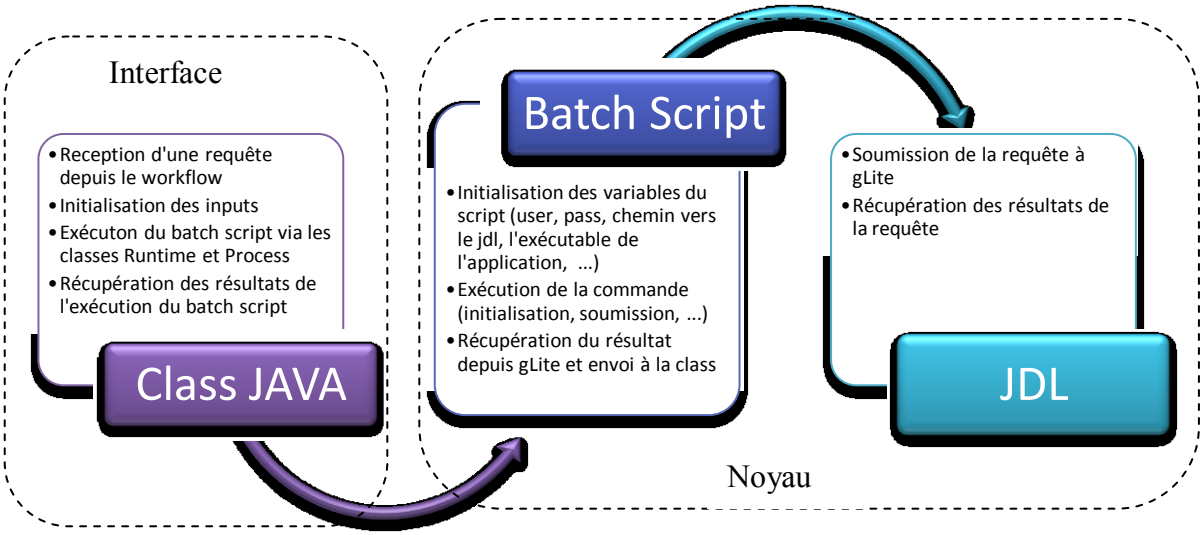

Figure 4. 16: Mécanisme de communication des composants du *grid service*

4.7.2. Implémentation du modèle proposé

Pour pouvoir mettre en place l'architecture proposée, nous avons intégré les environnements adéquats puis nous avons modélisé les *grid services*.

4.7.2.1. Intégration des environnements

4.7.2.1.1. Le middleware g-Lite

Comme nous l'avons vu, gLite est le middleware de la grille nationale DZ-Grid, c'est un middleware qui aide à respecter les exigences de la puissance de calcul demandée dans différents domaines scientifiques, de la géologie, la finance, la physique, la climatologie, etc.

Dans les sections (4.3, 4.4, 4.5 et 4.6) **[Ela 06] [Gli 09] [Glt 10]**, nous avons expliqué le fonctionnement de gLite et de ses différents composants. Dans notre architecture gLite représente l'OS.

4.7.2.1.2. Le moteur Orchestra

Orchestra **[Orc 10]** est une solution pour gérer les processus orientés services de longue durée. Elle fournit des fonctionnalités d'orchestration permettant de gérer des processus métiers complexes. Elle est basée sur le standard BPEL (Business Process Execution Language) est vise à l'amélioration et le contrôle du processus, l'interaction entre les services et la productivité et l'agilité de l'entreprise (l'application). Orchestra est entièrement Open Source et est téléchargeable sous la licence LGPL. Bull **[Bul 97]** est le leader de ce projet

C'est une suite logicielle pour le support du langage BPEL. La suite contient un moteur d'exécution, un designer graphique (pas encore intégré) et un outil de monitoring en temps

réel. Le moteur d'exécution est compatible avec les versions 1.1 et 2.0 du standard BPEL. Il propose également le support de l'extension BPELJ (BPEL pour JAVA) **[Mbl 04]** qui permet de rajouter du code Java dans la définition BPEL du processus métier. Le moteur peut exécuter les processus soit de manière **persistante** ce qui permet de relancer les instances du processus en cours d'exécution en cas de problème, soit de manière non persistante ce qui permet d'obtenir de très bonne performance et peut notamment être utile pour les processus de courte durée. Le moteur permet également de faire à partir d'un processus BPEL des appels non SOAP (par exemple envoyer un mail, …).

Comme l'indique la figure 4.16, l'architecture d'orchestra comprend principalement le conteneur de processus BPEL et le conteneur de web servies dans lequel seront déployés les *grid services* invoqués par le workflow. Ce moteur supporte bien d'autres composants.

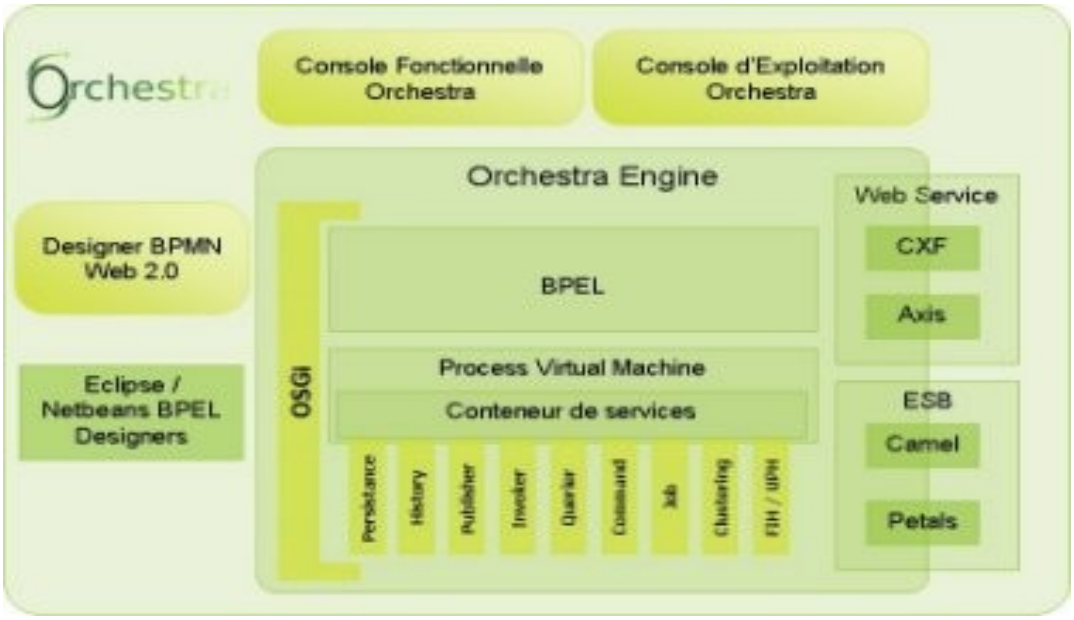

Figure 4. 17: Architecture d'Orchestra [Orc 10]

Orchestra bénéficie de la robustesse du serveur d'application pour fournir un service fiable. De plus, la fonctionnalité de "restart" permet de redémarrer les instances de processus en cours d'exécution en cas de crash machine. **[Nov 04]**

Nous avons mis dans l'annexe B une liste de moteurs d'orchestration commerciaux et open source. Dans notre implémentation, nous avons choisis orchestra comme moteur d'exécution de processus BPEL pour les avantages **[Orc 10]** que ce moteur offre :

- Une solution BPEL open source offrant y compris le moteur, la console d'administration, l'outil de monitoring et prochainement le designer ;

- Orchestra est plus efficace: l'adaptation au contexte du client, la gestion des versions, la fonction de redémarrage, etc. ;
- Intégration non intrusive:
  - Version allégée : déployable sur Tomcat ;
  - Version Entreprise : déployable sur n'importe quel serveur Java EE (JOnAS, JBoss, Weblogic, etc.) ;
  - Version "Sur Mesure": adaptée à tout environnement particulier d'un client
- Orchestra est une solution stable et robuste pour une infrastructure SOA. Une solution BPEL dans une plate-forme SOA est la pièce maîtresse pour gérer l'interaction des services dans un environnement complexe et hétérogène.
- La notion de persistance est un point fort pour les applications scientifiques qui peuvent s'exécuter sur de longues durées.
- Les processus à déployer sur orchestra peuvent être développés avec l'un des designers open source les plus populaire : Eclipse Bpel Designer et Netbeans Bpel Designer.

Ce moteur a été intégré à l'UI comme un composant à part entière.

4.7.2.1.3. Le conteneurAxis2

AXIS est l'implémentation par le groupe Apache du protocole SOAP définie par le W3C et qui en est actuellement à sa deuxième version. La dernière version du projet AXIS1 a été publiée en 2006 **[Fid 11]**. Par la suite le projet AXIS2 a été lancé en 2004 par la fondation Apache qui a complètement repensée et réécrit le Framework **[Fid 11]**.

AXIS2 oriente l'architecture vers un modèle tout XML qui s'avère plus flexible, efficace et configurable en comparaison à AXIS1. Cependant certains concepts ont été conservés dans la nouvelle architecture AXIS2. **[Fid 11]**.

Au niveau des ajouts, AXIS2 supporte maintenant SOAP 1.1 et SOAP 2 ainsi que la méthode REST pour la construction des services web. AXIS2 permet également l'ajout de fonctionnalités basées sur les spécifications WS-* à travers des modules qui viennent se greffer à la couche SOAP à travers des headers spécifiques. L'ajout de modules est permis par la modularité XML. Afin de sécuriser les services web, AXIS2 implémente un certain nombre des spécifications WS-***[Fid 11]**.

Il existe deux implémentations du moteur de services Web Apache Axis2 : Apache Axis2/Java et Apache Axis2/C. Dans notre implémentation nous avons utilisé Apache

Axis2/Java. **[Axi 10]**. Ce choix vient aussi du fait que les environnements de grille de calcul qui supportent l'architecture SOA, implémentent le conteneur Tomcat avec Axis pour le déploiement des services web. Aussi parceque le middleware gLite utilise JAVA dans ses différents composants.

Le propriétaire du conteneur Axis2 est le groupe « GridServiceGroup » qu'on a créé dans le système.

4.7.2.2. Grid services du modèle

Dans cette section, nous présenterons les *grid services* du modèle ainsi que le workflow qu'on a mis en œuvre pour valider le modèle. Nous avons implémenté 5 des services de la grille que nous avons déployé dans le conteneur Axis2, chacun de ces *grid services* suit le modèle de gridification (section 4.7.1.2):

- InitProxyService : pour initialiser le proxy ;
- jobSubmitService : pour la soumission de job ;
- jobStatusService : pour avoir le statut du job soumis ;
- jobOutputService : pour récupérer les sorties du job exécuté ;
- jobCancelService : pour quitter (arrêter) le job soumis ;

Dans le scénario qu'on va exécuter, le *grid service* jobSubmitService permet de soumettre une application en Scilab qui a en entrée une matrice et qui retourne trois fichiers tmp.txt, sol.txt et obj.txt qui contiennent respectivement le temps d'exécution, le résultat1 et le résultat2 (la valeur d'une fonction objective).

Le *grid service* jobOutputService renvoi le chemin vers les sorties de l'application (chemin de tmp.txt, sol.txt et obj.txt).

Pour implémenter et exécuter un workflow, nous avons choisi un scénario de base qui permet l'orchestration de 4 *grid services* :

1. Initialisation du proxy via le *grid service* InitProxy ;
2. Si l'initialisation est faite avec succès, alors invocation du *grid service* jobSubmit pour soumettre le job dont l'application, le jdl, les inputs, et les outputs sont prédéfinis. Sinon, afficher un message d'erreur ;
3. Si la soumission est faite avec succès alors invocation du *grid service* jobStatus pour récupérer l'état du job soumis. Sinon, afficher un message d'erreur ;
4. Boucler sur l'invocation du *grid service* jobStatus jusqu'à ce que le statut soit "Done", "Aborted" ou "en cas d'erreur". Si l'état retourné est "Done", alors invocation du *grid service* jobOutput pour récupérer les résultats du job soumis. Sinon, Afficher un message d'erreur ;

5. Si la récupération des résultats est faite avec succès, alors renvoi du chemin des résultats au client. Sinon, afficher un message d'erreur.

