République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche
Université Saad Dahleb de Blida
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme de Master

Option : *Génie des Systèmes Informatiques*
-GSI-

Intitulé du thème

Elaboration des services web pour la génération automatique des emplois du temps.

Présenté par : Atmani Hakim

Guellil Merouene

Promotrice : M^me^ Mancer Yasmine

Encadreur : Mr Benaissa Rachid

Remerciements

C'est un grand plaisir pour nous autant qu'un devoir de remercier toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Ce travail n'aurait jamais pu voir le jour sans la contribution et l'effort de Mme Yasemine Mancer à qui nous exprimons notre profonde gratitude pour avoir bien voulu accepter de diriger ce travail et au membre du Jury pour avoir bien voulu accepter de juger notre travail.

Nous remercions également, notre encadreur Mr Benaissa Rachid et Mourad Guerbi de l'IAES de Blida et notre enseignant Monsieur Chérif Zaher.

Nos vifs remerciements vont aussi à nos familles, et plus particulièrement nos parents. On les remercie pour leur soutien de tous les instants.

Je dédie ce travail à
Mes très chers parents Chabanne et Fatima.
A ma chère sœur Samira et son Fiancé Farid Bekhir
Mes très chers frères ; Samir, Farid, Mourad, Salim et Belka.
A mon cousin ; Nassim Dinho
Et a toute ma famille.
A mon ami Mustapha et à sa Chère Mère qui n'est plus dans ce monde ;
Je dédie ce modeste travail...

Atmani Hakim

Je dédie ce travail à
Mes très chers parents Boualem et Bedra.
A ma chère sœur Rania
Mes deux très chers frères ; Aimad et Alaa
Et a toute ma famille.
Je dédie ce modeste travail...

Guellil Merouene

Résumé : La gestion des emplois du temps est un problème difficile, notamment dans les universités, et qui consomme de nombreuses ressources humaines et financières, Le problème des emplois du temps consiste à *répartir dans le temps des séances pendant lesquelles s'effectue une activité* pédagogique nécessitant des ressources, en l'occurrence des enseignants, groupes, salles et matériels, Cependant, les contraintes concernant les souhaits des responsables et des enseignants sont difficiles à exprimer.

Notre travail consiste à réaliser des services web pour la gestion des emplois du temps destinée à la pédagogique de l'Institut d'Aéronautique et des Etudes Spatiales.

Mots clés : Emplois du temps, service web, interaction, message.

Abstract: The management of Schedules is a difficult problem, particularly in universities, and consumes man, human and financial resources. The problem of Schedules consists in distributing in time the *sessions*, during which performs a pedagogical activity needing resources, at the occurrence teachers, groups, rooms and equipment However, the constraints the wishes of the responsible and teachers are hard to express.

Our job is to realize web services for the management of Schedules destined to pedagogy of Aeronautical and Space Sciences Institute of Blida.

Key words: Schedules, web service, interaction, message.

ملخص: إدارة الجداول الزمنية هي مشكلة صعبة، وخاصة في الجامعات، وتستهلك الكثير من الموارد البشرية والمالية. مشكلة الجداول الزمنية هي توزيع في الوقت دورات والتي يحدث خلالها نشاط تعليمي متطلبا موارد منها المعلمين، والغرف والمجموعات والمعدات ومع ذلك، فإن رغبات المسؤولين و المدرسين من الصعب التعبير عنها.

مهمتنا هي تقديم خدمات الويب لإدارة جداول زمنية موجهة لبيداغوجيا معهد الطيران و الدراسات الفضائية

الكلمات الأساسية: الجداول الزمنية، خدمات الويب، تفاعل، رسالة.

Sommaire

Listes des figures

Chapitre I : Les services Web

Chapitre II : La Composition des services web

Chapitre III : Optimisation combinatoire : Algorithmes Génétiques pour la génération des emplois du temps

Chapitre IV: Conception

Liste des tableaux

Chapitre II : La Composition des services web

Chapitre III : Optimisation combinatoire

Chapitre IV: Conception

Chapitre V: Implémentation

Liste des acronymes

IAESB : Institut d'Aéronautique et des Etudes Spatiales de Blida.

XML : est une recommandation du W3C. C'est un langage de balisage définissant un format universel de représentation des données, permettant de décrire la structure hiérarchique d'un document. Ainsi, un document XML contient à la fois les données et les indications sur le rôle que jouent ces données. Ces indications permettent de déterminer la structure du document. XML est défini pour fonctionner indépendamment des plateformes et des systèmes d'exploitation.

SOA : Architecture Orientée Service.

WSDL : Web service Description Language.

SOAP : Simple Object Access Protocol.

UDDI : Universal Description Discovery and Integration

B2B (Business-to-Business) : Qualificatif signifiant une interaction entre deux entités d'affaires. Par exemple, nous pouvons utiliser l'expression marketing B2B, qui signifie le nmarketing d'entreprise à entreprise.

B2C : Le Business to Consumer aussi appelé Business to Customer est le nom donné à l'ensemble d'architectures techniques et logiciels informatiques permettant de mettre en relation des entreprises directement avec les consommateurs : en français, « des entreprises aux particuliers ».

IBM : International Business Machines.

QoS : Qualité de service.

COM (Component Object Model) : est une technique de composants logiciel crée par Microsoft. COM est utilisé en programmation pour permettre le dialogue entre programmes.

CORBA (Common Object Request Broker Architecture) : Architecture et spécifications visant à permettre de créer, publier et gérer des objets, sur un réseau, dans un contexte de programmation distribuée. Cette architecture a déjà été largement utilisée sauf par Microsoft qui avait développé sa propre architecture DCOM.

DCOM (Distributed Component Object Model) : Ensemble d'interfaces programmes et de concepts développé par Microsoft qui est en fait l'équivalent de CORBA. Il est utilisé entre autre pour faire des appels RPC (Remote Procedure Call).

W3C (World Wide Web Consortium) : Consortium industriel visant à favoriser l'interopérabilité des produits informatique et l'évolution du World Wide Web, par le développement de standards et de spécifications techniques.

OASIS (Organization for the Advancement of Structured Information Standards): OSBL technologique international visant la promotion et l'adoption de standards indépendants des manufacturiers, dans le domaine des technologies de l'information. OASIS tente de rapprocher les acteurs industriels et les groupes de standardisation afin qu'ils s'entendent sur l'utilisation universelle de XML.

Protocole : Ensemble de règles qui sont partagée entre deux ordinateurs afin de communiquer l'un avec l'autre.

HTTP (Hypertext Transfer Protocol) : Protocole de communication permettant l'échange de fichiers sur le World Wide Web. HTTP est un protocole d'application développé par le W3C.

SMTP (Simple Mail Transfer Protocol (littéralement) : est un protocole de communication utilisé pour transférer le courrier électronique vers les serveurs de messagerie électronique.

HTML (Hypertext Markup Language) : Ensemble de balises et de symboles insérés dans un fichier informatique visant à être affiché sur un navigateur World Wide Web. Les balises HTML définissent comment un navigateur Web devra afficher les informations contenues dans une page Web. HTML a été développé par le W3C.

Introduction générale

Plusieurs innovations technologiques ont eu lieu ces dernières années pour répondre aux besoins de partage des ressources et de communication entre les entreprises. Ceci a poussé la recherche dans ce domaine à trouver de meilleures façons de développer des solutions informatiques.

Apparus dès la fin des années 1990, à l'aube du 21ème siècle, les services Web ont provoqué une forte évolution dans le monde de l'informatique distribuée, et un bouleversement majeur dans la façon de concevoir des architectures. Un des intérêts du service Web est de faciliter l'interconnexion entre les différentes applications distantes, indépendamment des plateformes et des langages de programmation utilisés. Les services Web semblent être la solution de l'avenir pour implémenter les systèmes distribués, aujourd'hui, ces services sont distribués à large échelle sur Internet. Le développement d'Internet, la structuration des données via XML, et la recherche d'interopérabilité sont autant de facteurs qui ont favorisé l'essor des services Web. Les services Web constituent la technologie de base pour le développement d'architectures orientées services. Ces derniers sont des modèles qui définissent un système par un ensemble de services logiciels distribués, qui fonctionnent indépendamment les uns des autres afin de réaliser une fonctionnalité globale. Les architectures orientées services, et en particulier les services Web, permettent l'accessibilité, la découverte et l'utilisation universelle de n'importe quelle application logicielle sur le Web en utilisant des normes ouvertes. Ils constituent un paradigme permettant d'organiser et d'utiliser des services Web distribués.

La confection d'emploi du temps est une tache fastidieuse et répétitive qui fait généralement intervenir de nombreux éléments d'information. La difficulté du problème qui en résulte est liée à la nature des contraintes qu'il s'agit de satisfaire. A celle-ci s'ajoute la crainte que quelques incompatibilités ne se manifestent après la publication de l'emploi du temps définitif. Les techniques manuelles utilisées actuellement par la plupart des responsables horaires sont portées à disparaitre. Ces derniers sont prêts à faire appel à des outils informatiques qui les soulageraient dans leur activité en leur fournissant un horaire complet.

Un certain nombre de technique et approche ont été proposés dans la littérature pour résoudre le problème d'emplois du temps dans le domaine de l'éducation en général et l'emploi du temps universitaire en particulier. La plus part des première technique étaient basées sur la simulation de l'être humain pour la construction d'un emploi du temps. L'emploi du temps était construit pas à pas, on ajoute leçon âpres leçon à l'emploi du temps partiel en essayant de satisfaire l'ensemble de toutes les contraintes à chaque fois. Par la suite, un ensemble d'autres méthodes ont été appliquées dans le domaine de la recherche pour résoudre le problème. Entre programmation linéaire en nombre entier, Réseaux de flux, Le problème a été même réduit à un simple problème de coloriage de graphe. Plus récemment, des approches fondées sur les techniques de recherche utilisées également en Intelligence Artificielle apparu dans la littérature, nous avons recuit simulé, recherche tabou, algorithmes génétique.

- **Problématique**

Chaque année, les responsables pédagogiques des universités ont pour mission d'organiser les emplois du temps des différentes formations en essayant, au mieux, de satisfaire les contraintes « humaines » des enseignants et des étudiants, les contraintes pédagogiques imposées par la progression des enseignements et en tenant compte des contraintes « physiques » liées aux ressources matérielles (les salles, les équipements, etc.).

Le travail des responsables pédagogiques dans ce domaine et des plus compliqué car ça reste difficile de gérer plusieurs types de contraintes et les satisfaire au même temps, de plus, les responsables pédagogiques avouent que les difficultés ne viennent pas seulement de la génération des emplois du temps mais aussi de leurs manipulation.

- **Objectif**

C'est dans ce contexte que s'inscrit notre travail de fin d'études qui consiste à résoudre les problèmes de la génération automatique des emplois du temps de l'Institut d'Aéronautique et des Etudes Spatiales de Blida, à travers l'utilisation des services web.

L'objectif de notre travail est de réaliser une méthode d'adaptation avec satisfaction des contraintes pour la génération automatique des emplois du temps et réaliser des services web pour la gestion des emplois du temps destinée à la pédagogie de l'institut d'aéronautique et des études spatiales de Blida :

- ✓ Le premier service web a pour but la génération automatique des emplois du temps et la gestion des mises à jour.
- ✓ Le deuxième permettra aux enseignants de l'institut de consulter et de suggérer des modifications des séances de leur emploi du temps.
- ✓ Le troisième service web est désigné aux étudiants pour consulter leurs emplois du temps.

Notre mémoire est structuré en deux parties :

- ✓ La première partie constitue un état de l'art composé de trois chapitres suivants :
  - Le premier chapitre détaille les concepts de base des services Web ainsi que les principaux standards : SOAP comme protocole de transport, WSDL comme langage de description des services web et UDDI comme annuaire pour assurer la publication et la découverte des services.
  - Le deuxième chapitre est consacré à la composition des services web.

Chapitre I

Les services Web

I.1. Introduction

Les dernières décennies ont été marquées par le développement rapide des systèmes d'information distribués, et tout particulièrement par la diffusion de l'accès à Internet. Cette évolution du monde informatique a entraîné le développement de nouveaux paradigmes d'interaction entre applications tels que la SOA. Cette dernière a été mise en avant afin de permettre des interactions entre applications distantes. L'architecture SOA est une méthode de conception basée sur des standards, permettant de créer une infrastructure informatique intégrée capable de répondre rapidement aux nouveaux besoins d'un utilisateur. Elle fournit les principes et directives permettant de transformer un réseau existant de ressources informatiques hétérogènes, distribuées, complexes et rigides en ressources intégrées, simplifiées et particulièrement souples pouvant être modifiées et combinées afin de mieux satisfaire les objectifs de l'utilisateur **[Boukhadra 2011]**.

Pour répondre aux enjeux de la SOA, les Web services constituent une des technologies actuelles la mieux placée pour mettre en place l'architecture orientée services. Les Web services, qui sont basés sur des technologies Web dérivées du fameux standard XML, présentent de nombreux atouts pour faire communiquer des systèmes caractérisés par une hétérogénéité croissante. Ils permettent également de mettre en œuvre des services applicatifs partagés et de gérer la connectivité aux données. Ils sont devenus la technologie la plus utilisée pour l'interopérabilité et l'intégration des applications et des systèmes d'information. Dans cette perception, la composition des Web services est devenue une solution adéquate pour implémenter les processus métiers fortement distribués et pour répondre, d'une manière efficace et rapide, aux besoins aux requêtes des internautes.

Dans ce premier chapitre, on va élaborer un état d'art sur les services Web, l'Architecture Orientée Service ainsi que les différentes technologies et standards utilisés dans le domaine des services Web.

I.2. Architecture Orientée Service « SOA »

I.2.1. Introduction

L'architecture SOA est un modèle abstrait, qui définit un système par un ensemble d'agents logiciels distribués, qui fonctionnent de concert afin de réaliser une fonctionnalité globale préalablement établie. SOA se présente comme un style architectural. Elle fournit un ensemble de méthodes pour le développement et l'intégration de systèmes dont les fonctionnalités sont développées sous forme de services. L'approche ultime de cette vision consiste, donc, à créer des applications constituées uniquement de services qui interagissent entre eux. Dans ce cas, peu importe où est déployé le service, ce qui importe est que le service remplisse un rôle bien précis **[Schreiner 2005]**.

I.2.2. Définition de l'architecture SOA

Plusieurs définitions sont utilisées pour définir et expliquer l'architecture SOA. La plupart de ces définitions insiste sur les aspects techniques de l'architecture, alors que, d'autres s'intéressent aux caractéristiques métiers. Les définitions suivantes illustrent différentes vues de la SOA. Cependant, elles convergent toutes vers un seul sens :

I.2.2.1. Définition Métier

« *L'architecture orientée service est un ensemble de méthodes techniques, métiers, procédurales, organisationnelles et gouvernementales pour réduire ou éliminer les frustrations avec les technologies d'information, et pour mesurer quantitativement la valeur métier des technologies d'information, pendant la création d'un environnement métier agile pour un intérêt concurrentiel.* » **[Margolisand 2007]**.

I.2.2.2. Définition Technique

« *Une architecture SOA est une structure d'intégration de processus métier qui supporte une infrastructure des technologies d'information comme étant des composants et services sécurisés, standardisés et qui peuvent être combinés pour s'adresser aux priorités de changements métiers.* » **[Jennings 2007]**.

I.2.2.3. Définition Technique plus Large

« *L'architecture SOA est un paradigme permettant d'organiser et d'utiliser des savoir faires distribués pouvant être de domaines variés. Cela fournit un moyen uniforme d'offrir, de découvrir, d'interagir et d'utiliser des savoirs faires pour produire le résultat désiré avec des préconditions et des buts mesurables.* » **[Josuttis 2007]**.

Du point de vue des applications, l'architecture orientée services permet le développement d'une nouvelle génération d'applications dynamiques ou composites. Ces applications permettent aux utilisateurs d'accéder à des informations et à des processus hétérogènes, et de les utiliser de différentes manières, notamment via le Web. Du point de vue de l'infrastructure, l'architecture orientée services permet au service Web de simplifier l'intégration des applications et des systèmes, de recombiner et de réutiliser les fonctionnalités des applications, et d'organiser les différentes phases du processus de développement, dans un cadre cohérent et unifié. En réalité, la philosophie des SOA décompose une application monolithique en une suite de services assurant la modularité dans leurs fonctionnalités **[Devaux 2008]**.

I.2.3. Historique de l'architecture SOA

Dans les années 80, l'architecture orientée objet est apparu comme une nouvelle philosophie de développement basée sur des concepts intéressants : encapsulation, héritage, . . . etc. Le problème de l'approche objet est le fait d'offrir juste la réutilisation technique sans aucune vue métier. Cette architecture rend difficile l'intégration d'applications résidentes dans plusieurs plateformes. Enfin, l'architecture objets est fragile, car un service de ce système est offert comme une méthode d'une classe implémentée par un objet [**Bieberstein 2008**]. Pour pallier les défauts de l'approche objet, l'approche composant est apparue. Le modèle de composants le plus commun dans l'environnement Windows est le COM. Les composants COM dans chacune de ses couches peuvent être réutilisés par d'autres composants et applications. Dans le monde Java, CORBA est le plus utilisé. Il rend possible la distribution des tâches de développement à travers plusieurs programmeurs, et fait que, le système soit plus robuste, scalable, et maintenable [**Bieberstein 2008**].

Les composants sont des entités logicielles indépendantes. Ils interagissent à l'aide d'une infrastructure qui permet de gérer la communication entre des composants au sein d'un même système ou à travers un réseau, via une décomposition du logique métier en composants distribués. L'architecture de composants distribués a engendré un développement rapide et évolutif d'applications distribuées et complexes. L'écriture des composants pouvant être partagés doit prendre en considération que les différents langages de programmation sont incompatibles. Un composant écrit en C++ peut avoir des anomalies de fonctionnement, s'il est utilisé dans un environnement où le Visual Basic est le langage principal.

L'interopérabilité des plateformes n'était plus facile avec les modèles de composants utilisés ces dernières années. Cette interopérabilité s'avère encore plus difficile si le composant à invoquer est situé au-delà d'un pare-feu [**Erl 2008**]. La mise en œuvre de ces deux architectures soulève des difficultés dans le cadre d'une infrastructure ouverte telle qu'Internet. En effet, ces architectures, bien qu'utilisant un modèle objet distribué, proposent, chacune, sa propre infrastructure. Ce qui impose une forte liaison (un fort couplage) entre les services offerts par les composants et leurs clients. Ainsi, on ne peut assembler que des objets CORBA (ou COM) entre eux. Le résultat est que les systèmes construits à base de ces architectures sont homogènes.

Dans les années 2000, sont apparus les services Web pour pallier à tous ces problèmes, et en ne s'intéressant qu'à la manière d'interagir avec ce service Web sans connaître sa structure ou sa technologie. Une application orientée services Web est simplement l'agrégation de services en une logique simple et en une application unifiée. Cependant, la clé de l'interopérabilité entre les services Web est l'utilisation des protocoles standards, des messages, et des contrats. En outre, après l'avènement du B2C, où les entreprises mettaient en ligne leurs services pour leurs consommateurs à travers des applications Web, celles-ci souhaitaient accroître leurs productivités à l'aide du paradigme B2B. Le B2B repose sur l'échange de produits, d'informations et de services entre entreprises. Ceci implique l'utilisation de services et la collaboration avec des systèmes proposés par d'autres concepteurs, et par conséquent, une maîtrise de l'hétérogénéité [**Papazoglou 2003**]. L'interopérabilité est ainsi devenue une nécessité pour l'entreprise dans le monde du B2B.

C'est justement ce que les services Web apportent par rapport aux solutions dites homogènes. D'une certaine façon, le modèle des services Web n'est pas une révolution, mais une évolution du modèle des composants distribués. Cette évolution est rendue nécessaire par l'utilisation intensive d'Internet [**Papazoglou 2003**].

I.2.4. Caractéristiques de l'architecture SOA

L'approche SOA confère aux entreprises plus de flexibilité dans leur activité en améliorant les applications et l'infrastructure informatique. Elle contribue à réduire les dépenses informatiques. Elle fournit un accès rapide à des informations plus précises et permet à l'entreprise d'identifier et de résoudre plus efficacement les problèmes de flux. Parmi ces caractéristiques, on peut citer les plus primordiales suivantes [**Bejaoui 2008**] [**Kaabi 2008**] [**Albertella 2009**] :

- **Autonomie**

Le service est indépendant de son environnement tout en gardant des relations de collaborations. Cette notion d'indépendance se traduit, aujourd'hui, par le fait qu'un service est indépendant du serveur car il a son propre conteneur qui peut être une simple application. Le service est doté d'une sécurité autonome, et il doit protéger ses fonctions et ses messages envoyés sans avoir besoin de connaitre le degré de sécurité des clients.

- **Frontières explicites**

Les messages d'interaction avec un service sont sous forme de contrat en XML pour assurer l'interopérabilité. Les messages permettent de créer des systèmes faiblement couplés qui recouvrent plusieurs systèmes d'exploitation.

- **La plateforme**

La plateforme utilisée pour supporter une implémentation d'un service ne doit pas être pertinente aux consommateurs. Ceci inclut les couches intermédiaires du système d'exploitation, du protocole de communication et même les couches de l'application. Cependant, l'architecture liant les demandeurs et fournisseurs de services est sous un contrôle direct. On peut gagner des avantages significatifs dans les performances et dans la gestion du système en évitant la séparation de cet aspect de l'architecture.

- **Architecture décentralisée**

Parmi les grands avantages des SOA, est la décentralisation de l'architecture avec orchestration automatique. L'emplacement de différentes instances d'un service dans différentes localisations est l'un des facteurs primordiaux, pour avoir une architecture moins couplée. La scalabilité d'une livraison d'un service est facile à atteindre en mettant plusieurs instances de

celui-ci dans des terminaux différents. L'identité d'un service fournisseur peut être négociée à travers un composant, jouant le rôle d'intermédiaire. La négociation peut être selon le critère géographique de la localisation du service, le critère de l'identité du client, d'une information sur le plan de l'adhésion des membres ou d'autres critères d'attachement des clients à leurs fournisseurs de services.

- **Les protocoles**

De même que les considérations de la plateforme de déploiement, l'indépendance des protocoles peut ou ne pas être nécessaire en fonction du modèle métier et du contexte. Comme l'exemple d'une entreprise de vente en détail, où les protocoles entre l'entreprise et ses branches sont totalement contrôlés par l'entreprise. Ainsi, le choix des protocoles peut être unifié, bien que ceci ne limite pas l'exposition d'un service pour être utilisé par d'autres protocoles en même temps. En pratique, les protocoles de communications sont définis d'habitude par un fichier de configuration dans l'interface d'un service SOA. Ceci permet de modifier cette configuration au besoin sans être obligé de modifier le code du service.

- **Le langage de programmation**

Une SOA doit être implémentée indépendamment des spécifications des langages de programmation. Cependant, la pratique a révélé quelques problèmes d'interopérabilité entre les demandeurs et les fournisseurs de services en matière de représentation de types de données très complexes (tableaux, pointeurs nuls, . . . etc.). Ils sont implémentés différemment dans différents systèmes et qui ont différents comportements en performances.

- **Les modèles d'invocation**

Un modèle d'invocation est la circulation générale des flux d'interactions entre un demandeur et son fournisseur de service. Un service peut avoir besoin d'être synchronisé avec un autre service, il peut avoir aussi besoin de fonctionner d'une façon asynchrone si le modèle le permet.

- **Le modèle d'un service**

On peut décrire les différents services et leurs relations chacun avec l'autre en termes d'un modèle, qui peut être sauvegardé, échangé et utilisé pour automatiser les interactions et faciliter la création d'une orchestration des services. Ceci nous permet d'utiliser dans plusieurs cas les mêmes services, pour créer des modèles d'interaction différents, et pour la construction de nouveaux domaines ou de nouvelles solutions.

I.2.5. Conclusion

Le paradigme orienté services représente une nouvelle tendance d'ingénierie logicielle. Il assure un développement d'applications plus rapide et à moindre coûts. SOA est une architecture fournissant une infrastructure nécessaire, pour intégrer les applications isolées afin de les utiliser en tant que services dans un réseau. La SOA est entrée depuis peu de temps dans le domaine du réel, grâce à un ensemble de normes appelées collectivement services Web. Les services Web, réalisation concrète des architectures SOA, sont la déclinaison du paradigme des architectures orientées service, sur le Web. Ces services Web constituent un moyen de plus en plus normalisé, étendu et puissant pour mettre en œuvre une architecture SOA. Cette architecture est construite autour de la notion de service Web, qui est matérialisée par un composant logiciel assurant une fonctionnalité particulière et accessible via son interface. Un service Web est toujours accompagné d'une description fournissant aux applications les informations nécessaires à son utilisation. En réalité, la philosophie des SOA décompose une application homogène en une suite de services assurant la modularité dans leurs fonctionnalités [**Brown 2008**].

I.3. Les services Web

I.3.1. Introduction

De nos jours, les entreprises expriment un grand besoin pour échanger des informations et des services. Ceci nécessite des langages communs de communication. Les efforts d'élaboration de ces langages ont donné lieu à une technologie émergeante qui a permis de tracer quelques pistes intéressantes pour la communication entre entreprises. Cette technologie est celle de services web [**Fakhfakh 2006**]. Les services Web constituent le développement ultime dans ce domaine. Le terme service Web est souvent utilisé de nos jours, mais pas toujours avec la même signification, car la signification est tributaire de l'application de cette technologie.