La figure 4.18 présente un diagramme de séquence pour l'exécution qu'on a décrit ci-dessus:

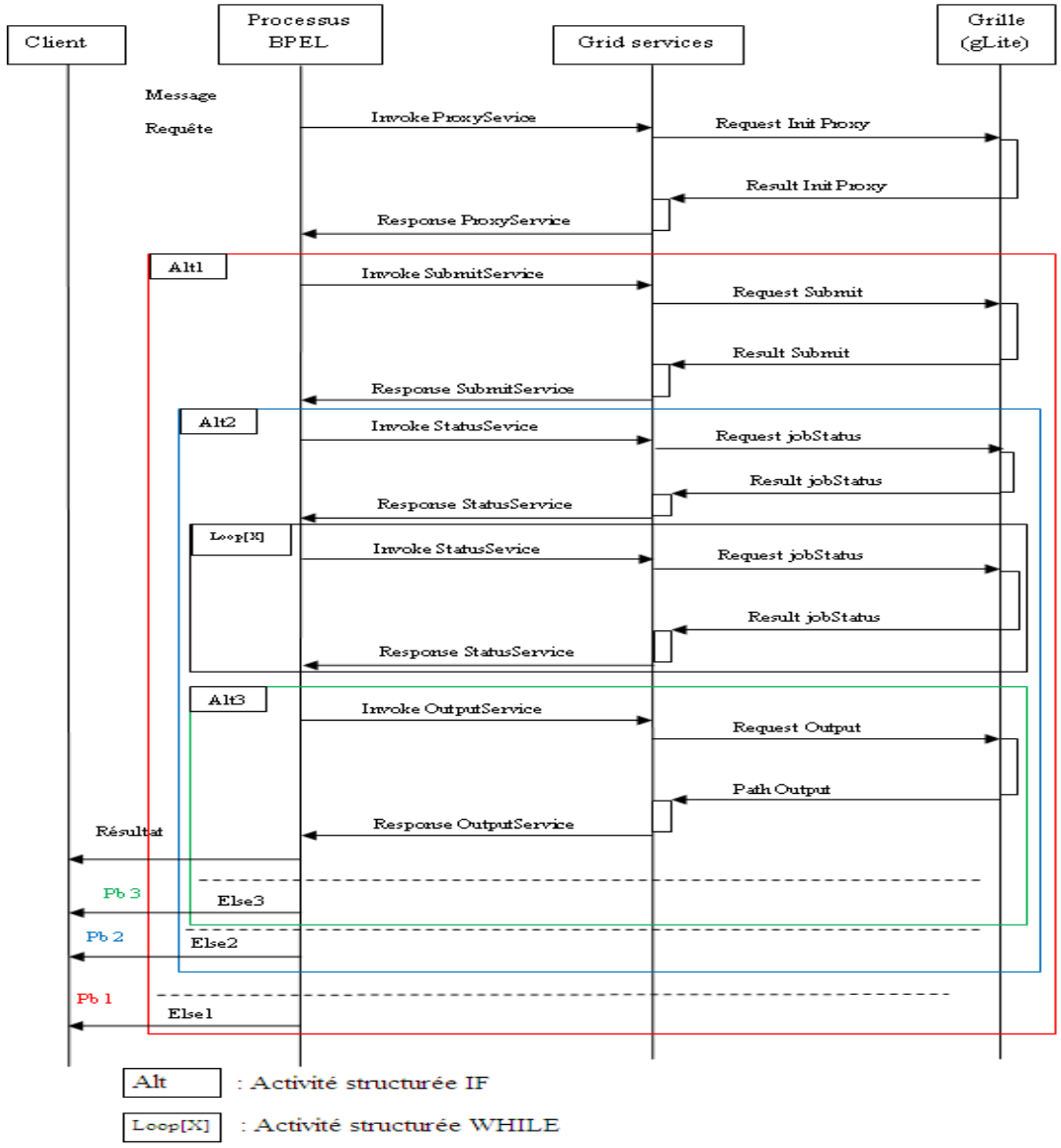

Figure 4. 18: Diagramme de séquence du workflow

4.7.2.3. Les langages de développement

Pour implémenter le modèle de gridification, nous avons opté pour les langages suivants :

4.7.2.3.1. BPEL

Comme nous l'avons vu dans le chapitre 3, BPEL **[Tan 03] [Mbj 06]** est devenu la norme dominante pour l'orchestration de services Web. Avec BPEL, les services Web peuvent être intégrés, en utilisant des grammaires XML, pour créer une application de niveau supérieur. La grammaire XML qui définit un processus BPEL est interprétée et exécutée par un moteur d'orchestration qui expose le processus en tant que service Web. BPEL, qui est construit sur XML-Schema, WSDL et XPath, tisse des activités basiques et structurées de manière à créer la logique du processus. BPEL fournit de nombreuses constructions pour la gestion des activités du processus, y compris les boucles, les branchements conditionnels, le traitement des erreurs, et la gestion des événements (tels que le **timeout**). Il permet aux activités d'êtres exécutées de façon séquentielle ou parallèle.

Le processus BPEL peut gérer les exceptions générées par les appels de service. Ces exceptions peuvent être générées en externe par les services Web partenaires invoqués ou à l'intérieur du processus.

Le fichier BPEL définit le processus, ou l'enchaînement et la logique des actions qui seront exécutées par le moteur d'orchestration. La structure du fichier BPEL est la même que celle du processus. Ce fichier est véritablement le code source de l'application que constitue le processus, le moteur d'orchestration agissant comme une machine virtuelle capable d'exécuter le code BPEL. Il est constitué des balises suivantes (obligatoires et non) :

- **La balise <process> :** est l'élément racine (au sens XML) du fichier BPEL. C'est à l'intérieur de cette balise que se retrouve la description complète du processus. Grâce à l'attribut name, on peut donner un nom au processus.

```
<process name="processName"
        xmlns=http://docs.oasis-open.org/wsbpel/2.0/process/executable
        targetNamespace=http://example.com
         xmlns:tns="http://example.com"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema">
         [...]
</process>
```

- **La balise <import> :** Cette balise permet d'importer un fichier WSDL par exemple:

```
<import
namespace="http://example.com"
location="fichier.wsdl" importType="http://schemas.xmlsoap.org/wsdl/" />
```

- **La balise <partnerLinks> :** Cette balise permet de lier des actions définies dans le fichier WSDL (via partnerLinkType) au process BPEL. L'attribut myRole ou partnerRole définit si c'est une action qui appelle le processus ou si c'est une action appelée par le processus.

```
<partnerLinks>
  <partnerLink name="PartnerLink1" partnerLinkType="tns:examplePL" myRole="exampleRole" />
</partnerLinks>
```

- **La balise <variables> :** Cette balise permet de définir les variables utilisées par le processus.

```
<variables>
        <variable name="var" messageType="tns:exampleMessage" />
</variables>
```

- **La balise <sequence> :** Cette balise va contenir des actions ou de la structure directement liée à l'exécution du processus.

```
<sequence name="Main">
  [Actions]
</sequence>
```

- **La balise <receive> :** Cette balise permet de recevoir un signal de l'extérieur du processus. Cela permet d'instancier un processus par exemple, ou plus généralement d'attendre qu'un évènement se termine avant de continuer le processus.

```
<receive name="Receive1" createInstance="yes" PartnerLink="PartnerLink1"
operation="exampleOperation" portType="examplePortType" variable="var1In"/>
```

- **La balise <reply> :** Cette balise permet de renvoyer une réponse à un partnerLink qui en attend une. D'abord le receive, puis, après le traitement, le reply.

```
<reply name="Reply1" PartnerLink="PartnerLink1" operation="exampleOperation"
portType="examplePortType" variable="var1Out"/>
```

- **La balise <invoke> :** Cette balise permet d'appeler un service web. Elle utilise un partnerLink "sortant" et peut ou non recevoir une réponse.

```
<invoke name="invoke1" PartnerLink="PartnerLink1" operation="exampleOperation"
portType="examplePortType" inputVariable="varIn" outputVariable="varOut"/>
```

- **La balise <forEach> :** La balise *forEach* permet d'effectuer une boucle. On déclare un compteur (variable qui sera incrémentée à chaque itération), une valeur de départ et une valeur finale pour ce compteur. La boucle peut s'exécuter en mode parallèle.

```
<forEach name="superBoucle" parallel="yes" counterName="index">
  <startCounterValue>1</startCounterValue>
  <finalCounterValue>$uneVariable</finalCounterValue>
  [Instructions]...
</forEach>
```

Dans cet exemple, si *$uneVariable* est égale à 3, la boucle sera exécutée 3 fois. Le compteur peut être appelé comme n'importe quelle variable (ici *$index*).

- **La balise <while> :** Boucle.

```
<while>
  <condition>
    [Condition en utilisant XPath]
  </condition>
  <sequence>
    [Séquence à executer]
  </sequence>
</while>
```

- **La balise <repeatUntil> :** Boucle

```
<repeatUntil>
  <sequence>
    [Séquence à executer]
  </sequence>
  <condition>
    [Condition en utilisant XPath]
  </condition>
</repeatUntil>
```

- **La balise <correlationSet> :** Lorsque plusieurs actions sont lancées en mode parallèle, il peut être indispensable de les relier entre elles, par exemple, un receive peut être lié à un invoke. La balise *correlationSet* permet de lier des invoke et des receive entre eux. Il ne peut y avoir qu'un seul *correlationSet* par process, donc lorsqu'on veut utiliser des *correlationSet* dans un *forEach* en mode parallèle, le *correlationSet* doit être défini dans le *forEach*.

```
<correlationSet name="CorSetEx" properties="exns:propEx"/>
```

- **La balise <flow> :** cette balise permet l'exécution parallèle d'une ou plusieurs activités

```
<flow standard-attributes>
        [Action à executer en parallele]
</flow>
```

- Il y a aussi les balises **<scope>**, **<if>**, **<throw>**, **<rethrow>**, **<exit>**, **<pick>** et bien d'autres pour gérer tout le processus d'orchestration.

Nous avons choisi BPEL pour notre implémentation, car c'est un langage normalisé qui fournit un soutien pour de nombreuses fonctionnalités et il est très expressif. De plus il a fait ses preuves dans les environnements de grilles de calcul et dans la modélisation des applications scientifiques (bioinformatique, mathématique, physique, ..) comme nous l'avons vu dans **[Kll 06]**, **[Ija 08]**, **[Oez 07]**, **[Tdo 07]**, **[Pjg 07]**, **[Wem 05]** et d'autres travaux qu'on n'a pas cité.

4.7.2.3.2. JDL[83]

C'est le langage de description de job à soumettre dans la grille qu'on a vu dans la section 4.3.3.

4.8. Expérimentation et validation

Pour expérimenter ce modèle, nous avons modélisé le scénario de la section 4.7.2.2 en workflow avec BPEL. Nous avons déployé le workflow et nous l'avons exécuté sur orchestra. De plus nous avons discuté les points dont le moteur d'orchestration doit assurer pour le bon déroulement de l'application.

4.8.1. Modélisation du workflow scientifique

La définition (modélisation) d'un workflow scientifique consiste en la définition des activités de base, des flux de données, des flux de contrôle, de la composition hiérarchique et de la gestion des erreurs. Nous avons utilisé Eclipse BPEL Designer[84] pour modéliser (définir) le processus décrit dans la section 4.7.2.2. La figure 4.19 représente le processus définit.

[83] JDL : Job Description Language
[84] Bpel Designer : http://www.eclipse.org/bpel

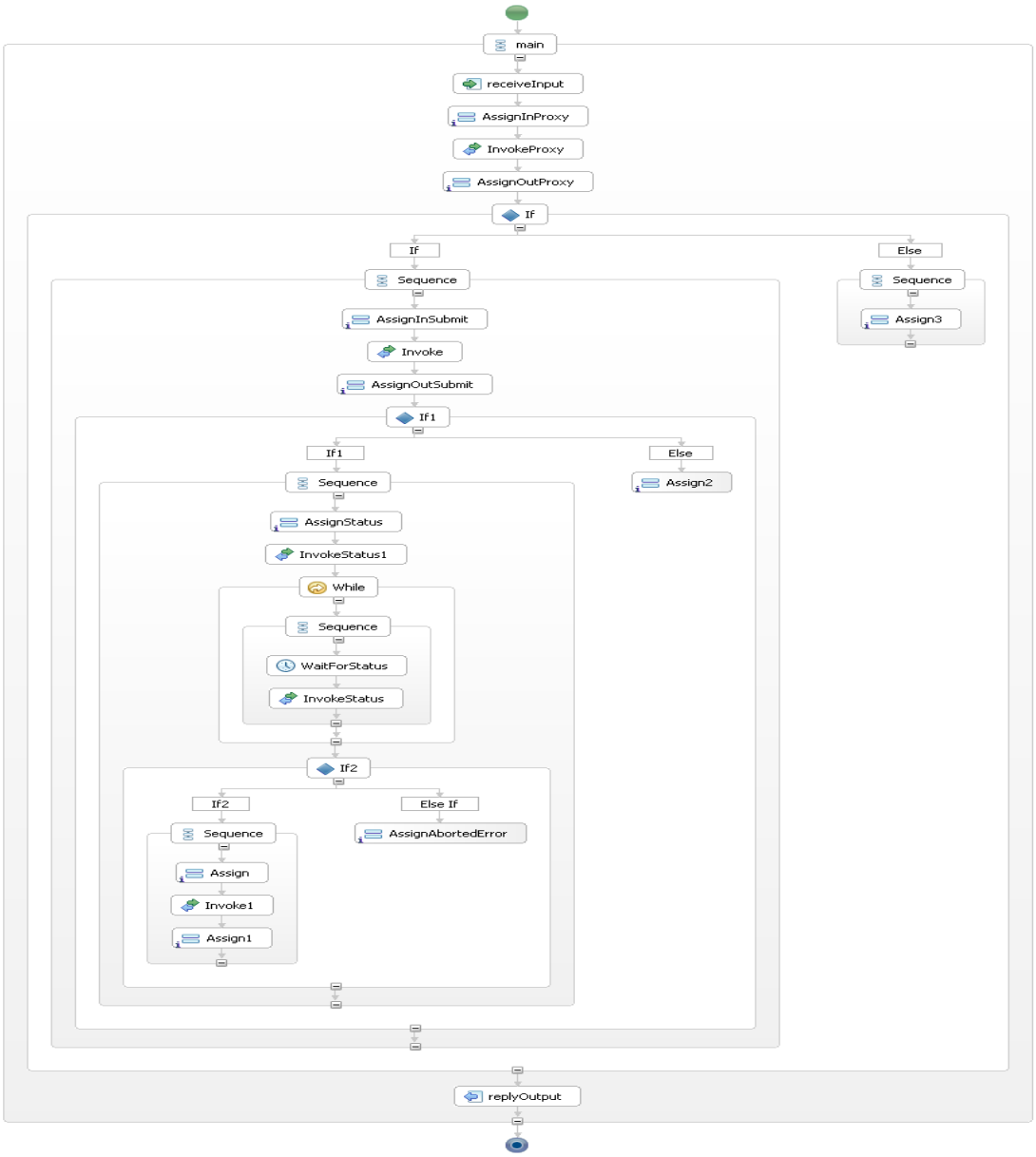

Figure 4. 19: Processus BPEL

4.8.1.1. Les Activités de base

Les activités que la définition du workflow devrait être en mesure de déterminer incluent l'invocation d'un *grid service*, la synchronisation, le rassemblement du contenu du message qui doit être transmis à un *grid service*, l'extraction des résultats à partir des réponses du *grid service*, et la signalisation et la réponse aux erreurs.

4.8.1.1.1. Invocation de service <invoke>

L'activité de base la plus importante dans BPEL est utilisée pour invoquer les services.

```
<bpel:invoke
        name="InvokeJobSubmit"
        partnerLink="SubmitPL"
        operation="JobSubmitBPEL"
        portType="ns0:submitBPELWSServicePortType"
        inputVariable="SubmitPLRequest"
        outputVariable="SubmitPLResponse">
</bpel:invoke>
```

Les partenaires doivent définir les types de port WSDL (portType) pour chaque interface qui est utilisée dans la définition du workflow. De plus, BPEL a besoin des définitions WSDL des services pour fournir un lien partenaire (partnerLink), en précisant un rôle à chaque interface (portType). Le workflow BPEL peut référencer ces partenaires-link-rôles pour décrire comment une interface est utilisée dans une interaction entre le workflow et un service partenaire, par exemple, de faire une distinction entre une interface qui est nécessaire pour envoyer des requêtes et une interface utilisée pour envoyer des messages de rappel à l'instance du workflow.

4.8.1.1.2. Synchronisation

L'invocation d'un service peut être soit synchrone ou asynchrone. Les deux sont définis par le processus BPEL. Les invocations qui donnent à la fois les variables d'entrée et de sortie sont exécutées de manière synchrone et le workflow est bloqué tant que le service s'exécute. Par conséquent, le code ci-dessus est un appel synchrone, ce qui est approprié car l'opération JobSubmitBPEL retourne des contrôles dès qu'elle a traité la soumission de job.

4.8.1.1.3. Réception <receive>

Chaque processus BPEL est, en fait, un service web à part entière. Ce service peut être invoqué par l'envoi d'un message SOAP au moteur d'orchestration. Nous exploitons ce mécanisme pour structurer hiérarchiquement des workflows scientifiques qui peuvent être complexes, et les rendre modulaires et réutilisables individuellement. La primitive utilisée pour atteindre cet objectif est la balise <receive>, comme indiqué ci-dessous. La déclaration affirme que le processus implémente l'opération **process** et le type de port ***processBP***. Ce processus déclare aussi que lors de la réception d'une requête il stocke les paramètres pour le message dans une variable appelée ***input***. En outre, chaque fois que **process** est invoquée, une instance du processus est créée qui engendre un nouveau processus BPEL qui s'exécute en même temps que tous les processus précédemment créé qui n'ont pas encore terminé.

```
<bpel:receive
    name="receiveInput"
    partnerLink="client"
    portType="tns:processBP"
```

```
        operation="process"
        variable="input"
        createInstance="yes"
/>
```

4.8.1.1.4. Affectation <assign>

Une autre activité de base importante nous permet de manipuler les données stockées dans des variables. Ceci est fait en utilisant les affectations. Une affectation consiste en un certain nombre de balises <copie> qui copie les données d'une source à une destination cible. La source peut être des données XML (donnée comme un littéral), le résultat d'évaluation d'une expression, une requête XPath qui extrait des données d'une autre variable, ou le résultat d'appel d'une procédure dans un langage de programmation tel que Java ou JavaScript qui peuvent être incorporés en utilisant le mécanisme d'extension BPEL. Le code suivant représente la notion d'affectation.

```
<bpel:assign validate="no" name="AssignOutProxy">
        <bpel:copy>
          <bpel:from><bpel:literal><tns:processBPResponse xmlns:tns="grid"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                  <tns:result>tns:result</tns:result>
                 </tns:processBPResponse>
                 </bpel:literal>
          </bpel:from>
          <bpel:to variable="output" part="payload"></bpel:to>
        </bpel:copy>
        <bpel:copy>
          <bpel:from part="parameters" variable="ProxyPLResponse">
            <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                <![CDATA[ns:return]]>
            </bpel:query>
          </bpel:from>
          <bpel:to part="payload" variable="output">
            <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                <![CDATA[tns:result]]>
            </bpel:query>
          </bpel:to>
        </bpel:copy>
 </bpel:assign>
```

4.8.1.1.5. Autres activités de base

BPEL comprend un certain nombre d'autres activités, comme l'attente d'une période donnée (la balise <wait>). L'exemple ci-dessus permet une attente de 10 secondes avant l'invocation du *grid service*.

```
<bpel:wait name="WaitForStatus">
        <bpel:for><![CDATA['PT10S']]></bpel:for>
</bpel:wait>
```

4.8.1.2. Les flux de données

Du point de vue des **flux de données**, les scientifiques doivent être en mesure de déclarer des variables pour le stockage temporaire des entrées / sorties des appels de services. Ces déclarations doivent être flexibles et déterministes. Le workflow doit définir comment la donnée qui est utilisée comme une entrée pour une requête de service est assemblée et même comment les parties pertinentes des données de sortie sont extraites.

4.8.1.2.1. Les Variables <variable> et <variables>

Des types peuvent être utilisés pour déclarer des variables locales. Ces variables sont utilisées pour gérer les flux de données entre les différents appels de *Grid services*. En attribuant des types à ces variables, il devient alors possible de valider que les données transmises vers et à partir d'un *Grid service* en utilisant des messages SOAP sont compatibles avec les paramètres déclarés dans le WSDL du service, ce qui réduit les possibilités d'erreurs. L'exemple ci-dessus montre l'utilisation de ces variables.

4.8.1.2.2. Champ d'application structuré <scope>

Dans la pratique, il est nécessaire d'utiliser un grand nombre de variables (section 4.8.1.2.1). Il est parfois commode de pouvoir utiliser les mêmes noms pour les différentes variables, par exemple s'ils doivent être utilisés dans un certain nombre de sous-processus qui s'exécutent en parallèle. Afin d'éviter les interférences entre ces processus, BPEL fournit la notion de champ d'application (la balise **<scope>**). Cela évite d'avoir à inventer des noms artificiels pour ces variables. <scope> prévient non seulement les conflits de noms, mais elle permet aussi au moteur d'orchestration de décider quand les variables ne seront plus nécessaires.

```
<bpel:scope  name="SubmitScope" >
    <bpel:variables>
        <bpel:variable name="SubmitPLRequest"  messageType="ns0:JobSubmitBPELRequest">
        </bpel:variable>
     …
    </bpel:variables>
</bpel:scope>
```

4.8.1.3. Les flux de contrôle

Le flux de contrôle devrait permettre le regroupement d'activités de base dans des séquences d'activités ainsi que l'exécution répétitive et conditionnelle de ces séquences. Pour le calcul scientifique, la possibilité de spécifier l'exécution simultanée de séquences d'activités est d'une importance primordiale. BPEL fournit des primitives de contrôle qui représentent

des activités structurées telles que <sequence>, <while>, <switch>, <if> et <pick>. <sequence>, <while>, <switch> et <if> ont la sémantique classique. <pick> est utilisée pour le choix non-déterministes.

L'exemple ci-dessous définit un processus en attente de la terminaison de la requête pour récupérer le statut du job soumis. Cette loupe se termine si l'une des trois possibilités soit vraie (Statut Done, Aborted ou dans le cas d'une erreur).

```
<bpel:while name="WhileStatus">
        <bpel:condition><![CDATA[$StatusPLResponse.parameters/ns2:return!="Done" and
$StatusPLResponse.parameters/ns2:return!="Error" and
$StatusPLResponse.parameters/ns2:return!="Aborted" ]]>
        </bpel:condition>
        <bpel:sequence>
                <bpel:wait name="WaitForStatus">
                        <bpel:for><![CDATA['PT10S']]></bpel:for>
                </bpel:wait>
                <bpel:invoke
                        name="InvokeStatus"
                        partnerLink="StatusPL"
                        operation="JobStatusBPEL"
                        portType="ns2:statusBPELWSServicePortType"
                        inputVariable="StatusPLRequest"
                        outputVariable="StatusPLResponse">
                </bpel:invoke>
        </bpel:sequence>
</bpel:while>
```

Un autre exemple sur les activités structurées est la balise <if> que nous avons utilisé pour invoquer le *grid service* jobOutput ou renvoyer une erreur :

```
<bpel:if name="IfStatusDone">
        <bpel:condition><![CDATA[$StatusPLResponse.parameters/ns2:return="Done"]]>
        </bpel:condition>
        <bpel:sequence>
                        …
                        <bpel:invoke
                            name="InvokeOutputService"
                            partnerLink="OutputPL"
                            operation="JobOutputBPEL"
                            portType="ns3:outputBPELWSServicePortType"
                             inputVariable="OutputPLRequest"
                            outputVariable="OutputPLResponse">
                        </bpel:invoke>
                        …
        </bpel:sequence>
<bpel:elseif>
        <bpel:condition><![CDATA[$StatusPLResponse.parameters/ns2:return="Aborted" or
$StatusPLResponse.parameters/ns2:return="Error"]]>
        </bpel:condition>
          …
</bpel:elseif>
</bpel:if>
```

4.8.1.4. La composition hiérarchique

Les définitions de workflows scientifiques peuvent paraitre raisonnablement complexes. Cette complexité doit être maîtrisée. Ceci peut être réalisé grâce à l'introduction de l'abstraction. Il devrait être possible de traiter la définition d'un workflow comme un seul *grid service* de sorte que ce *Grid service* peut ensuite être utilisé dans un workflow supplémentaire. De cette manière, des workflows scientifique peuvent être composés de plusieurs workflow de base, s'appuyant sur des services qui sont eux-mêmes des workflows. Dans notre cas les *grid services* jobStatut et jobOutput peuvent constitués un sous-workflow dans le workflow principal. Ce sous-workflow peut être réutilisé dans d'autres applications. De même le workflow principale peut être utilisé dans d'autres workflows plus complexes.