I.3.2. Définition des services Web

On peut dire qu'un service Web est souvent vu comme une application accessible à d'autres applications sur le Web, mais il existe plusieurs définitions pour les services Web :

- **Définition 1**

Le consortium W3C définit un service Web comme étant : « *une application, ou un composant logiciel qui vérifie les propriétés suivantes* » [**Cerami 2002**] :

- Il est identifié par un URI.
- Ses interfaces et ses liens peuvent être décrits en XML.
- Sa définition peut être découverte par d'autres services Web.

- Il peut interagir directement avec d'autres services Web à travers le langage XML en utilisant des protocoles Internet standards.

- **Définition 2**

« 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. » **[Melliti 2004]**.

- **Définition 3**

« Les services Web sont la nouvelle vague des applications Web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le Web. Les services Web effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un service Web est déployé, d'autres applications (y compris des services Web) peuvent le découvrir et l'invoquer. » **[Ponge 2008]**.

I.3.3. Caractéristiques des services Web

Plusieurs acteurs définissent les services Web par des caractéristiques technologiques distinctives, qui sont **[Cerami 2002]** :

- **Un service Web est une application logicielle qui est reconnue par un URI**

URI est la façon d'identifier un point de contenu sur le Web comme un document tel qu'un texte, audio ou vidéo. L'URI le plus connue est l'adresse d'une page Web. Le service Web est donc accessible en spécifiant son URI, c'est-à-dire que le service Web est caractérisé par un seul objet et une seule fonctionnalité. A partir de cela, on peut faire la construction d'une application logicielle très large comportant plusieurs fonctionnalités, afin de sélectionner les fonctionnalités qui sont recherchées par les URI spécifiques.

- **Capacité des interfaces et liaisons d'être publiées, localisées et invoquées via le langage XML**

Les principales tâches d'un service Web sont : La publication dans un registre, la localisation en interrogeant le registre qui l'héberge et l'invocation par un ou plusieurs services Web après sa localisation. Ces tâches sont réalisées via l'utilisation d'XML.

- **Capacité d'interagir avec les composantes des logiciels via des éléments XML avec l'utilisation des protocoles Internet standards**

Un service Web est créé pour être utilisé et interagir avec d'autres logiciels contrairement à une page Web, ou à une autre application qui n'utilise pas les services Web. C'est l'interopérabilité basée sur l'utilisation de l'XML et les protocoles Internet standards, par exemple, HTTP, SMTP, FTP, . . . etc.

- **Composante logicielle légèrement couplée à interaction dynamique**

Un service Web ayant un programme qui permet de l'invoquer est appelé consommateur de service Web. Le service Web et son consommateur sont indépendants l'un de l'autre. Si une modification est à faire sur le consommateur, on n'a pas besoin de connaitre la machine, le langage de programmation, le système d'exploitation ou autre paramètre, afin d'établir à nouveau une communication entre le service Web et son consommateur. Le consommateur possède une fonctionnalité qui correspond à faire une localisation et une invocation sur le service Web, au moment de l'exécution du programme de service Web de manière automatique.

I.3.4. Fonctionnement des services Web

Le fonctionnement des services Web s'articule autour de trois acteurs principaux illustrés par le schéma suivant :

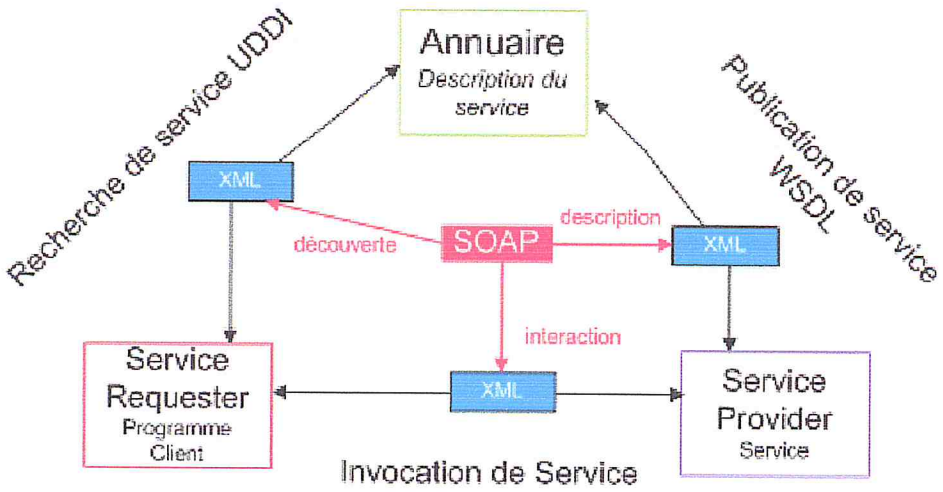


Figure I.1 « Fonctionnement des services Web » **[Chab 2007]**

On peut décortiquer ce schéma de la manière suivante :

- **Service provider service :**

Le fournisseur de service met en application le service Web et le rend disponible sur Internet.

- **Service requester programme client :**

C'est n'importe quel consommateur du service Web. Le demandeur utilise un service Web existant en ouvrant une connexion réseau et en envoyant une demande en XML.

- **Annuaire service registry :**

Le registre de service est un annuaire de services. Le registre fournit un endroit central où les programmeurs peuvent publier de nouveaux services ou en trouver.

Les interactions entre ces trois acteurs suivent plusieurs étapes :

- **La publication du service** : Le fournisseur diffuse les descriptions de ses services Web dans l'annuaire.
- **La recherche du service** : Le client cherche un service particulier, il s'adresse à un annuaire qui va lui fournir les descriptions et les URL des services demandés afin de lui permettre de les invoquer.
- **L'invocation du service** : Une fois que le client récupère l'URL et la description du service, il les utilise pour l'invoquer auprès du fournisseur de services.

I.3.5. Avantages des services Web

L'idée essentielle derrière les services Web est de partager les applications et les programmes en un ensemble d'éléments réutilisables appelés service, de sorte que, chacun de ces éléments effectuent une tâche principale et efficace, afin de faciliter l'interopérabilité entre tous ces services Web. D'autre part les services Web **[Chappell 2002]** :

- Permettent l'interopérabilité dans des environnements applicatifs, cela veut dire, que les logiciels et les applications écrits dans différents langages de programmation, et évoluant sur différents systèmes d'exploitation peuvent communiquer et/ou échanger des données entre eux facilement.
- Permettent de profiter de différents environnements et langages de développement par une publication, localisation, description et une invocation via XML. Les services Web sont très flexibles, indépendants des langages de programmation et des systèmes d'exploitation.

- Permettent d'accéder aux applications à travers les pare-feu à l'aide d'utilisation via le langage XML et les protocoles Internet standards comme HTTP sur le port 80, qui est généralement ouvert. Cela permet d'assurer une transmission des données transactionnelles et sécurisé.
- Utilisables à distance via n'importe quel type de plateforme, et sont accessibles depuis n'importe quel type de clients.
- Peuvent servir au développement d'applications distribuées. Ils appartiennent à des applications capables de collaborer entre elles de manière transparente pour l'utilisateur, et permettent d'avoir un partage des fonctionnalités et facilitent grandement le développement.

I.3.6. Architecture des services Web

Techniquement, un service Web peut donc être perçu comme étant une interface décrivant une collection d'opérations accessibles via le réseau, à travers des messages XML standardisés. D'un point de vue technique, la description d'un service Web inclut tous les détails nécessaires à l'interaction avec le service constituant, ce qu'on appelle l'architecture des services Web, par exemples, le format des messages, les signatures des opérations, le protocole de transport et la localisation du service Web [**Kreger 2001**].

L'interopérabilité est l'objectif premier des services Web. Pour permettre cet échange d'information entre des applications distantes, les services Web sont composés des couches standards. Deux types d'architecture existent pour les services Web : La première dite de référence, elle contient trois couches principales. La seconde architecture est plus complète, elle utilise les couches standards de la première architecture en ajoutant au-dessus d'autres couches plus spécifiques. Elle est appelé architecture étendue ou encore en Pile [**Kreger2001**].

I.3.6.1.Architecture de référence

Cette architecture vise trois objectifs importants [**Kreger2001**] :

- Identification des composants fonctionnels.
- Définition des relations entre ces composants.
- Etablissement d'un ensemble de contraintes sur chaque composant de manière à garantir les propriétés globales de l'architecture.

L'architecture de référence s'articule autour des trois rôles suivants (Figure I.2) :

- **Le fournisseur de service :** Correspond au propriétaire du service Web. D'un point de vue technique, il est constitué par la plateforme d'accueil du service Web.

- **Le client :** Correspond au demandeur de service Web. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service Web. L'application cliente peut-être elle-même un service Web.

- **L'annuaire des services :** Correspond à un registre de descriptions des services Web offrant des facilités de publication des services Web à l'intention des fournisseurs, ainsi que des facilités de recherche des services Web à l'intention des clients.

Les interactions de base entre ces trois rôles incluent les opérations de publication, de recherche et de liens d'opérations. Nous citons, notamment les standards émergents suivants:

SOAP : standard supporté par le W3C, il définit un protocole de transmission de messages basé sur XML.

WSDL : soumise en tant que note auprès du W3C en mars 2001, il introduit une grammaire pour la description des services Web.

UDDI : standard supporté par le consortium OASIS, il fournit l'infrastructure de base pour la publication, et la découverte des services Web.

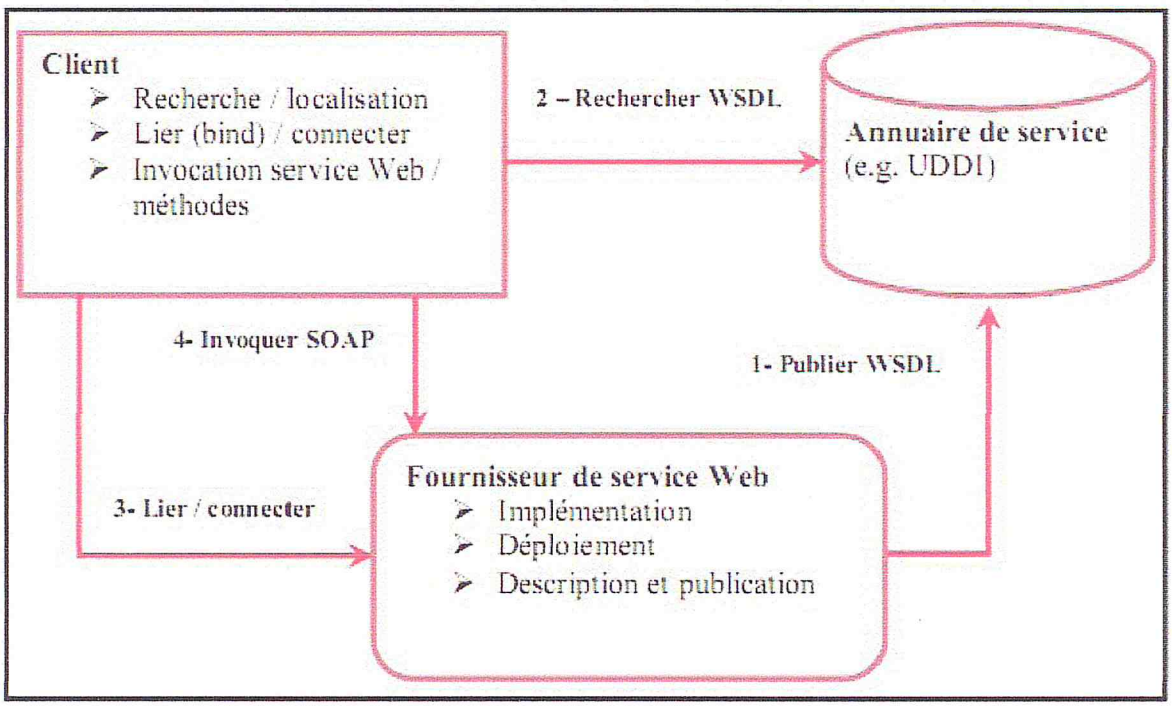


Figure I.2 : « Architecture de référence des services Web » [Kreger 2001].

I.3.6.2. Architecture étendue

Une architecture étendue est constituée de plusieurs couches se superposant les unes sur les autres. La pile est constituée de plusieurs couches, chaque couche 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.

Nous apportons une explication de la mise en relief des trois types de couches (Figure I.3).