4.8.1.5. La gestion des erreurs

Il est dans la nature du calcul distribué que les activités échouent et cela est certainement vrai pour les *grid services*. Ainsi, la définition de workflow devrait supporter la définition des mécanismes de gestion des erreurs, par la détection des exceptions et l'application de traitements selon que l'exception soit simple (exemple : re-soumission des jobs) ou complexe (exemple : cohérence d'un état répartis sur différentes bases de données en cas d'échec).

```
<bpel: faultHandlers>
  <bpel: catch
        faultName ="InvokeExeception"
        faultContainer ="error" >
        <bpel: sequence name ="fault-sequence">
        …
```

4.8.2. Déploiement du workflow scientifique

Les workflows peuvent être exécutés sur la même machine où ils ont été définis. Ceci est approprié si les workflows sont de courte durée et sont changés très souvent. Si les workflows sont des services qui sont utilisés par d'autres workflows, ils sont longs en exécution, et doivent être hautement évolutive, ils pourraient être mieux exécutés sur des serveurs dédiés. Ils doivent alors être transférés de la machine sur laquelle ils ont été définis sur le serveur où ils peuvent être interprétés par un interprète. Le transfert est connu sous le terme **déploiement** et l'interprète sous le terme **moteur** (engine) de workflows. Dans notre cas, on a utilisé Eclipse[85]+ode[86] pour le déploiement et l'exécution en locale, puis orchestra

[85] Plate forme de développement Eclipse : http://www.eclipse.org
[86] APACHE ODE : moteur d'exécution BPEL, http://ode.apache.org/

comme serveur dédié. Nous montrons sur la figure 4.20, l'exécution du workflow avec Eclipse et ODE.

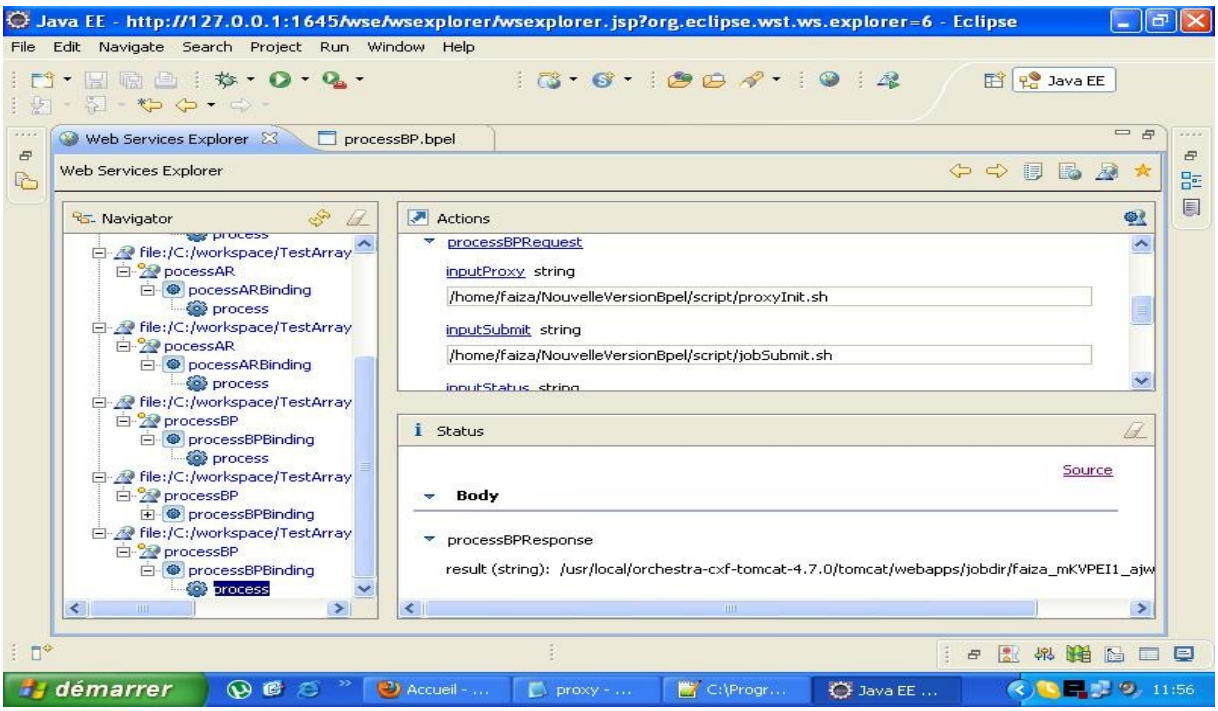

Figure 4. 20: Exécution du workflow sur Eclipse+ODE

4.8.2.1. Test des grid services

Avant le déploiement et l'exécution du workflow scientifique, les *grid services* intervenants dans ce workflow doivent être testés individuellement. Idéalement, les tests devraient être entièrement automatisés utilisant, par exemple, des clients (java, c, …) ou encore la plate forme Eclipse pour les services web. Nous avons déployé les *grid services* que nous avons développés dans le conteneur Axis2. Pour tester ces *grid services* individuellement, nous avons développé des clients Java qui s'intègrent dans l'environnement de gridification.

La figure 4.21 montre les services déployés sur la grille.

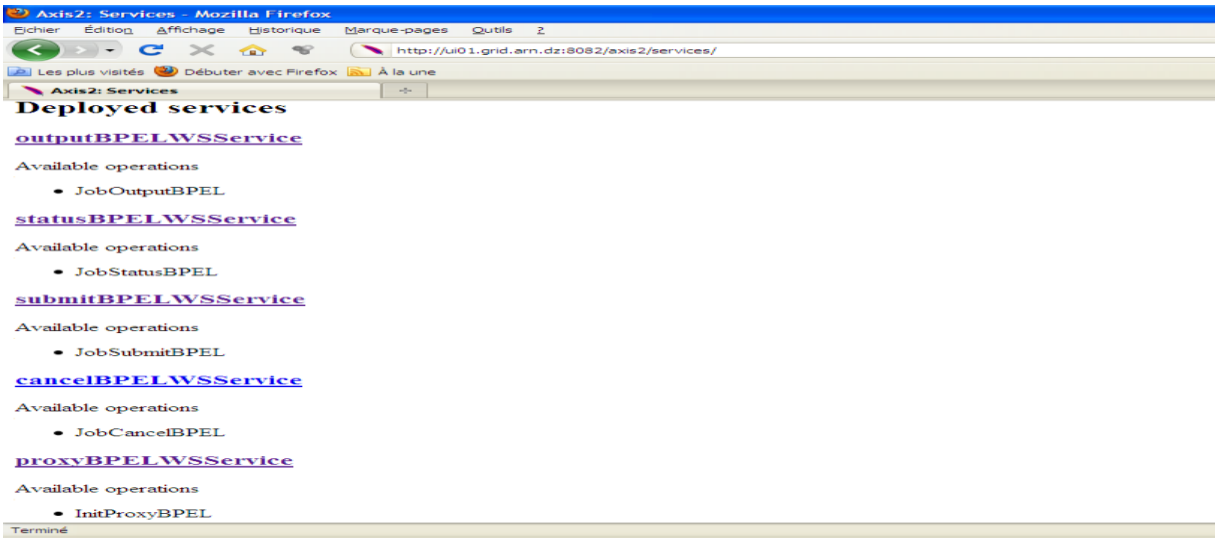

Figure 4. 21: Grid services déployés sur Axis2

4.8.2.2. Déploiement

Le workflow doit être auto-déployé. Pour cela, le moteur de workflows doit être capable de lire une archive qui contient toutes les descriptions nécessaires au workflow. Il devrait être possible de déployer rapidement de nouvelles définitions du workflow sans avoir à redémarrer le moteur de workflows. Cela est encore plus important dans le calcul scientifique du fait que les workflows sont souvent longs en exécution et les moteurs de workflows peuvent être utilisés pour plus d'un processus à la fois. Ce serait surtout un inconvénient si les nouveaux processus ne pouvaient être déployés que lorsque tous les autres processus ont été achevés.

Orchestra répond à cette exigence en offrant deux façons pour déployer un processus, la première en utilisant une archive (**.bar**) qui contient le fichier BPEL et les différents fichiers WSDL (celui du processus et ceux des services partenaires) et la deuxième en utilisant la ligne de commande. L'archive peut être uploadé via une console d'administration (figure 4.22).

Une fois l'archive uploadée, orchestra déploie automatiquement le processus et si un fichier est changé, orchestra met à jour le processus déployé. Cela permettra une intégration relativement simple du processus de déploiement dans un environnement de grille par exemple par l'ajout d'un *grid service* qui effectue l'ensemble de l'opération.

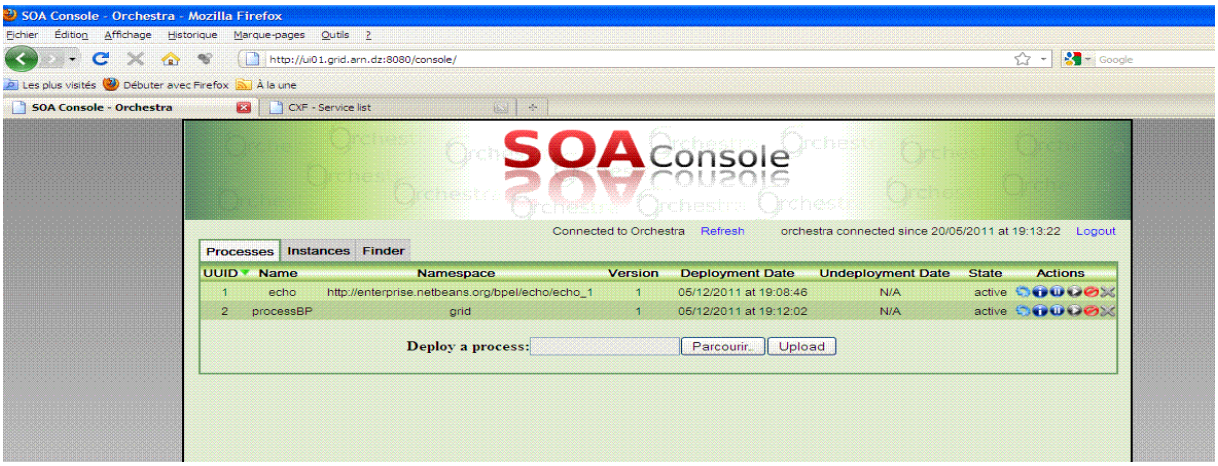

Figure 4. 22: Uploade d'archive via la console

4.8.3. Exécution du workflow scientifique

4.8.3.1. Exécution

Le moteur de workflows doit supporter l'appel à distance d'un workflow en exposant une interface WSDL afin qu'il puisse être invoqué à distance par l'envoi d'un message SOAP. Une fois un tel message reçu, le moteur devrait démarrer le traitement du workflow en exécutant les flux de contrôle et en gérant les flux de données décrits dans le workflow. Orchestra expose les processus Bpel déployés via des interfaces WSDL comme des services web. Pour tester l'exécution de notre scénario, nous avons généré un client java avec Eclipse (utilisant la commande wsdl2java d'axis) pour invoquer le processus BPEL. Ce dernier retournait le chemin vers les résultats du job soumis.

La figure 4.23 présente le WSDL du workflow déployé sur orchestra.

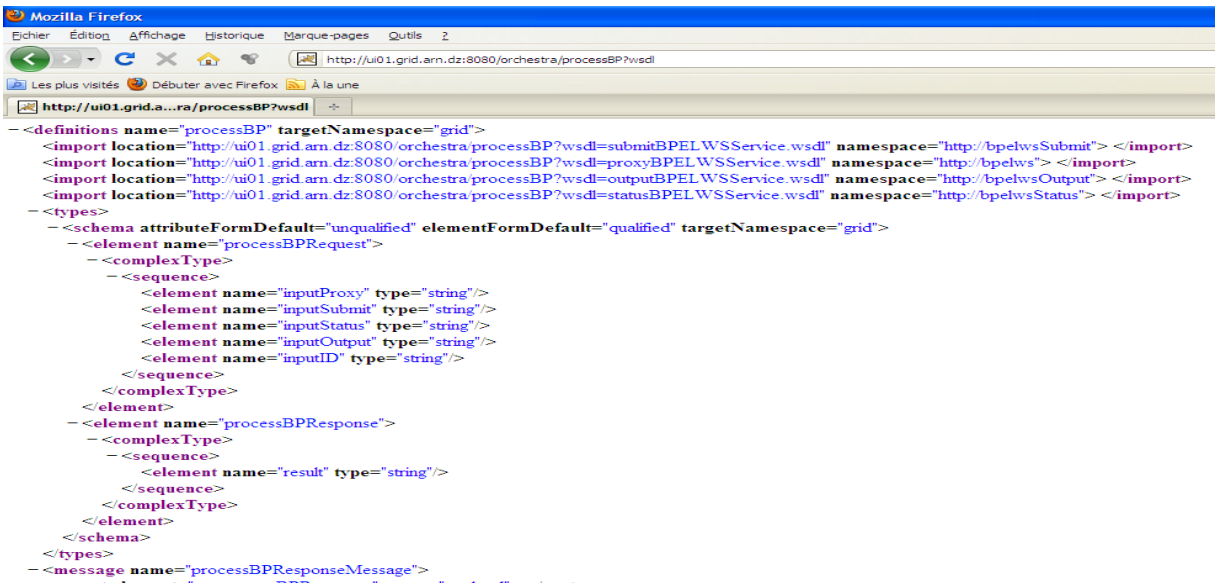

Figure 4. 23: WSDL du workflow, exposé par orchestra engine

4.8.3.2. Concurrence

Le moteur doit également supporter l'exécution simultanée de workflows identiques ou différents. Il est tout à fait concevable que différents scientifiques exécutent le même workflow avec différents paramètres, donc le moteur doit être capable de créer une nouvelle instance du workflow lors de la réception d'une invocation. De même, différents scientifiques voudront déployer et exécuter différents workflows en même temps. Orchestra permet la création d'une nouvelle instance pour chaque appel du processus, comme le montre la figure 4.24.

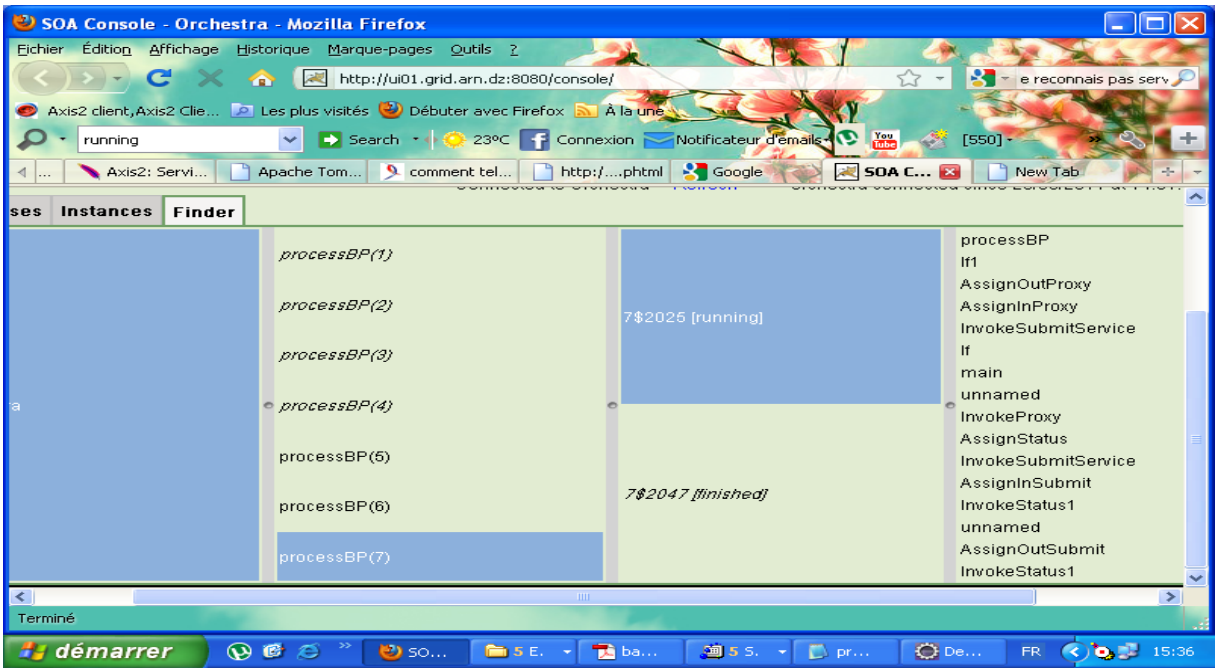

Figure 4. 24: Concurrence entre deux instances du même processus

4.8.3.3. Monitoring

Les workflows scientifiques sont potentiellement long d'exécution. Il est donc d'une importance cruciale pour les scientifiques d'être en mesure d'observer et de surveiller (monitorer) l'exécution en cours d'un workflow. Il est utile d'être en mesure de voir quelles activités du workflow ont été effectuées, lesquelles sont en cours et celles qui ont échoué. Dans ce dernier cas il est utile d'être capable d'observer les conditions exceptionnelles qui ont provoqué l'échec. La console d'administration d'orchestra permet le suivi des processus déployés, des instances en cours d'exécution et des activités de chaque instance.

- La figure 4.22 liste tous les processus dans orchestra, en spécifiant : le nom, l'identifiant dans l'engine, la version, la date de déploiement, la date d'indéploiement, l'état et les actions à effectuer sur le processus ;

- La figure 4.24 représente le monitoring du processus en cours d'exécution dans orchestra. La console offre la possibilité de monitorer le processus en visualisant ses différentes instances ainsi que les activités exécutées en temps réel pour chaque instance ;
- La figure 4.25 montre les différentes activités du processus ainsi que leurs types.

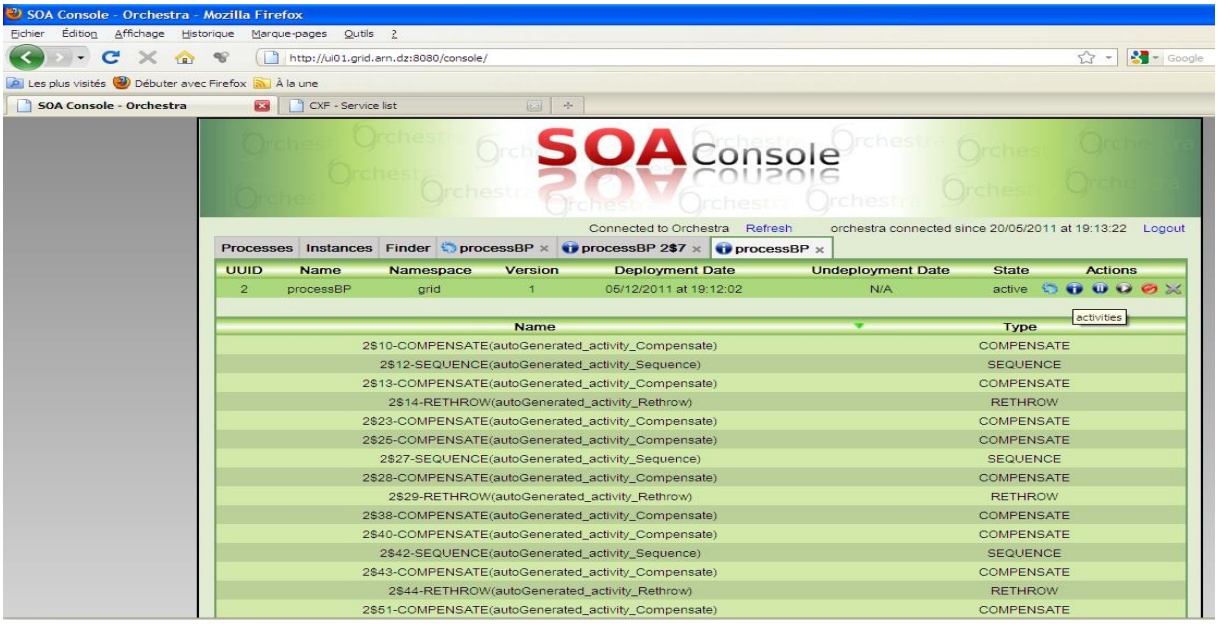

Figure 4. 25: Activités du workflow déployé

Pour le suivi des variables et de leur contenu, orchestra affiche les contenus à l'instant 't' via sa console de lancement mais n'affiche pas sur la console d'administration. Dans le cas d'un échec, la console d'administration d'orchestra lance un pop-up contenant l'exception rencontrée. Cette même exception sera sur la console de lancement.

4.8.4. Fiabilité

La fiabilité et la tolérance aux fautes sont importantes pour les workflows scientifiques qui sont souvent longs d'exécution. Il y a différents types de fautes que le moteur doit tolérer. Il s'agit entre autres de la capacité à gérer les fautes (échecs) des services orchestrés, les erreurs de communication avec les services orchestrés, … Nous allons discuter de la façon de faire face à ces défaillances.

Orchestra implémente BPEL 2.0, y compris la gestion des erreurs, qui peut être utilisée pour définir dans le workflow comment le moteur doit se comporter si l'un des services orchestrés échoue. Orchestra est aussi configuré pour la persistance des processus en cours d'exécution (l'état des activités en cours et l'état du processus) et cela grâce au système de base de données. Orchestra s'installe par défaut avec le système de fichier H2, mais il supporte les systèmes Oracle, HSQL, PostgreSQL et MySQL.

Dans notre implémentation, nous avons configuré orchestra avec MySQL car ce dernier est adopté comme système de gestion de base de données dans les nœuds de gLite.

4.9. Conclusion

Dans cette dernière étape de notre travail, nous avons proposé un modèle de gridification et nous avons réalisé un prototype du modèle proposé. A cet effet, nous avons utilisé plusieurs technologies notamment le middleware gLite, et les technologies des services web. Nous avons présenté les différents aspects nécessaires à la bonne exécution d'un workflow scientifique et surtout ce que doit offrir le moteur d'orchestration.

Ce travail nous a permis d'acquérir une certaine maitrise de ces hautes technologies durant cette période. Bien entendu, l'objectif que nous avons tracé au départ a été globalement atteint. Le prototype proposé a été testé et a donné des résultats sur lesquels nous nous baserons pour améliorer le système. Notre objectif est de pouvoir faire bénéficier nos scientifiques sur l'échelle nationale et éventuellement à l'échelle régionale de ces technologies et de cette architecture.

CONCLUSION GÉNÉRALE

La technologie de grille de calcul est adoptée dans un grand nombre de domaines scientifiques, notamment, la physique, l'astronomie, le biomédical, l'observation de la terre et climatologie, la nanotechnologie, le géo-spatial, et bien d'autres. Arrivée à un certain point de maturité, cette technologie a exprimé le besoin de normalisation, ce qui a donné naissance aux standards OGSA et WSRF qui sont basés sur les technologies des services web dans des environnements de grille de calcul. La combinaison de ces différentes technologies (grille et services web) s'est communément propagée sous le nom de "*grid services*".

Un nombre important de travaux se sont orientés vers la notion d'orchestration de *grid services* pour la modélisation des workflows scientifiques en utilisant le langage d'orchestration BPEL. Dans ces travaux, l'orchestration s'est faite avec des *grid services* basés sur les standards OGSA, WSRF ou de simples services web. Les auteurs ont utilisé le middleware Globus ToolKit 3 ou 4 qui implémente respectivement OGSA et WSRF.

La contribution majeure de notre travail est la mise en place d'une infrastructure qui combine les technologies de grille et de services web et permet le déploiement de workflows scientifiques modélisés avec le langage BPEL. Cette infrastructure offrira aux utilisateurs de notre grille nationale, DZ-Grid, plus de souplesse et de flexibilité dans la gridification de leurs applications scientifiques.

Pour ce faire, nous avons fait une étude sur ces différentes technologies. Nous avons présenté les différents aspects et composants de la technologie de grille, nous avons détaillé les composants et les architectures des technologies des services web et nous avons expliqué l'apport de la composition de services web dans la modélisation des applications scientifiques. Dans notre étude nous nous sommes basés sur notre grille nationale, DZ-Grid, et par conséquent sur le middleware gLite. Ce dernier étant orienté SOA mais ne supportant pas encore les standards OGSA et WSRF, nous avons proposé un modèle de gridification pour le déploiement et l'exécution des *grid services* et des workflows scientifiques en combinant le middleware gLite, le conteneur de services web, Axis2, et le moteur d'orchestration, orchestra. Nous avons atteint notre objectif initial qui a été validé par une expérimentation sur l'infrastructure composite avec un workflow que nous avons implémenté. Un workflow pouvant être composé de plusieurs sous-workflows, le workflow implémenté pourra être utilisé comme un sous-workflow dans un workflow plus complexe.

Nous avons mis en évidence les différents aspects à respecter pour le bon déploiement et exécution du workflow dans cette nouvelle infrastructure.

Après qu'on ait atteint une certaine maitrise des différentes technologies intervenantes dans la gridification des applications via la composition de *grid services*, la continuité de ce travail portera principalement sur certains points que nous présentons ci-dessous.

En premier lieu, nous gridifierons un workflow complexe décrivant une solution d'optimisation en Matlab (Recherche Opérationnelle) basée sur le paradigme de "Branch-and-Bound", qui consiste à effectuer un parcours implicite et intelligent d'une arborescence de sous-ensembles de solutions afin de découvrir les solutions optimales. Ce workflow complexe est constitué de sous-workflows qui s'exécutent en parallèle. Cette gridification nous permettra d'estimer la performance et l'évolutivité du modèle proposé et du moteur d'orchestration qu'on a intégré.