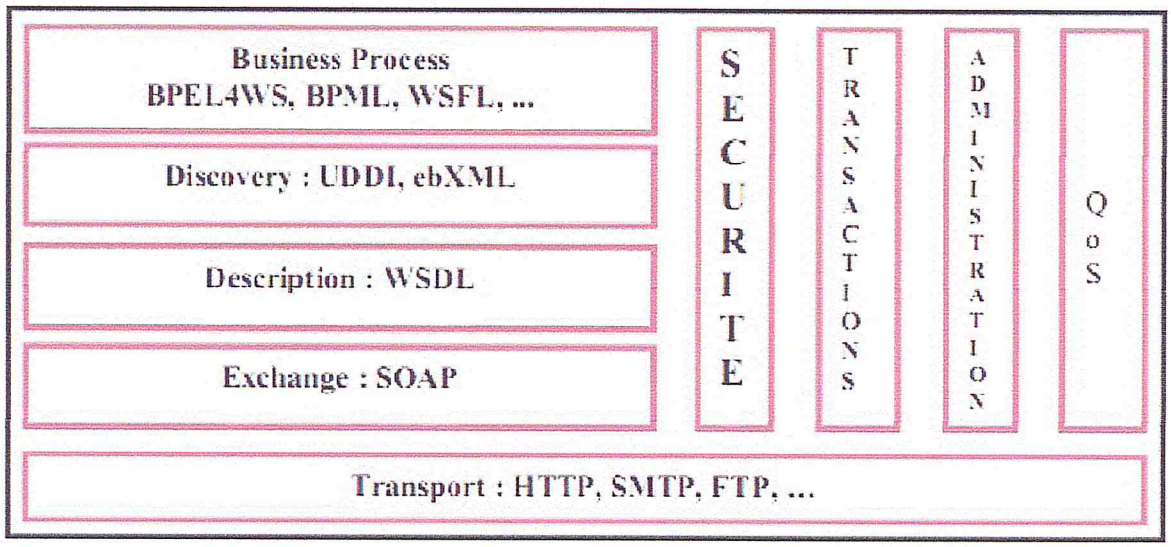


Figure I.3 «Architecture en Pile des services Web » [Clain 2000].

- **Infrastructure de base (Discovery, Discription, Exchange) :**

Ce sont les fondements techniques établis par l'architecture de référence. Nous distinguons les échanges des messages établis par SOAP (W3C), la description de service par WSDL (W3C) et la recherche de services Web que les organisations souhaitent utiliser via le registre UDDI (OASIS).

- **Couches transversales (Security, Transactions, Administration, QoS) :**

Ce sont des couches qui rendent viable l'utilisation effective des services Web dans le monde industriel.

- **La couche Business Processus (BusinessProcess):**

Cette couche supérieure permet l'intégration de services Web, elle établit la représentation d'un « *BusinessProcess* » comme un ensemble de service Web. De plus, la description de

l'utilisation de différents services composant ce service est disponible par l'intermédiaire de cette couche.

- **La couche Transport :**

La couche de transport est la base de la pile de protocoles nécessaires aux services Web. Elle est responsable de la transmission des messages XML entre les applications. Le protocole de transport le plus couramment utilisé est le protocole http.

La technologie des services Web offre de fortes potentialités pour surmonter les problèmes d'interopérabilité des systèmes. Elle constitue un cadre prometteur pour l'intégration des applications, et pour la gestion des interactions entre divers partenaires dans un environnement distribués, hétérogènes, ouvert et versatile qui est le Web [**Bertino 2004**].

I.4. Les principales technologies standards autour des services Web

Les Services Web reprennent la plupart des idées et principes du Web (HTTP, XML), et les appliquent à des interactions entre machines. Comme pour le World Wide Web, les Services Web communiquent via un ensemble de technologies fondamentales qui partagent une architecture commune. Ils ont été conçus pour être réalisés sur de nombreux systèmes développés et déployés de façon indépendante. Les technologies utilisées par les Services Web sont HTTP, WSDL, XML, SOAP et UDDI.

I.4.1. HTTP

HTTP « *Hyper Text Transfer Protocol* » est le plus populaire des protocoles Internet, il a été créé au début des années 1990 pour aider les scientifiques à trouver et partager de l'information. HTTP c'est rapidement répandu dans le domaine d'internet puisque les sites Web l'utilisent comme leur mécanisme de communication, cela ne signifie pas que HTTP est limité au World Wide Web **[Fielding et al, 1999]**. En fait, le Web est juste une application utilisant HTTP pour transmettre les informations entre les serveurs et les clients.

Les premières versions de la spécification des services web se sont basées uniquement sur HTTP comme moyens de transport et de communication entre les clients et les services. En conséquence, de nombreux services en ce moment-là utilisent HTTP, et il est le plus commun de tous les protocoles de transport. **[Fielding et al, 1999]**

I.4.1.1. Principes

HTTP repose sur un ensemble de principes très simple :

- **Intégré dans le système d'adressage URI/URL :**

Les URI (*Uniform Resource Identifiers*) se composent de deux sous-ensembles, les URL (*Uniform Resource Location*) et les URN (*Uniform Resource Name*). HTTP est l'un des services les plus utilisés dans les URL. Ce système d'adressage permet non seulement d'identifier les ressources publiées sur le Web mais également de décrire les liens entre ressources. **[Fielding et al, 1999]**

- **Protocole client/serveur :**

HTTP est un protocole client/serveur (asymétrique) : un client envoi une requête et le serveur lui envoie une réponse.

- **Protocole sans connexion et sans état :**

Le protocole HTTP par lui-même est sans connexion et sans état, c'est-à-dire que chaque requête est considérée de manière indépendante sans qu'il ne soit demandé au serveur de mémoriser aucun contexte. Cela n'interdit pas à un serveur Web de simuler une connexion et de maintenir un état, mais ces pratiques devraient être limitées au maximum afin de respecter les principes architecturaux du Web.

- **S'appuie sur les types mime « Internet Media Types » :**

Les documents transmis par les requêtes et les réponses sont décrits en suivant les types MIME (Multipurpose Internet Mail Extensions) rebaptisés (Internet Media Types), ce qui permet de transmettre tout type de documents, y compris en format binaire.

I.4.1.2. Fonctionnement

Le fonctionnement du protocole HTTP est basé sur le modèle client-Serveur. Le client est toujours l'initiateur d'une communication, car c'est lui qui envoie une requête au serveur. Ce dernier réagit en fonction de la requête reçue.

Le scénario de dialogue classique entre un navigateur et un serveur Web est le suivant. Le navigateur Web client établit une connexion TCP avec le serveur Web qui contient la page qui l'intéresse. Une fois la connexion établie, le client émet une requête HTTP contenant une commande, une URL, et parfois d'autres informations. Lorsque le serveur Web reçoit la requête il essaie d'exécuté la commande qu'elle contient. Il retourne ensuite comme réponse le résultat obtenu qui peut être des données, un message d'erreur, et d'autres informations. Au moment où le serveur effectue le traitement de la demande, la connexion entre le client et le serveur reste ouvert et le client est effectivement bloqué en attente de la réponse. Une fois que le client a reçu sa réponse la connexion est fermée et détruit.

I.4.1.2.1. Requête HTTP

Une requête HTTP permet au navigateur web de communiquer avec le serveur, Elle se compose des éléments suivants :

- Une ligne de requête (3 éléments séparés par un espace) qui précise la méthode, l'URL et la version du protocole utilisé par le client (généralement HTTP1.1)
- Les champs d'en-tête de la requête : ensemble de lignes facultatives donnant des informations supplémentaires sur la requête et/ou le client (navigateur ; OS)
- Le corps de la requête : ensemble de lignes optionnelles séparées des lignes précédentes par une ligne vide.

Une requête exprimée par un client HTTP aura la syntaxe suivante :

<Méthode><URL>http/<version>
[<entête 1> :<valeur 1>]
…
[<entête i> :<valeur i>]
(Ligne vide)
Corps de la requête

I.4.1.2.2. Exemple de requête HTTP

La figure suivante représente une requête envoyée par un client au serveur.

GET /index.html /HTTP/1.1

Host : poweredge.paris.dyomedea.com

User-Agent : Mozilla/5.0 (X11 ; U ; Linux i686 ; en-US ; rv :1.6)

Gecko/20040413 Epiphany/1.0.8

Accept :text/xml,application/xml,application/xhtml+xml,text/html ;q=0.9,text/

Plain ;q=0.8,image/png,image/jpeg ,image/gif ;q=0.2,*/* ;q=0.1

Accept-Language : en,en ;q=0.5

Accept-Encoding : gzip,deflate

Accept-Charset :ISO-8859-1,uft-8 ;q=0.7,* ;q=0.7

Keep-Alive : 300

Connection : keep-alive

Cookie : Apache=10.0.0.2.13945105953845970

La requête est exprimée par la première ligne : la « méthode » est GET qui signifie « donne-moi un document », le document est « /index.html », la version du protocole http utilisé ici est « 1.1 » et il se trouve sur la machine « poweredge.paris.dyomedea.com ».

Les autres lignes sont des entêtes qui apportent des informations complémentaires sur le contexte dans lequel cette requête est effectuée.

I.4.1.2.3. Principales Méthodes Utilisées

Dans le protocole HTTP, une méthode est une commande spécifiant un type de requête, c'est-à-dire qu'elle demande au serveur d'effectuer une action. En général l'action concerne une ressource identifiée par l'URL qui suit le nom de la méthode. Les méthodes les plus utilisées sont :

- **GET :** La sémantique de la méthode « GET » est la demande d'une ressource située à l'URL spécifique dans la requête. L'utilisation de GET est « sûre », c'est-à-dire que l'état de la ressource ne doit pas être modifié par un GET. Ceci autorise les liens, la mise en cache, les favoris. Il faut donc utiliser GET pour des opérations qui ressemblent à des questions ou à des lectures de l'état de la ressource.

Exemple de requête HTTP GET

GET /season1/index-fall.html HTTP/1.1

Host : www.joes-hardware.com

- **HEAD :** La sémantique de la méthode « HEAD » est la même que celle de la méthode « GET » sauf que seuls les entêtes (headers) sont renvoyées dans la réponse. Cette méthode permet donc de tester si un document existe sans le télécharger et de vérifier son type, sa date de modification, …etc.

Exemple de requête HTTP HEAD

HEAD / season1/index-fall.html HTTP/1.1

Host : www.joes-hardware.com

- **POST :** La sémantique de la méthode « POST » est le transfert d'information du client vers le serveur en utilisant le corps de la requête. Elle permet d'ajouter une nouvelle ressource (un message sur un forum ou un article dans un site). L'URL fournit est l'URL d'une ressource associé à la nouvelle ressource (comme l'URL du forum ou site) et non l'URL de la ressource nouvellement crée.

Exemple de requête HTTP POST

POST /inventory-check.cgi HTTP/1.1

Host : www.joes-hardware.com

Content-Type : text/plain

Content-Lenght : 18

Item=bandsaw 2647 ----------> Corps de Requête http

- **PUT :** La sémantique de la méthode « PUT » est la mise à jour d'une ressource accessible sur le serveur. La requête PUT est accompagnée du document à publier et l'URL associé est l'adresse à laquelle ce document doit être publié.

Exemple de requête HTTP PUT

PUT /product-list.txt HTTP/1.1

Host : www.joes-hardware.com

- **DELETE :** La sémantique de la méthode « DELETE » est l'effacement physique ou logique d'une ressource accessible sur le serveur.

Exemple de requête HTTP DELETE

DELETE / product-list.txt HTTP/1.1

Host : www.joes-hardware.com

I.4.1.3. Réponse HTTP

Une réponse HTTP est un ensemble de lignes envoyé par le serveur au navigateur web.

Elle comprend :

- Une ligne de statut qui indique la version du protocole utilisé, le code de statut et la signification du code.
- Les champs d'en-tête de la réponse : ensemble de lignes facultatives donnant des informations supplémentaires sur la réponse et/ou le serveur.
- Le corps de la réponse : contient le document demandé

Une réponse émise par le serveur aura la forme suivante :

HTTP/<version><statut (code retour)><commentaire>

[<entête 1> :<valeur 1>]

…

[<entête i> :<valeur i>]

(Ligne vide)

Corps de la réponse.

I.4.1.3.1. Exemple de réponse HTTP

La réponse du serveur concernant la requête précédente est la suivante :

HTTP/1.1 200 OK

Date : Thu, 10 Jun 2004 12 :19 :17 GMT

Server : Apache/1 .3.26 5Unix) Debian GNU/Linux PHP/4.1.2

ApacheJServ/1.1.2

Last-Modified : Tue, 27 Aug 2002 16 :21 :3 6 GMT

ETag : „" 3a802b-120-3d6ba710 „"

Accept-Ranges : bytes

Content-Lenght : 288

Keep-Alive : timeout = 15, max=100

Connection : Keep-Alive

Content-Type : text /html ; charset =iso-8859-1

<html> ..... </html>

La première ligne donne le compte rendu (ici, « 200 » signifie OK). Les lignes suivantes sont des headers donnant des informations complémentaires suivis, après un saut de ligne, par le contenu de la réponse (ici un document HTML).

I.4.1.3.2. Les codes de statut

La réponse envoyée par un serveur contient un code de statut composé de 3 chiffres. Ce code informe le client sur le résultat de la requête (requête invalide, document non trouvé, ...etc).

Les principales valeurs des codes de statut HTTP sont :

- 10x : Messages d'informations. Ces codes ne sont pas utilisés dans HTTP 1.0.
- 20x : Réussite. Ces codes indiquent le bon déroulement des transactions. Exemple : 200 ok.
- 30x : Message de redirection. Ces codes indiquent que la ressource n'est plus à l'emplacement indiqué.
- 40x : Erreur due au client. Ces codes indiquent que la requête est incorrecte.
- 50x : Erreur due au serveur. Ces codes indiquent qu'il y a eu une erreur interne du serveur.

I.4.1.4. Portées et limites

HTTP est relativement simple à comprendre et à coder. Sa légèreté, sa rapidité ainsi que sa large utilisation sur internet lui ont permis de devenir le protocole de transport de prédilection des Services Web. Les serveurs Web sont conçus pour traiter un grand nombre de demandes et la plupart des pare-feu permettent le trafic HTTP sans aucune configuration spéciale. Le protocole a été conçu pour accepter des données textuelles dans les demandes (ou codés en binaire ou ASCI), de sorte qu'il peut facilement manipuler XML des deux extrémités de la communication.

Bien que HTTP soit devenu omniprésent et normalisé. Il ne garantit pas la livraison des données entre clients et serveurs. Comme nous l'avons dit précédemment, Les clients basés sur HTTP sont obligés de bloquer jusqu'à ce qu'ils reçoivent une réponse de service. Ce genre de comportements peut entraver l'évolutivité et il n'est souvent pas souhaitable dans les systèmes d'entreprise. **[Fielding et al, 1999]**

I.4.2. XML

XML constitue la technologie de base des architectures Web services ; c'est un facteur important pour contourner les barrières techniques. XML est un standard qui permet de décrire des documents structurés transportables sur les protocoles d'Internet. En effet, il apporte à l'architecture des Web services l'extensibilité et la neutralité vis à vis des plates-formes et des langages de développement.

La technologie des Web services a été conçue pour fonctionner dans des environnements totalement hétérogènes. Cependant, l'interopérabilité entre les systèmes hétérogènes demande des mécanismes puissants de correspondance et de gestion des types de données des messages entre les différents participants (clients et fournisseurs). C'est une tâche où les schémas de type de données XML s'avèrent bien adaptés. C'est pour cette raison que la technologie des Web services est essentiellement basée sur XML ainsi que les différentes spécifications qui tournent autour (les espaces de nom, les schémas XML, et les schémas de Type).

I.4.3. Le Protocol SOAP

SOAP (W3C), veut dire « Simple Object Access Protocol » et la traduction de cette définition en français donnerait « Protocole Simple d'Accès aux Objets ». En effet, le protocole SOAP consiste à faire circuler du XML via du HTTP sur le port 80. Cela facilite grandement les communications, car le XML est un langage standard et le port utilisé est le port 80 qui ne pose pas de problèmes pour les firewalls. SOAP peut donc être utilisé dans tous les styles de communication : synchrone ou asynchrone, point à point ou multipoint, Intranet ou Internet [**Kulchenko 2001**].

Le SOAP est un protocole à la fois simple et facile à implémenter dans les serveurs Web, destiné à l'échange d'informations dans un environnement distribué et décentralisé [**Gardien 2002**]. Le SOAP fait partie de la couche de communication des services Web. La force de ce protocole réside dans son universalité et sa flexibilité. Il définit la structure des messages XML utilisés par les applications pour dialoguer entre elles.

La figure 1.4 montre un exemple de message SOAP requête d'un service web qui additionne deux entiers.

```
<? xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv: Header/>
<soapenv:Body>
<Addition xmlns="http://doc">
<a>4</a>
<b>7</b>
</Addition>
</soapenv:Body>
</soapenv:Envelope>
```

Figure I.4 « Le formatage visuel d'un message SOAP » **[Ben Halima 2009]**.

Le message est englobé dans une enveloppe et divisés en 2 parties : l'entête et le corps.

- L'entête (*Header*) : offre des mécanismes flexibles pour étendre un message SOAP sans aucune préalable connaissance des parties communicantes. Les extensions peuvent contenir des informations concernant l'authentification, la gestion des transactions, le payement, etc.

- Le corps (*Body*) : offre un mécanisme simple d'échange des informations mandataires destinées au receveur du message SOAP. Cette partie contient les paramètres fonctionnels tels que le nom de l'opération à invoquer, les paramètres d'entrés et de sortis ou des rapports d'erreur.

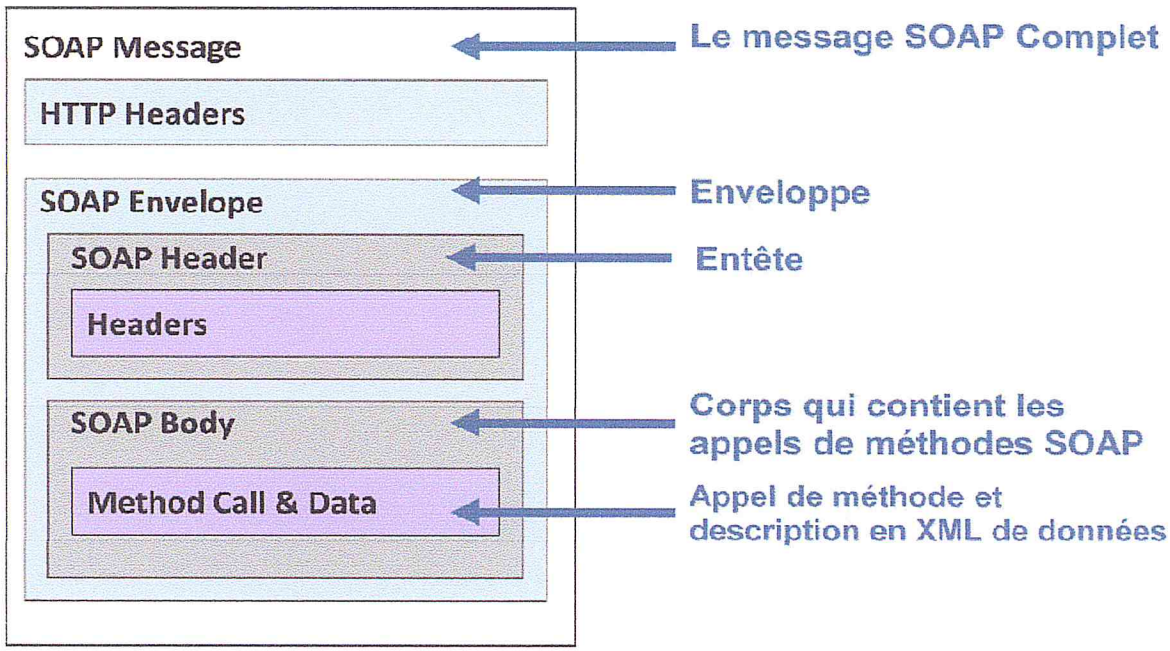


Figure I.5 « les composants d'un message SOAP » [Ben Halima 2009].

I.4.4. Le langage WSDL

Le langage WSDL (W3C), est l'acronyme de « Web Service Description Langage »est un langage basé sur XML permettant de décrire et de publier les interfaces et protocoles des services Web d'une manière standard. L'interface d'un service Web décrit en fait tout le fonctionnement d'un service Web, et cache tout le détail de l'implémentation du service Web, donc elle est indispensable pour pouvoir invoquer un service Web par une application cliente, ou un autre service Web permettant une utilisation indépendante de la plateforme utilisée ainsi que du langage utilisé [Scott 2002].

Le langage WSDL présente un format commun pour la description et la publication des interfaces et protocoles relatifs aux services Web. Une description WSDL d'un service Web est faite sur deux niveaux, niveau abstrait et niveau concret. Au niveau abstrait, la description du service Web consiste à définir les éléments de l'interface du service Web [Chinnici 2004] [Gardien 2002].

I.4.4.1. La structure d'un document WSDL

Une instance WSDL est un document XML qui dispose d'un élément racine <definition> spécifiant l'espace de noms WSDL et qui définit une série de services Web, comme une collection de points de terminaison réseau ou de ports.

Un fichier WSDL commence par <Defenition> et finit par </ Definition>. C'est à l'intérieur de cette espace qu'on déclare tous les éléments constituants la description.
La figure I.6 ci-dessus représente la structure d'un document WSDL :

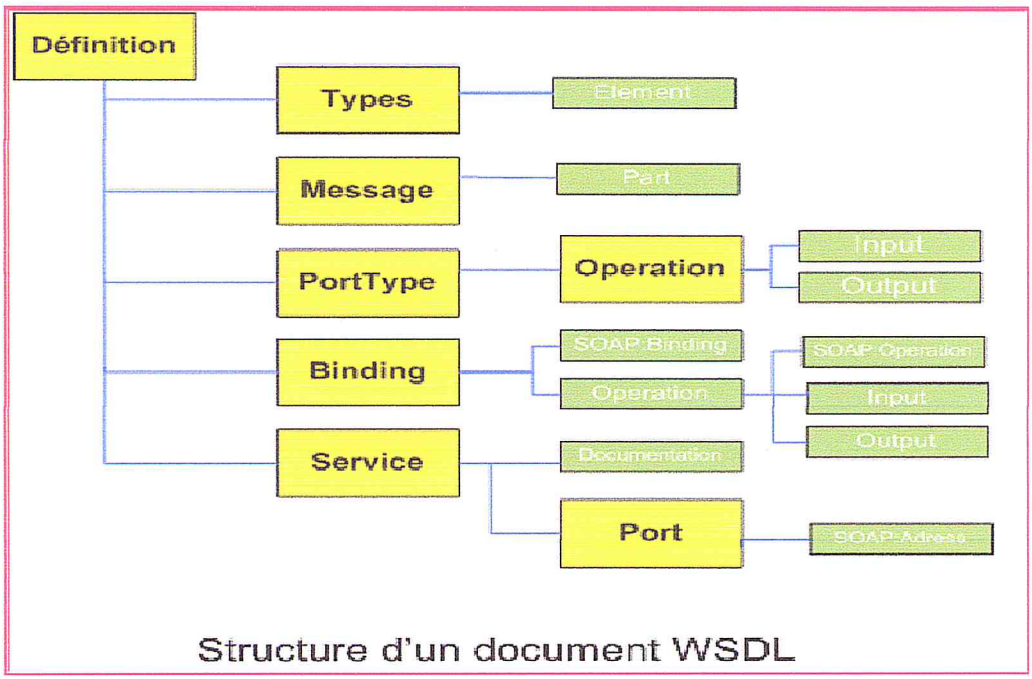


Figure I.6 « La structure d'un document WSDL » **[Chab 2007]**

- **Type**

« Data types » est l'élément qui définit les types de données utilisées dans les messages échangés par le service Web. Une fois définie, les « Data types », ou type peuvent être référencés dans n'importe quel message.

```
<definition targetNamespace= "http://WS/ name="WSGestionService">
<types>
<xsd:schema><xsd:import namespace="http://WS/
schemaLocation="http://localhost:8080/GestionEDT/WSGestionService?xsd=1"/>
</xsd:schema></types>
```

Figure I.7 « Le formatage visuel d'un Type » **[Ben Halima 2009]**.

- **Messages**

L'élément « Message »spécifie les types d'opérations supportées par le service Web, il permet d'incorporer une séquence de messages corrélés sans avoir à spécifier les caractéristiques du flux de données, par exemple, un message Input et un message Output corrélés sont mis en correspondance dans une seule opération de type « Request/Response ».

```
<message name= "AuthentificationAdmin ">
<part name="parameters" element="tns:AuthentificationAdmin"/>
</message>
<message name= "AuthentificationAdminReponse ">
<part name="parameters" element="tns:AuthentificationAdminReponse"/>
</message>
```

Figure I.8 « Le formatage visuel d'un Request/Response » **[Ben Halima 2009].**

- **Opération**

L'élément « Operation »spécifie les types d'opérations supportées par le service Web, il permet d'incorporer une séquence de messages corrélés sans avoir à spécifier les caractéristiques du flux de données.

- **PortType**

Le « PortType »est un groupement logique ou une collection d'opérations supportées par un ou plusieurs protocoles de transport, il est analogique à une définition d'un objet contenant un ensemble de méthodes.

```
<portType name= "WSGestion ">
<operation name="AuthentificationAdmin"/>
<input wsam :Action="http://WS/WSGestion/AuthentificationAdminRequest"
Message="tns:AuthentificationAdmin"/>
<output wsam :Action="http://WS/WSGestion/AuthentificationAdminReponse"
Message="tns: AuthentificationAdminResponse"/>
</operation>
</portType>
```

Figure I.9 « Le formatage visuel d'un PortType » **[Ben Halima 2009].**

- **Liaison**

Décrit la façon dont un type de port est mis en œuvre pour un protocole particulier (HTTP par exemple), et un mode d'invocation (SOAP par exemple). Cette description est faite par un ensemble donné d'opérations abstraites. Pour un type de port, on peut avoir plusieurs liaisons, pour différencier les modes d'invocation ou de transport de différentes opérations.