Nous pensons aussi à l'intégration d'un modèle de fabrique de *grid services* qui seront exploités pour la gestion des données dans la grille "Data Management grid services" et la publication de tous ces *grid services* via le système d'information de la grille, notamment, le modèle R-GMA qui représentera un registre UDDI, pour publier les *grid services* disponibles sur la grille.

D'autre part, nous envisageons d'intégrer un designer BPEL dans le modèle qui sera exploitable en ligne pour avoir plus de flexibilité dans la mise à jour des workflows.

Enfin, puisque WSRF permet la modélisation et l'accès aux ressources avec état en utilisant les services web et étant donné que gLite ne supporte pas encore cette norme, nous pensons à proposer une extension de notre architecture qui utilisera les spécifications WSRF pour la gestion des ressources de la grille avec les services web.

BIBLIOGRAPHIE

[Aar 02] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi, I. Trickovic, S. Zimek. *"Web Service Choreography Interface (WSCI) 1.0"*. W3C Note, 2002. http://www.w3.org/TR/wsci

[Aba 05] A. Barros, M. Dumas, P. Oaks. *"A Critical Overview of the Web Services Choreography Description Language"*. Business Process Trends White Paper, 2005. http://www.bptrends.com/publicationfiles/03-05%20WP%20WSCDL%20Barros%20et%20al.pdf

[Abo 10] A. Bosin, N. Dessì, M. Bairappan. *"A Service-based Approach for the Execution of Scientific Workflows in Grids"*. In CF '10 Proceedings of the 7th ACM international conference on Computing frontiers, 2010.

[Axi 10] Axis2. *"Welcome to Apache Axis2/Java"*. 2010. http://axis.apache.org/axis2/java/core

[Bbe 05] B. Benatallah, R. Dijkman, M. Dumas Z. Maamar. *"Service Composition: Concepts, Techniques, Tools and Trends"*. In Z. Stojanovic, A. Dahanayake, Eds. Service-Oriented Software Engineering: Challenges and Practices. Idea Group Inc (IGI), pp.48-66, 2005.

[Bei 09] BeInGrid. *"Business experiments in grid"*. 2009. http://www.beingrid.com

[Bio 09] BioGrid. *"Construction of a supercomputer network"*. Japanese, 2009. http://www.biogrid.jp

[Bli 03] B. Limthanmaphon, Y. Zhang. *"Web Service Composition with Case-Based Reasoning"*. In ADC'03 Proceedings of the 14th Australasian database conference, Volume 17, 2003.

[Bpi 05] BPMI, *"Business Process Management Initiative"*. 2005. http://www.bpmi.org

[Bsr 03] B. Srivastava, J. Koehler. *"Web Service Composition – Current Solutions and Open Problems"*. In Proceeding of the 13th International Conference on Automated Planning and Scheduling (ICAPS 2003), Workshop on Planning for Web Services Trento, Italy, June 2003.

[Bul 97] Bull. *"Bull, Architect for an open world"*. 1997. http://www.bull.com/fr/

[Cer 08] CERN. *"The Grid: separating fact from fiction"*. Adapted from an article originally published in Symmetry Breaking, 2008. http://public.web.cern.ch/public/en/spotlight/SpotlightGridFactsAndFiction-en.html

[Cfe 04] C. Fellenstein, J. Joseph. *"Grid Computing"*. Prentice Hall/IBM Press, ISBN-10: 0-13-145660-1, 2004.

[Chi 09] ChinaGrid. *"China Education and Scientific Research Grid Project"*, 2009. http://www.chinagrid.edu.cn/chinagrid/index.jsp

[Clo 04] C. Lopez. *"Service web et adaptabilité à l'utilisateur"*. IMAG : Institut d'informatique et de Mathématique Appliquée de Grenoble, 2004.

[Clv 08] C. Lopez-velasco. *"Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation"*. Thèse de doctorat, Université JOSEPH FOURIER, Laboratoire d'Informatique, Grenoble, 2008.

[Cng 09] CNGrid. *"China National Grid"*. 2009. http://www.cngrid.org/en_index.htm

BIBLIOGRAPHIE

[Con 10] Condor. *"High thorughput computing"*. 2010. http://www.cs.wisc.edu/condor/

[Cor 10] CORBA. *"CORBA Spécifications"*. 2010. http://www.corba.org/

[Cpe 03] C. Peltz. *"Web Services Orchestration and Choreography"*. IEEE Computer, vol.36, n°10, pp.46-522003, 2003.

[Cph 03] C. Peltz, H. Packard. *"Web services orchestration, a review of emerging technologies, tools, and standards"*. Technical report, Hewlett-Packard Company, 2003. http://xml.coverpages.org/HP-WSOrchestration.pdf

[Cro 09] CrossGRID. *"Key project achievements"*. 2009. http://www.crossgrid.org/main.html

[Dat 09] DataGrid. *"The DataGrid project"*. 2009. http://eu-datagrid.web.cern.ch/eu-datagrid

[Dbo 04] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard. *"Web Services Architecture"*. W3C Working Group Note, 11 February, 2004. http://www.w3.org/TR/ws-arch/wsa.pdf

[Dcl 06] D. Claro, P. Albers, J. Hao. *"Web Services Composition"*. In: Cardoso, J., Sheth, A., Eds. Semantic Web Services, Processes and Applications. Springer, pp.195-225, 2006.

[Dgr 09] D-Grid. *"D-Grid initiative"*. 2009. https://www.d-grid.de

[Dkb 03] D. K. Barry. *"Web Services and Service-Oriented Architectures"*. The Savvy Manager's Guide. Morgan Kaufmann, 2003.

[Dle 02] D. Levy. *"Coordination de Web Services : Langages de description et plate-formes d'exécution"*. Rapport de recherche XRCE Grenoble, septembre 2002.

[Dma 04] D. Martin, M. Burstein, J. Hobbs, O. Paolucci Lassila, D. McDermott, S. McIlraith, D. McGuinness, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, K. Sycara. *"OWL-S: Semantic Markup for Web Services"*. W3C Member Submission, 2004. http://www.w3.org/Submission/OWL-S

[Dsa 08] D. Salter, F. Jennings. *"Building SOA-Based Composite Applications Using NetBeans IDE 6"*. Published by Packt Publishing Ltd. ISBN 978-1-847192-62-2, 2008.

[Dth 97] D. Thompson, C. Exton, L. Garrett, A.S.M. Sajeev. D. Watkins. *"Distributed Component Object Model (DCOM)"*. 1997.

[Dut 09] DutchGrid. *"Large-scale distributed computing in the Netherlands"*. 2009. http://www.dutchgrid.nl/

[Ech 01] E. Christensen, F. Curbera, G. G. Meredith, S. Weerawarana. *"Web Services Description Language (WSDL) 1.1"*. W3C Note , 2001. http://www.w3.org/TR/wsdl

[Ege 09] EGEE. *"The EGEE project: Enabling Grids for E-sciencE"*, 2009. http://www.eu-egee.org

[Ejb 10] EJB. *"Entreprise Java Beans"*. 2010. http://java.sun.com/products/ejb/docs.html

[Ela 06] E. Laure, C. Gr, S. Fisher, A. Frohner, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buncic, R. Byrom, L. Cornwall, M. Craig, A. Di Meglio, A. Djaoui, F. Giacomini, J. Hahkala, F. Hemmer, S.

Hicks, A. Edlund, A. Maraschini, R. Middleton, M. Sgaravatto, M. Steenbakkers, J. Walk, A. Wilson. *"Programming the Grid with gLite"*. In Computational Methods in Science and Technology, Vol. 12, Key: citeulike: 4292588, 2006.

[Eum 10] EumedGrid. *"EumedGrid Initiative"*. 2010. http://www.eumedgrid.eu/

[Fcu 02] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, et S. Weerawarana *"Unraveling the Services Web An Introduction to SOAP,WSDL, and UDDI"*. IEEE Internet Computing 2, PP. 86-93, 2002.

[Fcw 01] F. Curbera, W. A. Nagy, S. Weerawarana. *"Web services: Why and how?"*. In OOPSLA 2001 Workshop on Object-Oriented Web Services, 2001.

[Fid 11] Fidens. "Sécurité des Web Services". 2011. http://www.fidens.fr/articles/securite-des-web-services-70.html

[Fle 01] F. Leymann. *"Web Service Flow Language 1.0"*. IBM Report, 2001. http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[Fwy 03] F. Wynen. *"Conception et développement d'une plate-forme de coopération inter-organisationnelle : le point de synchronisation"*. Mémoire d'ingénieur CNAM –Conservatoire Nationale des Arts et Métiers, Nancy, 102p, 2003.

[Gal 04] G. Alonso, F. Casati, H. Kuno, V. Machiraju. *"Web Services: Concepts, Architectures and Applications"*. 354p. Springer, 2004.

[Gar 09] GarudaIndia. *"GARUDA: the national grid computing initiative"*. 2009. http://www.garudaindia.in

[Gfe 03] G. Feltin, G. Doyen, O. Festor. "*Les protocoles Peer-to-Peer, leur utilisation et leur détection"*. 2003. http://home.etu.unige.ch/~roko1/gra/+/skoli/oint/kazaa%20jumps/paper.70.pdf

[Gli 09] gLite. *"An overview of grid middleware and gLite"*. EGEE-II INFSO-RI-031688, 2009. http://www.eu-egee.org

[Glo 09] Globus. *"Globus Alliance"*. 2010. http://www.globus.org

[Glt 10] gLite3.0. *"GLITE 3.2 USER GUIDE"*. CERN, 2010. https://edms.cern.ch/file/722398/1.4/gLite-3-UserGuide.pdf

[Gnu 10] GNUtella. 2010. http://sourceforge.net/projects/rfc-gnutella/

[Gri 10] Grid Café, *"What is grid computing? "*. http://www.gridcafe.org/version1/whatisgrid/whatis.html

[Grl 09] Grl. *"Grille de calcul: l'internet du calcul intensif"*. IN2P3, CNRS, 2009. http://www.in2p3.fr/presentation/thematiques/grille/grille.htm

[Grp 09] GridPP. *"Grid for Particle Physics"*. UK computing for particle physics, 2009. http://www.gridpp.ac.uk

[Gr5 09] Grid'5000. *"Grid'5000 at a glance"*, 2009. https://www.grid5000.fr

[Hgj 08] H. Gjermundrod, M. D. Dikaiakos, M. Stumpert, P. Wolniewicz, H. Kornmayer. *"g-Eclipse - an integrated framework to access and maintain Grid resources"*. In GRID '08 Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing, 2008.

[Hsa 05] H. Salah, B. Mahmoud Mentouri, B. Nacer. *"An Architecture for the Interoperability of Workflow Models"*. In IHIS '05 Proceedings of the first international workshop on Interoperability of heterogeneous information systems, ACM, ISBN:1-59593-184-8, 2005.

BIBLIOGRAPHIE

[Ibm 02] IBM, *"IBM Solutions Grid for Business Partners: Helping IBM Business Partners to Grid-enable applications for the next phase of e-business on demand"*, International Business Machines Corporation, 2002.

[Ifo 02] I. Foster. *"What is the Grid? A Three Point Checklist"*. GRIDToday, 2002. http://www.mcs.anl.gov/~itf/Articles/WhatIsTheGrid.pdf

[Ifs 01] I. Foster, C. Kesselman, S. Tuecke. *"The anatomy of the grid: Enabling scalable virtual organizations"*. International J. Supercomputer Applications, 2001.

[Ija 08] I. Janciak, C. Kloner, P. Brezany. *"Workflow Enactment Engine for WSRF-Compliant Services Orchestration"*. In GRID'08 Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing, 2008. http://joung.im.ntu.edu.tw/teaching/distributed_systems/documents/IBM_grid_wp.pdf

[Inf 09] INFN. *"The Infn grid project"*. Italy, 2009. http://grid.infn.it

[Jic 01] Ji. Clark, C. Casanave, K. Kanaskie, B. Harvey, Ja. Clark, N. Smith, J. Yunker, K. Riemer. *"ebXML Business Process Specification Schema 6 Version 1.01"*, 2001. http://www.ebxml.org/specs/ebBPSS.pdf

[Jra 04] J. Rao, X. Su, *"A Survey of Automated Web Service Composition Methods"*. Norwegian University of Science and Technology Department of Computer and Information Science N-7491, Trondheim, Norway, 2004.

[Jya 04] J. Yang, M.P. Papazoglou. *"Service Component for Managing Service Composition Life-Cycle. Information Systems"*. vol.29, n°2, pp.97-125, 2004.

[Jyu 05] J. Yu, R. Buyya, *"A taxonomy of scientific workflow systems for grid computing"*, In Newsletter ACM SIGMOD Record, Volume 34 Issue 3, September 2005.

[Kaz 10] KazaA. 2010. http://www.kazaa.com/

[Kfr 02] K. Frascaria. *"Le Grid démultiplie la puissance de calcul"*. Decision Micro (n° 506), 29/04/2002. http://www.01net.com/article/182614.html

[Khe 01] K. Heather. *"Web services conceptual architecture (WSCA 1.0)"*, may 2001. http://www-06.ibm.com/software/solutions/webservices/pdf/WSCA.pdf

[Kll 06] K. L. L. Tan, K. J. Turner. *"Orchestrating Grid Services using BPEL and Globus Toolkit 4"*.In Proceeding of the 7th PGNet Symposium, pages 31–36, 2006.

[Kta 09] K. Tan, F. Magoulès, J. Pan, A. Kumar. *"Introduction to Grid Computing"*. CRC Press 2009, ISBN: 978-1-4200-7406-2, 2009.

[Lba 06] L. Baduel, F. Baude, D. Caromel, A. Contes, F. Huet, M. Morel, R. Quilici. *"Programming, Composing, Deploying for the Grid"*. Grid Computing: Software Environments and Tools (O. F. Cunha, C. Jose, ed. Rana), pp. 205 – 229, Springer, 2006. http://hal.archives-ouvertes.fr/inria-00486114/en

[Lcg 09] LCG. *"LHC Computing Grid Project"*. Worldwide LHC computing grid: distributed production environment for physics data processing, 2009. http://lcg.web.cern.ch/LCG/

[Leg 10] Legion. *"A worldwild virtual computer"*. 2010. http://legion.virginia.edu/

[Mba 02] M. Baker, R. Buyya, D. Laforenza. *"Grids and Grid technologies for wide-area distributed computing"*. International Journal of Software: Practice and Experience (SPE), vol. 32, no 15, Wiley Press, USA, Nov, 2002.

BIBLIOGRAPHIE

[Mbj 06] M. B. Juric, B. Mathew, P. Sarang. *"Business Process Execution Language for Web Services Second Edition"*. Published by Packt Publishing Ltd. 32 Lincoln Road Olton Birmingham, B27 6PA, UK, ISBN 1-904811-81-7, January 2006.

[Mbl 04] M. Blow, Y. Goland, M. Kloppmann, F. Leymann, G. Pfau, D. Roller, M. Rowley. *"BPELJ: BPEL for Java"*. A joint white paper by BEA and IBM, 2004. http://public.dhe.ibm.com/software/dw/webservices/ws-bpelj/ws-bpelj.pdf

[Mke 04] M. Keidl, A. Kemper. *"Towards Context-Aware Adaptable Web Services"*. In Proceeding of the 13th international World Wide Web Conference (WWW 2004), NewYork, USA, pp.55-65, May 2004.

[Mqu 09] M. Quinson, F. Suter. *"Grilles informatiques et algorithmique distribuée avancée"*. 2008-2009. www.loria.fr/~suter/files/SDR08-09.pdf

[Mvi 04] M. Vialette, *"Web Service : communication inter- langage"*. Sun Microsystem laboratoire SUPINFO des technologies, 2004.

[Nar 09] NAREGI. *"The national research grid initiative"*. 2009. http://www.naregi.org/index_e.html

[Nka 05] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, C. Barreto. *"Web Services Choreography Description Language Version 1.0"*. W3C Candidate Recommendation, 2005. http://www.w3.org/TR/ws-cdl-10 .

[Nmi 03] N. Mitra, Y. Lafon. *"Simple Object Access Protocol (SOAP)"*, W3C (2003), 2003. http://www.w3.org/TR/soap

[Nov 04] NovaForge. *"BPM – Orchestra"*. 2004. http://www.novaforge.org/novaforge/fr-selectionner/bmp/orchestra

[Nte 07] N. Temglit. *"Un modèle de composition des services web sémantiques"*. Thèse de magister, Université des Sciences et de la Technologie HOUARI BOUMEDIENNE, Faculté d'Electronique et d'Informatique, 2007.

[Oez 07] O. Ezenwoye, S. Masoud Sadjadi, A. Cary, M. Robinson, *"Orchestrating WSRF-based Grid Services"*, In OTM'07 Proceedings of the 2007 OTM confederated international conference on the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part II, 2007.

[Ogs 08] OGSA. *"Open Grid Services Architecture"*. 2008. http://www.globus.org/ogsa

[Ogsd 09] OGSA-DAI. *"What is OGSA-DAI ? "*, Web Published, 2009. http://www.ogsadai.org.uk/about/ogsa-dai

[Orc 10] Orchestra, *"Overview"*. 2010. http://orchestra.ow2.org/xwiki/bin/view/Main/WebHome

[Osg 09] OSG. *"Science on the Open Science Grid"*, 2009. http://www.opensciencegrid.org

[Pdu 04] P. Duvanel, *"Les services Web : La problématique de la spécification"*. rapport personnel, Août 2004.

[Pjg 07] P. J. Gobe Hobona, D. Fairbairn. *"Workflow Enactment of Grid-Enabled Geospatial Web Services"*. In Proceedings of the 2007 UK e-Science All Hands Meeting, 2007.

BIBLIOGRAPHIE

[Pke 04] P. Kellert et F. Toumani. *"Les Web services sémantiques"*. Laboratoire LIMOS - UMR (6158), CNRS ISIMA, 2004. http://www.revue-i3.org/hors_serie/annee2004/revue_i3_hs2004_01_07.pdf

[Ppl 05] P. Plaszczak, R. Wellner. *"Grid computing"*. Elsevier/Morgan Kaufmann, San Francisco, 2005.

[Rbu 10] R. Buyya. *"Grid Computing Info Centre (GRID Infoware)"*. http://www.gridcomputing.com/

[Rmi 10] RMI. *"Remote Method Invocation Home"*. 2010. http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html

[Rsv 05] R. Buyya, S. Venugopal. *"A Gentle Introduction to Grid Computing and Technologies"*. CSI Communication, 2005. http://www.buyya.com/papers/GridIntro-CSI2005.pdf

[Sdg 09] SDG, *"Scientific Data Grid"*. Chineese, 2009. http://www.sdg.ac.cn

[Smo 06] S. Monnet. *"Gestion des données dans les grilles de calcul : support pour la tolérance aux fautes et la cohérence des données"*. Thèse de doctorat, université de RENNES1, École doctorale MATISSE, 2006.

[Sss 10] S3. *"Simlpe Storage Service"*. 2010. http://aws.amazon.com/fr/s3/

[Sth 01] S. Thatte. *"XLANG: Web Services for Business Process Design"*. Microsoft Specification, 2001. http://xml.coverpages.org/XLANG-C-200106.html

[Stu 03] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, P. Vanderbilt. *"Open Grid Services Infrastructure (OGSI) v1.0"*. Technical report, Global Grid Forum, 2003. http://www.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf

[Tan 03] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weerawarana. *"Business Process Execution Language for Web Services, Version 1.1"*. IBM Specification, 2003. http://www-128.ibm.com/developerworks/library/specification/ws-bpel

[Tdo 07] T. D¨ornemann, T. Friese, S. Herdt, E. Juhnke, B. Freisleben. *"Grid Workflow Modelling Using Grid-Specific BPEL Extensions"*. In Proceedings of German e-Science Conference 2007, pages 1–9, 2007.

[Ter 09] TeraGrid. *"The TeraGrid project"*. 2009. http://www.teragrid.org

[Tfl 08] T. Fleuren, P. Müller. *"BPEL Workflows Combining Standard OGCWeb Services and Grid-enabled OGC Web Services"*. In Proceeding of the 34th Euromicro Conference on Software Engineering and Advanced Applications, Sep 1-5, Parma, Italy, 2008.

[Tme 04] T. MELLITI, *"Interopérabilité des services Web complexes, Application aux systèmes multi-agents"*. Thèse de Doctorat de l'Université Paris IX Dauphine, décembre 2004.

[Tpi 05] T. Priol. *"Défis et perspectives scientifiques des grilles informatiques"*. IRISA, INRIA, 2005. http://rangiroa.essi.fr/cours/systeme2/02-flips-grid.pdf

[Tpr 04] T. Priol. *"Grid computing: approches et tendances"*. IRISA, INRIA, 2004. http://www-sop.inria.fr/intech/grid/priol.pdf

[Udd 03] UDDI, *"Universal description, discovery, and integration (UDDI)"*, W3C (2003), 2003. http://www.uddi.org

BIBLIOGRAPHIE

[Uni 10] Unicore. "Distributed computing and data resources". 2010. http://www.unicore.eu/

[Wem 05] W. Emmerich, B. Butchart, L. Chen, B. Wassermann, and S. L. Price. *"Grid Service Orchestration Using the Business Process Execution Language (BPEL)"*. J. Grid Computing, 3(3-4):283–304, 2005.

[Wsr 07] WSRF. *"Web Services Resource Framework (WSRF)"*. Technical report, OASIS, 2007. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

[Yde 07] Y. Deng, F. Wang. *"A Heterogeneous Storage Grid Enabled by Grid Service"*. ACM SIGOPS Operating Systems Review, Volume 41 Issue 1, 2007.

ANNEXE A : Aperçu rapide des projets Grid

A.1. Les projets américains

- **Globus [Glo 09]** travaille principalement sur les technologies de l'infrastructure de grille. Le noyau de Globus Grid est la boîte à outils Globus Toolkit (GT). La version actuelle GT4 a été publiée. GT comprend un ensemble d'outils de la grille en couches réalisant des services de base pour la sécurité, la localisation des ressources, la gestion des ressources, la communication, etc. Ces composants ont été déployés au dessus de Globus Ubiquitous Supercomputing Testbed (GUSTO) à travers 17 sites. Ils supportent efficacement l'infrastructure de grille d'application. La combinaison de Globus Toolkit et du service Web met l'avenir d'un produit de recherche standardisé sur les grilles.
- **Open Science Grid (OSG) [Osg 09]** est une infrastructure de grille américaine pour la recherche scientifique. Elle a organisée une masse de ressources de calcul et de stockage, et les a rendus dans une cyber-infrastructure uniforme et partagée. Ses 50 sites sont dispersés sur les Etats-Unis, en Asie et en Amérique du Sud. Elle dispose de deux grilles: une grille d'intégration et une grille de production. La grille d'intégration fait face à la recherche scientifique pour ses applications et services de test. La grille de production fait face à l'industrie et offre aux utilisateurs un traitement (développement) stable et les ressources de stockage de données. Une des motivations de l'OSG est de développer de nouveaux services et puis les mettre dans l'environnement de production. La version actuelle de l'OSG comprend les Computing Element (CE), les Storage Element (SE), la Visual Organization (VO), le Membership Service et le Service Catalogue.
- **TeraGrid [Ter 09]** est un ensemble de ressources de calcul haute gamme communes aux États-Unis. Ces ressources comprennent des ordinateurs de haute performance et les ressources de données réparties sur 7 sites. L'outil Common TeraGrid Software Stack (CTSS) a été développé pour l'utilisation de ces ressources. Le CTSS est installé dans tous les ordinateurs, ce qui garantit l'homogénéité des services et des outils sur différentes ressources. Account Management Information Exchange (AMIE) réalise une gestion automatique des comptes. En ce qui concerne la sécurité, gx-carte peut gérer les CA (Certificate Authority) des utilisateurs.

A.2. Les projets européens

- **BEinGRID [Bei 09]** (Business Experiments in GRID) est un projet européen de grid. Son objectif est de conduire l'utilisation universitaires et de recherche des grilles dans les secteurs d'activité. Dix-huit expériences commerciales vont être lancées dans le projet BEinGRID. En outre, BEinGRID prévoit d'élaborer un ensemble d'outils de référentiel de composants de grid services pour soutenir (supporter) ainsi les entreprises européennes. Ce logiciel va pleinement profiter des composants de grille existant afin d'éviter le réaménagement.
- **EGEE [Ege 09]** (Enabling Grids for E-sciencE) est un projet visant à fournir des ressources informatiques pour la recherche universitaire et la production industrielle. La grille EGEE est une grille à travers le monde. Les utilisateurs de ce système de grille ne sont pas limités par leur situation géographique. EGEE offre non seulement une ressource stable et robuste de grille (plus de 30000 CPU, plus de 5 péta-octets d'espace de stockage), mais également des services de formation pour ses utilisateurs. Les applications de ce système de grille peuvent être diverses. À l'heure actuelle, ses applications sont principalement dans deux domaines: la physique des hautes énergies

(HEP) et biomédicale. Plus d'applications commerciales et généralisés seront lancées sur la grille EGEE à l'avenir.