Au niveau concret, le service Web est défini grâce aux deux éléments : Port et Service. Ces deux dernières décrivent des informations liées à un usage contextuel du service Web. On y trouve: l'adresse du fournisseur implémentant le service, et le service qui est représenté par les adresses des fournisseurs.

- **Bindings**

L'élément « Port », dans la partie concrète, spécifie une adresse URL qui correspond à l'implémentation du service Web par un fournisseur, et identifie un ou plusieurs « Bindings » (ou liaisons) aux protocoles de transports (HTTP, FTP, . . .) pour un «Port-Type » donné. La séparation du protocole de transport de la définition du « PortType» permet à un service Web d'être valable à travers plusieurs protocoles de transports, sans avoir à redéfinir l'ensemble du fichier WSDL.

La figure I.10 suivante présente un exemple d'un élément « Binding » et son contenu :

```
<binding name= "WSGestionPortBinding "type="tns: WSGestion">
<soap: binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<operation name="AuthentificationAdmin"/>
<soap:operration soapAction=""/>
<input>
<soap: body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation></binding>
```

Figure I.10 « Le formatage visuel d'un Binding » **[Ben Halima 2009].**

- **Service**

Spécifie l'adresse complète du service Web, et permet à un point d'accès d'une application distante de choisir à exposer de multiples catégories d'opérations pour divers types d'interactions.

```
<service name= "WSGestionService ">
<port name="WSGestionPort"binding="tns:WSGestionPortBinding">
<soap: address location="http://localhost:8080/GestionEDT/WSGestionService"/>
</port></service>
```

Figure I.11 « Le formatage visuel d'un Service » **[Ben Halima 2009]**.

I.4.5. L'annuaire UDDI

UDDI (OASIS), a été conçu en 2000 à l'initiative d'un ensemble d'industriels (Ariba, IBM, Microsoft), en vue de devenir le registre standard de la technologie des services Web. Pour convenir à la technologie des services Web, les services référencés dans UDDI sont accessibles par l'intermédiaire du protocole de communication SOAP, et la publication des informations concernant les fournisseurs et les services doit être spécifiée en XML afin que la recherche et l'utilisation soit faite de manière dynamique et automatique. UDDI constitue un méta-service possédant des fonctions de publication et de recherche **[Lopez Velasco2008]**. En clair, l'UDDI gère l'information relative à la publication, la découverte et l'utilisation d'un service Web. Ce standard définit donc, un registre des services Web sous un format XML. Les organisations publient les informations décrivant leurs services Web dans l'annuaire, et l'application client ayant besoin d'un certain service, consulte cet annuaire pour la recherche des informations concernant le service Web qui fournit le service désiré, pour une éventuelle interaction, ainsi l'UDDI a été créé pour faciliter la découverte de services Web en plus de leurs publications. Par une API SOAP, on peut interagir avec l'UDDI au moment de la conception et l'exécution des applications afin de découvrir des données techniques, et administratives sur les entreprises et leurs services Web. L'annuaire UDDI repose sur le protocole SOAP, les requêtes et les réponses sont des messages SOAP **[Scott 2002]** **[Gardien 2002]**.

L'UDDI est subdivisé en deux parties principales : partie publication ou inscription, et partie découverte. La partie publication regroupe l'ensemble des informations relatives aux entreprises et à leurs services. Ces informations sont introduites via une API d'enregistrement. La partie découverte facilite la recherche d'information contenue dans UDDI grâce à l'API SOAP **[Newcomere 2004]**.

L'UDDI peut être vu comme un annuaire contenant les parties suivantes :

- **Les pages blanches (*White Paper*)**

Ce composant permet de connaître les informations à propos de l'organisation proposant le service. Cette description contient toutes les informations jugées pertinentes pour identifier l'organisation (telles que son nom, son adresse physique). Le futur client du service retrouve dans les pages blanches les informations que le fournisseur a renseignées dans l'élément *Business Entity* lors de la publication.

- **Les pages jaunes (*Yellow Paper*)**

Les pages jaunes d'UDDI détaillent la description de l'organisation faite dans les pages blanches en répertoriant les services proposés. Dans cette section, sont décrits : la catégorie de l'entreprise, le secteur d'activité dans lequel exerce l'entreprise, les services offerts par cette organisation, le type de services et les conventions d'utilisation (prix, qualité de service,…etc.). La description des services contenue dans les pages jaunes est non technique et est renseignée par les fournisseurs eux-mêmes.

- **Les pages vertes (*Green Paper*)**

Les pages vertes comportent les informations techniques liées aux services Web et basées sur leur description WSDL. À l'origine, il existait des registres UDDI dits publics (tels que ceux de Microsoft ou IBM) pour lesquels n'importe qui pouvait devenir, soit fournisseur, soit client de services Web. L'universalité de ces registres devait amener UDDI à devenir le standard de publication des services Web. En 2006, le nombre de services Web publiés a atteint le nombre de 50000. Malgré ce nombre, UDDI n'a jamais atteint son but : devenir le registre standard des services Web **[Lopez-Velasco 2008]**.

La figure I.12 suivante représente une organisation structurelle de l'annuaire UDDI.

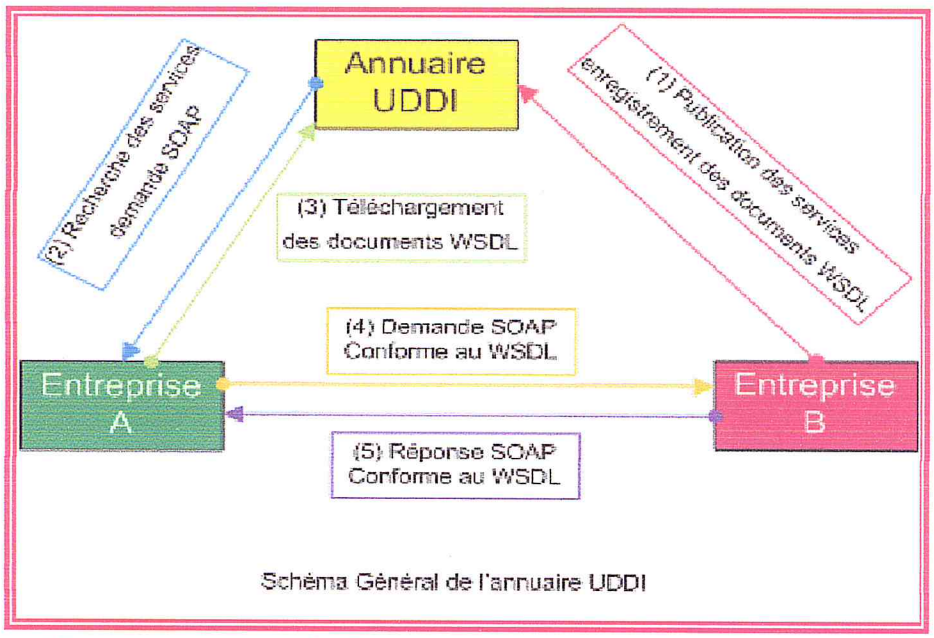


Figure 1.12 « Organisation structurelle de l'UDDI » **[Chab 2007].**

I.5. Conclusion

De plus en plus, avec l'essor d'Internet, le développement tend vers les technologies du Web. Les Services Web sont des composants logiciels représentant une fonction applicative, ils représentent un mécanisme de communication entre applications distantes à travers le Web. Les services Web permettent le changement de la nature du Web, du document utilisé par les organisations pour la publication des informations, au Web orienté service, qui permet aux serveurs d'applications la communication entre eux.

Les services Web sont suffisamment développés pour que les développeurs les utilisent maintenant dans tous les domaines de l'informatique, afin de récolter les divers bénéfices de la technologie. Ils peuvent être vus comme des ressources actives, dynamiques, relativement à des ressources statiques, celles contenues dans les bases de données disponibles sur le Web.

Dans ce chapitre nous avons construit un état d'art portant sur les services web ainsi que sue les standards et les différentes technologies utilisées par ces derniers. Parmi ces standards on trouve le langage de description des services web WSDL et le protocole de transport SOAP. Ces deux standards sont à l'heure actuelle indéfectiblement liés et représentent une norme industrielle solide et prometteuse pour les grandes entreprises.

Dans le prochain chapitre, on abordera le volé conception dans lequel on décrira nos services web pour la gestion des emplois du temps.

Chapitre II

Composition des services web

II.1. Introduction

L'évolution d'internet et la compétitivité entre les entreprises ont été les facteurs de l'explosion des Services Web.

Les Services Web sont des applications accessibles sur internet réalisant chacune une tâche spécifique pour fournir une solution à une tâche complexe, on peut regrouper des Services Web pour n'en former qu'un service web ; on parle alors de composition de Service Web.

II.2. Définition de la composition des Services Web

La composition des Services Web est le processus de construction de nouveaux Services Web à valeur ajoutée, à partir de deux ou plusieurs Services Web déjà présents et publiés sur le Web. Un Service Web est dit composé ou composite lorsque son exécution implique des interactions avec d'autres Services Web, et des changements des messages entre eux afin de faire appel à leurs fonctionnalités. La composition de Services Web spécifie quels services ont besoin d'être invoqués, dans quel ordre et comment gérer les conditions d'interaction. **[Arenaza, 2006]**

Suite à l'étude de différents travaux sur la composition de services Web (**[Alonso et al., 2004]**, **[Benatallah et al., 2005]**, **[Claro et al., 2006]**) nous retenons la définition de **[Benatallah et al., 2005]** qui nous paraît la plus générale. Dans **[Benatallah et al., 2005]**, 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. Ces mêmes auteurs ont défini le cycle de vie d'une composition de services Web reposant à partir de six activités **[Benatallah et al, 2002]** représentées dans la Figure II.1 :

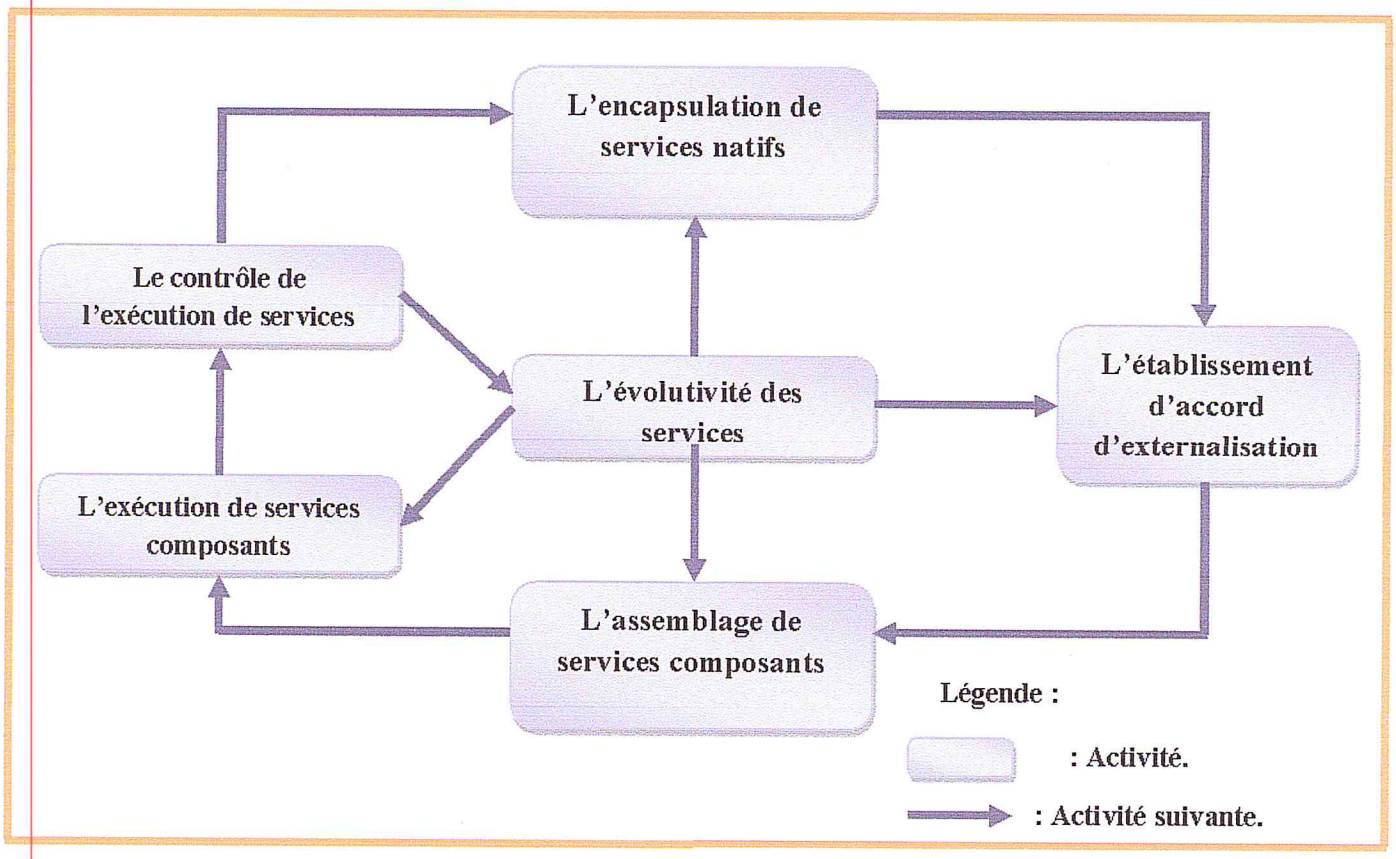


Figure II.1 : « Illustration du cycle de vie de d'une composition de Services Web. » [Benatallah et al, 2002]

- **L'encapsulation de services natifs (*Wrapping services*) :** Cette première activité permet de s'assurer que tout service peut être appelé lors d'une composition, indépendamment de son modèle de données, de son format de message, et de son protocole d'interaction.