- **Grid5000 (France) [Gr5 09]** est un projet national de grid de la France. Il s'agit d'une plate-forme de grille pour la recherche universitaire; 5000 processeurs répartis sur 9 sites en France. Les utilisateurs peuvent réserver le PC quand ils veulent réaliser leurs expériences. Ils peuvent également configurer les machines par elles-mêmes. Cette plate-forme de grille fournit le mécanisme de réservation et de configuration pour les utilisateurs. En outre, Grid5000 a offert un wiki comme site web pour la communication des utilisateurs. Les utilisateurs peuvent soumettre leurs rapports d'expériences sur ce site web.
- **D-Grid initiative [Dgr 09]** est une plateforme de grid allemande fondée en 2005 pour l'enseignement et la recherche. Malgré la contribution d'une ressource de haute performance de la grille, D-Grid est consacrée au traitement et à l'accès à de grandes quantités de données scientifiques. Sur cette plate-forme, une masse de données scientifiques, provenant de divers domaines, tels que la physique des hautes énergies, l'astrophysique, la médecine, etc., sont recueillies et partagées.
- **DutchGrid [Dut 09]** est une plate-forme Open Grid pour la recherche dans les Pays-Bas. Elle fournit une ressource de calcul pour divers types de déploiements d'expérience de recherche. En ce qui concerne la sécurité, le service DutchGrid Certificate Authority, développé par NIKHEF à Amsterdam, permet à l'utilisateur d'accéder ou de partager les ressources de calcul aux Pays-Bas ou en Europe.
- **GridPP [Grp 09]** (Grid for Particle Physics : Grille de physique des particules) est un projet britannique pour une grille de physique des particules. La motivation de ce projet de grille est d'offrir des outils et des infrastructures pour que les utilisateurs puissent utiliser les ressources de manière transparente sans avoir à les rechercher. Les utilisateurs sont les physiciens travaillant pour le LHC (lancé en 2007), qui ont besoin d'une coopération efficace et de traiter les données massives générées par le LHC. En fait, GridPP est une partie du projet EGEE, et il constitue la contribution du Royaume-Uni pour le LCG.
- **INFN [Inf 09]** (l'Institut national italien de physique nucléaire) est un projet de recherche qui vise à l'installation et l'utilisation généralisée d'une plateforme de grid à grande échelle. En outre, l'INFN a beaucoup de collaboration en Europe et partout dans le monde, y compris LCG du CERN. L'INFN développe plusieurs applications middleware pour la planification de tâches distribuées et la surveillance (monitoring), la gestion de ressources de la grille (ressources de calcul et ressources de stockage), la collecte des informations utilisateur, DataGrid et outils web-based.
- **CrossGrid [Cro 09]** Le projet est un système de grille avec la fonction de réponse en temps réel. Il permet aux utilisateurs de surveiller et contrôler l'application pendant l'exécution, par exemple, en changeant ses configurations. La plupart des applications CrossGrid ont besoin d'une interaction en temps réel, telles que les simulations distribuées en temps réel de l'environnement, qui implique l'interaction de médecins. Les applications principales de CrossGrid sont en traitement médical, les inondations, la physique des particules et météo / pollution.
- **Le CERN [Cer 08]** est célèbre pour son invention énorme du World Wide Web. Le Large Hadron Collider (LHC), le plus grand instrument scientifique dans le monde, est désormais opérationnel au CERN. L'énorme quantité de données produites par le LHC est un énorme défi pour les informaticiens. Cette tâche ne peut être accomplie par un seul ordinateur. En raison de la nécessité de traiter, de stocker et d'analyser statistiquement la quantité massive de données, le LHC Computing Project (LCG) **[Lcg 07]** est lancé en utilisant l'architecture de grille de calcul à cause de sa facilité de l'entretien des systèmes

distribués et inférieur possibilité d'un échec global (les données sont transférées et enregistrées dans plusieurs sites). Mais cette architecture apporte aussi certains défis, tels que l'assurance de la communication entre les sites, la gestion des matériels et logiciels hétérogènes, les données de sécurité et sa gestion de l'information de partage.

- **DataGrid [Dat 09]** est un projet financé par l'Union européenne. Il vise à établir une infrastructure de calcul de la prochaine génération, qui fournit des calculs intensifs et analyse des bases de données partagées à grande échelle, à partir de centaines de téraoctets aux pétaoctets, largement diffusé à travers les communautés scientifiques. Le DataGrid est axé sur les applications de la physique des hautes énergies du CERN. Il porte sur le stockage et la manipulation décomposée en questions de données massives. Ensuite, les résultats de la recherche sera étendue aux autres domaines d'application, tels que la biologie, l'observation de la terre et ainsi de suite. Le DataGrid s'appuie sur les technologies Grid émergentes qui devraient permettre le déploiement d'un environnement de calcul à grande échelle comprenant des collections réparties de fichiers, bases de données, ordinateurs, instruments scientifiques, et périphériques. La plate-forme GT est le logiciel supporté sous le DataGrid.
- **EumedGrid [Eum 10]** Coodonné par l'INFN, l'initiative d'EUMEDGRID vise à développer dans le bassin Méditerranéen une infrastructure de grid pour la Recherche, qui peut devenir une partie d'EGEE et être intégrée avec des initiatives analogues dans les Balkans, l'Europe du nord, l'Amérique Latine et le Loin-Est Asiatique. Un autre but d'EUMEDGRID est d'accroître la conscience et les compétences relatives aux grids parmi les chercheurs qui travaillent dans la Région Méditerranéenne, pour les rendre capables de bénéficier de ce nouvel et puissant instrument, pour stimuler la collaboration avec des projets Européens et dans le monde entier et pour soutenir le développement scientifique et industriel dans la Région.

A.3. Les projets d'Asie

- **CNGrid [Cng 09]** (China National Grid) est un important projet soutenu par la Chine. Il s'agit d'un banc d'essai qui intègre le calcul de haute performance et la capacité de traitement des transactions d'une infrastructure d'information. Il prend en charge efficacement la recherche scientifique. CNGrid a développé la grille orientée superordinateurs, et l'a installé dans huit sites à travers le pays. Dix sous-projets de CNGrid couvrent différents domaines de recherche, dans lesquels la Science Data Grid (SDG) est incluse.
- **SDG [Sdg 09]** (Scientific Data Grid) est basé sur les ressources de données scientifiques de masse. Ce projet vise à connecter les sources de données de masse de bases de données scientifiques, et de partager ces ressources données géographiquement distribuées, hétérogènes et autonomes, par le biais de la technologie de grille. Certains middlewares de grille pour l'accès aux données, le service d'information et les questions de sécurité ont été utilisés. Ces données impliquent les domaines de l'astronomie, la physique des hautes énergies et la science médicale.
- **ChinaGrid [Chi 09]**, aussi appelé China Education et la Scientific Research Grid Project, vise à construire une plateforme de service public pour la recherche et l'enseignement supérieur en Chine. Il est parrainé par 12 universités de pointe, et établi sur la Chine et de l'Education Research Network (CERNET). ChinaGrid Platform Support (CGSP) est le middleware de grille développé pour ChinaGrid. CGSP a mis en œuvre certains éléments complémentaires qui ne sont pas réalisés par Globus Toolkit.
- **NAREGI [Nar 09]** (National Research Grid Initiative) est un projet coopératif entre l'industrie japonaise, l'éducation et le gouvernement, qui vise à développer les

middlewares de grille et les technologies réseau, notamment la gestion des ressources, les modèles de programmation de grille, les outils de déploiement de grille, l'intégration de la grille logiciel, infrastructure de communication de réseau, etc. Dans le domaine de l'industrie, une application de la nano-technologie de la science est une partie du projet, avec l'objectif de prouver qu'un environnement de grille de calcul de haut de gamme peut être utilisé pour les nano-sciences.

- **BioGRID [Bio 09]** vise à construire une grille de données (datagrid), qui ne rassemble et ne traite pas seulement les bases de données massives et les ensembles de données, mais combine également diverses ressources de calcul dans le traitement des données. Il est d'abord conçu pour la recherche biologique au Japon. Ses trois principaux objectifs sont le déploiement d'un analyseur sur le réseau supercalculateur, la jonction parfaite entre les bases de données et le traitement des données et la technologie de grille de données pour établir des liens et des opérations entre les systèmes de bases de données hétérogènes.
- **GARUDA [Gar 09]** est un projet coopératif entre les chercheurs scientifiques et les expérimentateurs en Inde. Ses objectifs sont de créer une grille de calcul de test et d'intégrer les potentiels de recherche et tracer un plan à plus long terme de grille de calcul. Les activités du projet comprennent la construction de réseaux, middleware, outils de gestion des ressources de calcul et de données, et un portail web.

ANNEXE B: Présentation de quelques serveurs BPEL

Les serveurs BPEL fournissent un environnement pour l'exécution de processus métier BPEL. BPEL est fortement lié aux services web et aux plates-formes modernes qui supportent le développement de services Web, en particulier Java Enterprise Edition et Microsoft .NET. Les serveurs BPEL sont en étroite relation avec les environnements des serveurs d'applications Java Enterprise Edition ou .NET, où ils peuvent utiliser les services fournis par les serveurs d'applications, tels que la sécurité, les transactions, l'évolutivité (scalabilité), l'intégration avec les bases de données, les composants comme les EJBs (Enterprise Java Beans) et COM+ (Component Object Model), les systèmes de messagerie tels que JMS (Java Message Service) ou MSMQ (Microsoft Message Queue), etc.

Des serveurs BPEL existent pour Java Enterprise Edition, .NET, et d'autres plateformes. Les serveurs commerciaux les plus importants sont énumérés ci-dessous:

- Oracle BPEL Process Manager (http://www.oracle.com/technology/products/ias/bpel/index.html) Microsoft BizTalk (http://www.microsoft.com/biztalk/)
- IBM WebSphere Business Integration Server Foundation (http://www.ibm.com/software/integration/wbisf)
- IBM alphaWorks BPWS4J (http://www.alphaworks.ibm.com/tech/bpws4j)
- BEA WebLogic Integration and the related AquaLogic (http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/weblogic/integrate/)
- OpenStorm Service Orchestrator (http://www.openstorm.com/)
- Active Endpoints ActiveWebflow (http://www.active-endpoints.com/products/index.html)
- Sun Java Integration Suite for Java Enterprise Suite (http://www.sun.com/software/javaenterprisesystem/), formellement connu sous le nom de SeeBeyond eInsight Business Process Manager (http://www.seebeyond.com/software/einsight.asp)
- Cape Clear Orchestration Studio (http://www.capeclear.com/products/)
- OpenLink Virtuoso Universal Server (http://virtuoso.openlinksw.com/)
- Parasoft BPEL Maestro (http://www.parasoft.com/jsp/products/home.jsp?product=BPEL)
- Fiorano Business Integration Suite (http://www.fiorano.com/products/fesb/fioranobis.htm)
- PolarLake Integration Suite (http://www.polarlake.com/en/html/products/integration/index.shtml)
- Fuego BPM (http://www.fuegotech.com/fuego_software.html)
- Digité Enterprise Business Process Management (http://www.digite.com/4.0/products/digite_ent_business-process.htm)

Il ya aussi quelques implémentations open-source:

- ActiveBPEL Engine (http://www.activebpel.org/) maintenant devenus commercial
- FiveSight Process eXecution Engine PXE (http://www.fivesight.com/pxe.shtml)

- bexee BPEL Execution Engine (http://sourceforge.net/projects/bexee)
- Apache Agila (http://wiki.apache.org/agila/), formellement connu sous le nom de Twister (http://www.smartcomps.org/twister/)
- Orchestra (http://orchestra.ow2.org/)

Plusieurs outils de conception et de développement BPEL (BPEL designer) sont également disponibles. Ces outils permettent le développement graphique de processus BPEL. Certains outils de design sont fournis avec les serveurs. Ci-dessous une liste des outils importants de design et développement :

- Oracle JDeveloper 10g (http://www.oracle.com/technology/products/jdev/index.html)
- Oracle BPEL Designer for Eclipse (http://www.oracle.com/technology/products/ias/bpel/index.html)
- IBM WebSphere Studio Application Developer Integration Edition (http://www.ibm.com/software/integration/wsadie/)
- iGrafx BPEL (http://www.igrafx.com/products/bpel/index.html)
- itp Process Modeler for Microsoft Visio (http://www.itp-commerce.com/)
- Netbeans BPEL Designer (http://dlc.sun.com.edgesuite.net/netbeans/6.0/final/)
- Eclipse BPEL Designer (http://www.eclipse.org/bpel/)