- **L'établissement d'accord d'externalisation (*Setting outsourcing agreements*) :** Cette seconde activité consiste à négocier, établir, et appliquer des obligations contractuelles entre les services.

- **L'assemblage de services composants (*Assembling composite services*) :** Cette activité permet de spécifier, à un haut niveau d'abstraction, l'ensemble des services à composer afin d'atteindre l'objectif attendu. Cet assemblage comporte une phase d'identification des services et de spécification de leurs interactions conformément aux descriptions et aux accords entre services.

- **L'exécution de services composants (*Executing services*) :** Cette activité consiste en l'exécution des spécifications de la composition précédemment définies.

- **Le contrôle de l'exécution de services composites (*Monitoring services*) :** La phase de contrôle permet de superviser l'exécution de la composition en vérifiant, par exemple, l'accès aux services, les changements de statut, les échanges de messages. Ce contrôle permet de détecter des violations de contrats, de mesurer les performances des services appelés et de prédire des exceptions.

- **L'évolutivité des services (*Evolving services*) :** Cette dernière phase permet de faire évoluer la composition en modifiant les altérations de l'organisation de services, en utilisant de nouveaux services, ou en prenant en compte les retours de la phase de contrôle.

II.3. Description et fonctionnement

La plupart des travaux portant sur la composition de Services Web reconnaissent deux types de composition : l'***orchestration*** et la ***chorégraphie*** de services.
L'orchestration et la chorégraphie sont des moyens de concevoir la composition et les désignons comme des types de composition. Afin de choisir l'un ou l'autre de ces types de composition (orchestration ou chorégraphie), le concepteur de systèmes doit prendre en compte différents paramètres.

II.3.1. Orchestration

[Barros et al., 2005b] définissent l'orchestration comme un ensemble de processus exécutés dans un ordre prédéfini afin de répondre à un but. Ce type de composition permet de centraliser l'invocation des Services Web composants. Chaque service est décrit en termes d'actions internes. Les contrats entre deux services sont constitués selon le processus à exécuter.

À l'instar de **[Barros et al., 2005b]**, **[Benatallah et al., 2005]** définissent l'orchestration comme un processus exécutable. **[Benatallah et al., 2005]** ajoutent que l'orchestration est un ensemble d'actions à réaliser par l'intermédiaire de Services Web. Un moteur d'exécution, un Service Web jouant le rôle de chef d'orchestre, gère l'enchaînement des Services Web par une logique de contrôle. Pour concevoir une orchestration de Services Web, il faut décrire les interactions entre le moteur d'exécution et les Services Web. Ces interactions correspondent aux appels, effectués par le moteur, d'action(s) proposée(s) par les *Services Web composants*.

D'après **[Peltz, 2003]**, l'orchestration de Services Web consiste en la programmation d'un moteur qui appelle un ensemble de Services Web selon un processus prédéfini. Ce moteur définit le processus dans son ensemble et appelle les Services Web (tant internes qu'externes à l'organisation) selon l'ordre des tâches d'exécution.

La Figure II.2 illustre l'exécution du moteur (lui-même un Service Web – *Service Web Moteur*) permise par l'enchaînement de l'exécution de deux autres Services Web (le *Service Web 1* puis le *Service Web 2*). Cet enchaînement est possible via un opérateur d'ordonnancement (représenté par le losange dans la figure). L'exécution de la composition repose sur l'appel du *Service Web 1*, puis sur l'appel du *Service Web 2*, réalisés tous deux par le *Service Web Moteur*.

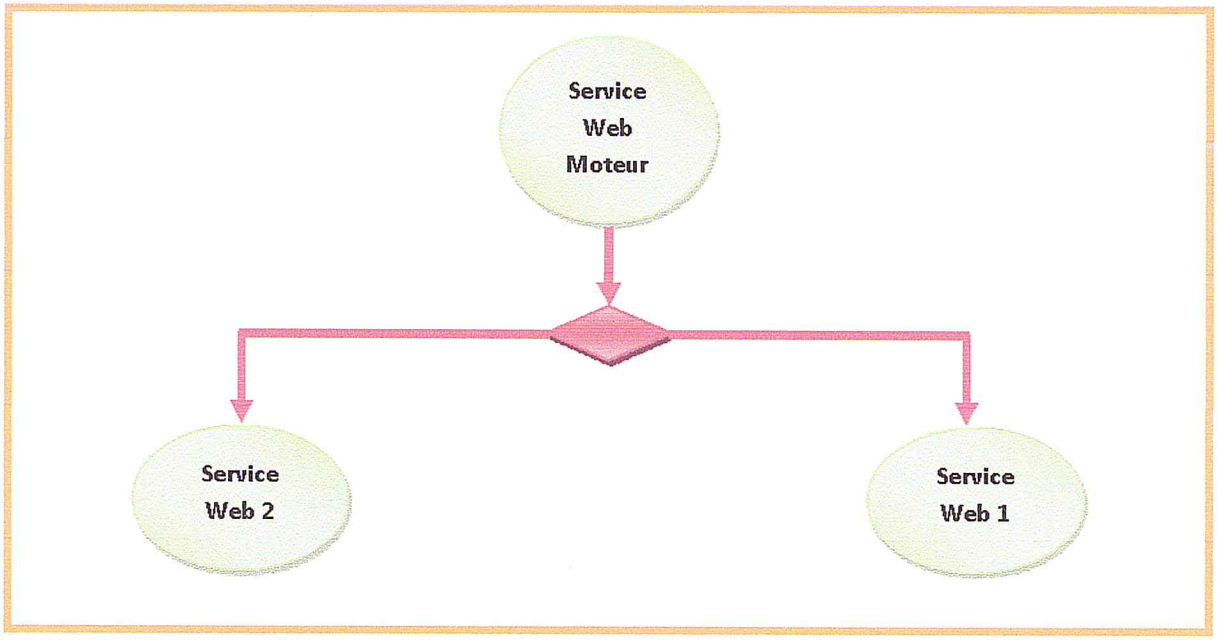


Figure I.2 : « Illustration de l'orchestration. » **d'après [Peltz, 2003].**

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 II.3 illustre un exemple de 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.

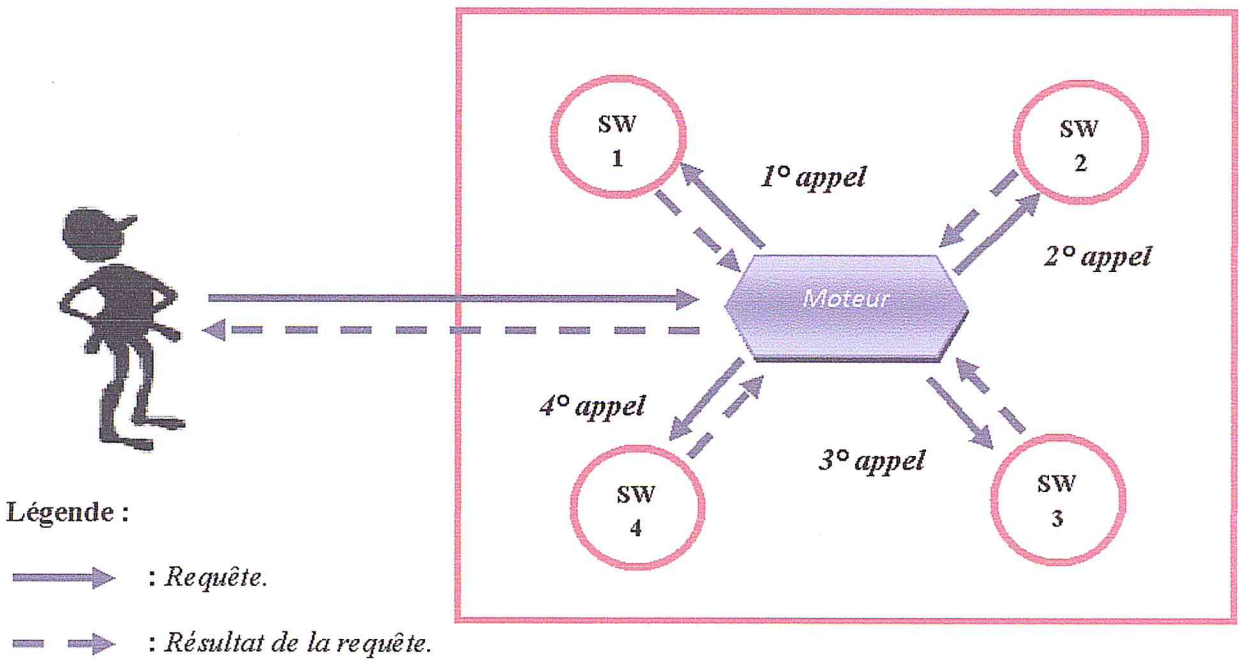


Figure II.3 : «Un exemple d'orchestration. » **[Velasco, 2008]**

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 (le canevas et le script) d écrivent 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.

II.3.2. Chorégraphie

D'après **[Barros et al., 2005b]**, la chorégraphie permet de décrire la composition comme un moyen d'atteindre un but commun en utilisant un ensemble de Services Web. La collaboration entre chaque Service Web de la collection (faisant partie de la composition) est décrite par des flots de contrôle. Le fait que la chorégraphie mette en œuvre un ensemble de Services Web afin d'accomplir un but commun. Pour concevoir une chorégraphie, les interactions entre les différents services doivent être décrites. La logique de contrôle est supervisée par chacun des services intervenant dans la composition. L'exécution du processus est alors distribuée.

D'après **[Peltz, 2003]**, la description de chaque Service Web intervenant dans la chorégraphie inclut la description de sa participation dans le processus. De ce fait, ces services peuvent collaborer à l'aide de messages échangés afin de savoir si tel ou tel service peut aider dans l'exécution de la requête. Chaque Service Web peut communiquer avec un autre Service Web par l'intermédiaire d'échange de messages.

La Figure II.4 représente un protocole d'initiation de collaboration entre deux services dans le cadre d'une chorégraphie. Dans cet exemple, le *Service Web 1* demande l'exécution d'une méthode du *Service Web 2* par l'intermédiaire d'un envoi de message (*Requête de service*). Cette requête est acceptée par le Service Web 2. Ce dernier envoie un message d'acceptation au *Service Web 1* (*Acceptation*). Le Service Web 1 accepte le service proposé par le Service Web 2 en lui envoyant un message (*Service admis*) accordé (*Acceptation*) par ce second service. Une fois ces messages échangés le *Service Web 1* peut invoquer les *Service Web 2* dans le cadre de la composition. **[Peltz, 2003]**

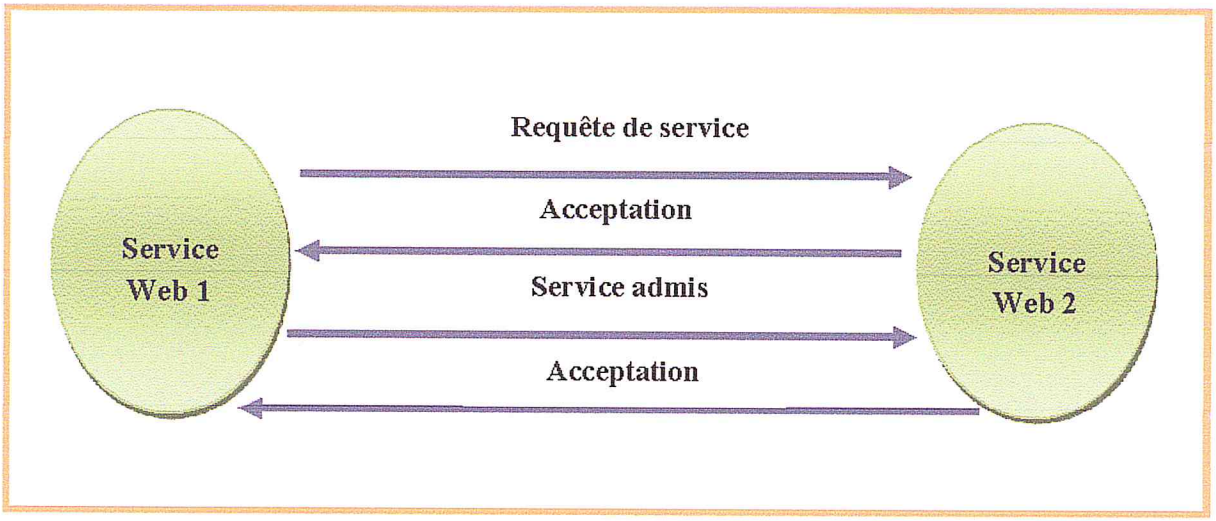
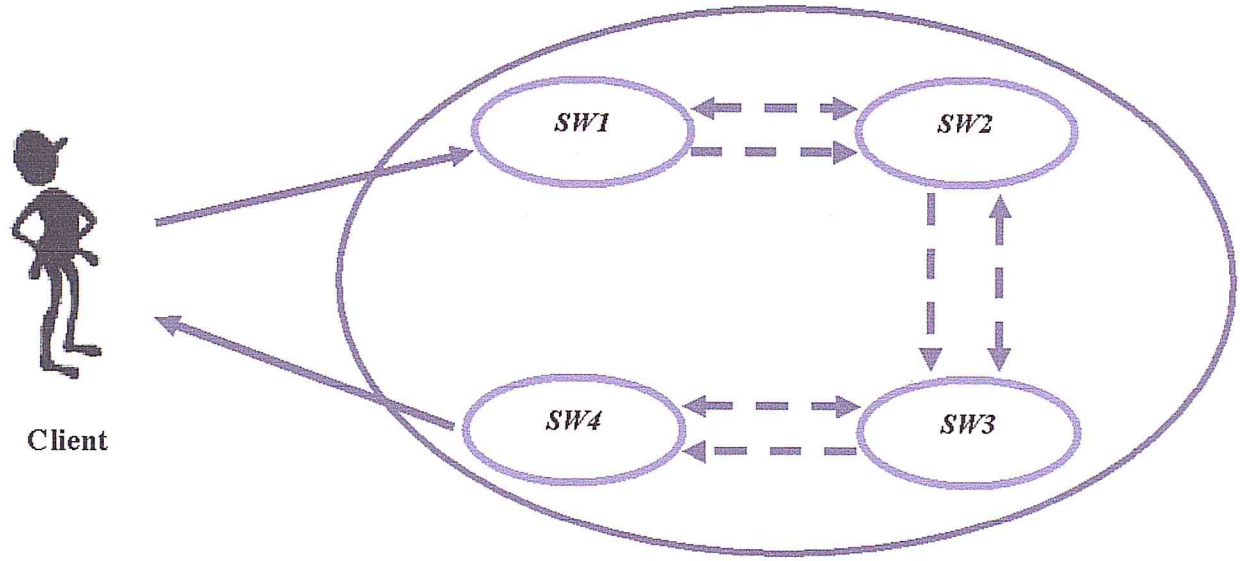


Figure II.4 : « L'illustration de la chorégraphie. » **D'après [Peltz, 2003]**

La chorégraphie est aussi appelée composition dynamique. 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.

Le W3C compte depuis 2002 parmi ses groupes de travail, le groupe de travail sur la chorégraphie de Services Web (*Web Services Choreography Working Group*). Pour ce dernier, la chorégraphie des Services Web concerne les interactions observables des services avec leurs utilisateurs (appelés aussi clients). Ces utilisateurs, automatisés ou non, peuvent être d'autres Services Web, des applications, ou des concepteurs d'applications. Cet ensemble spécifique d'interactions peut être comparé à une collaboration entre un ensemble de Services Web et leurs clients. La description d'une chorégraphie est un contrat multiparti qui décrit, à partir d'un point de vue global, le comportement observable externe entre plusieurs clients (généralement des Services Web). Chaque comportement externe observable est défini comme la présence ou l'absence de messages échangés entre un Service Web et ses clients.

La Figure II.5 permet d'illustrer un exemple 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 (comme illustré par la Figure *.5) 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.



Légende :

: *Requête.*

: *Résultat de la requête.*

: *Envoi du résultat du SW.*

: *Échange de messages.*

Figure II.5 : « Exemple d'une composition de Services Web de type chorégraphie ».

II.3.3. Orchestration Vs Chorégraphie

Il y a une différence importante entre l'orchestration et la chorégraphie de services Web. L'orchestration se base sur un procédé métier exécutable pouvant interagir avec les services Web internes ou externes. Elle offre une vision centralisée, le procédé est toujours contrôlé du point de vueP d'un des partenaires métier. La chorégraphie est de nature plus collaborative, chaque participant impliqué dans le procédé décrit le rôle qu'il joue dans l'interaction. Beaucoup de standards se sont intéressés initialement soit à l'orchestration soit à la chorégraphie. Mais les standards récents ont fusionné ces deux termes en un seul. Dans une chorégraphie, la logique qui contrôle les interactions entre les services composants est distribuée entre les participants. Il n'y a pas de coordinateur central. Chaque service impliqué dans la chorégraphie connaît exactement avec qui interagir et quand exécuter les opérations dont il est responsable. La composition par chorégraphie, aussi appelée collaboration dans la littérature, décrit d'une part les interactions entre les services et d'autre part les relations qui existent entre ces interactions. Les opérations internes des participants ne sont pas considérées. **[Helga, 2007]**

II.4. Classification de la composition de Services Web

L'idée de la composition de Services Web consiste à définir comment les Services Web vont être rassemblés selon certaines règles, pour atteindre le but demandé par un utilisateur. Une fois que, la description de la composition est réalisée, il est possible de savoir facilement quels Services Web appartiendront à cette composition. La composition peut être classifiée en 3 catégories :

- **La composition manuelle**

La composition manuelle des Services Web suppose que l'utilisateur gère la composition à la main via un éditeur de texte et sans l'aide d'outils dédiés.

- **La composition Semi-automatique**

Les techniques de composition semi-automatiques sont un pas en avant en comparaison avec la composition manuelle, dans le sens qu'elles font des suggestions sémantiques pour aider à la sélection des Services Web dans le processus de composition.

- **La composition automatique**

La composition totalement automatisée prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise.

Selon la sélection des services Web et que la gestion du flot soit faite ou non à priori, une autre approche sera dite statique ou dynamique :

- **La composition statique des services web :**

Dans cette approche, les services web à composer sont choisis à l'heure de faire l'architecture et le design. Les composants sont choisis et reliés ensemble, avant d'être compilés et déployés **[Dustdar et Schreiner, 2005].**Ceci peut marcher en autant que l'environnement des services web, les partenaires d'affaires, ainsi que les dits composants changent peu ou pas du tout. Microsoft Biztalk et Bea Weblogic sont deux exemples de moteurs de composition statique de services web **[Sun et al., 2003].** Si les fournisseurs de services proposent d'autres services ou changent les anciens services, des incohérences peuvent être causées, ce qui demanderait un changement de l'architecture du logiciel, voire de la définition du processus et créerait l'obligation de faire une nouvelle conception du système. Dans ce cas, la composition statique des services web est considérée trop restrictive : les composants doivent s'adapter automatiquement aux changements imprévisibles **[Sun et al., 2003].**

- **La composition dynamique des services web :**

Dans cette approche, les services sont sélectionnés et composés à la volée en fonction des besoins formulés par l'utilisateur **[Osman et al., 2005].**Cette approche offre le potentiel de réaliser des applications flexibles et adaptables, en sélectionnant et en combinant les services de manière appropriée sur la base de la requête et du contexte de l'utilisateur. Ce type de composition peut engendrer de nombreuses applications utiles, qui n'ont pas été prévues à l'étape de conception. Par conséquent, la composition dynamique de services Web est propice dans un environnement, tel que

le Web et l'informatique pervasive où les composants disponibles sont dynamiques et les attentes des utilisateurs variables et personnalisées.

II.5 Langages de Composition des Services Web

Plusieurs langages ont été proposés dans la perspective d'améliorer le travail de conception et de mise en œuvre des processus métiers, en particulier des processus de type B2B ainsi que toutes ses déclinaisons **[Helga, 2007]**. L'éventail de ces spécifications va des outils pour la conception graphique des processus jusqu'à la conception de plates-formes pour la composition. Actuellement il existe différents langages permettant de réaliser l'orchestration ou la chorégraphie, nous citons les plus importants BPEL4WS, WSCDL, BPML, et WSCI. **[Helga, 2007]**

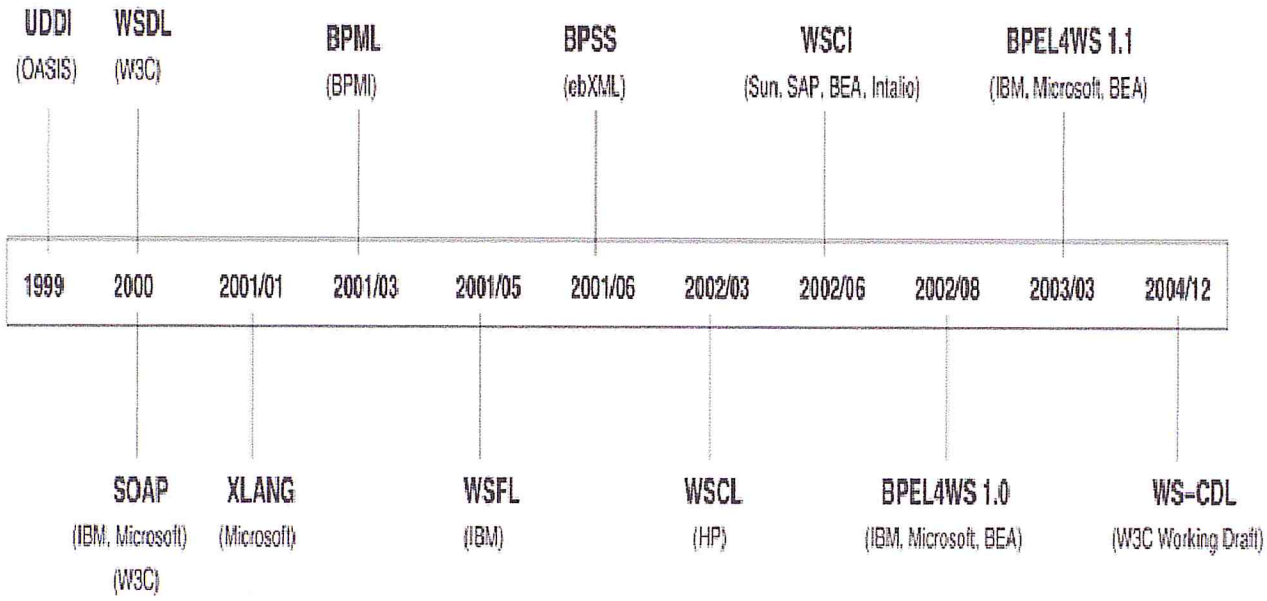


Figure II.6 : « Langages de composition de services web ». **[Helga, 2007]**

La Figure II.6 montre la chronologie d'apparition des différents langages pour décrire des orchestrations et des chorégraphies. UDDI, SOAP et WSDL sont dans la figure à titre de référence. Certains de ces langages permettent de concevoir la composition de services web centrée sur le paradigme de l'orchestration et d'autres, sur celui de la chorégraphie.

II.5.1. XLANG

Créé par Microsoft, le XLANG est une extension de WSDL. Elle fournit en même temps un modèle pour une orchestration des services et des contrats de collaboration entre celles-ci. Les

actions sont les constituants de base d'une définition de processus de XLANG. Les quatre types d'opérations WSDL (requête/réponse, sollicitation de la réponse, le sens unique, et la notification) peuvent être employés comme actions de XLANG. XLANG ajoute deux autres genres d'action : arrêts (date-limite et durée) et exceptions.

II.5.2. BPML (*Business Process Modeling Language)*

Le BPML est un langage de modélisation des processus métiers. Il permet de définir un modèle abstrait d'interaction entre collaborateurs participant à une activité de l'entreprise, voire entre une organisation et ses partenaires. Les processus métier sont représentés par un flux de données et un flux d'événements sur lesquels on peut influer en définissant des règles métier, des règles de sécurité, des règles de transactions. On peut ensuite lancer l'exécution du modèle et vérifier le fonctionnement théorique des différents processus.

II.5.3. WSFL *(Web Services Flow Language)*

WSFL est un langage base sur XML pour la description des Services Web composite. WSFL considère deux types de compositions de Services Web :

- Le premier type indique le modèle approprié d'utilisation d'une collection de Service Web, de telle manière que la composition résultante décrive comment réaliser le but particulier d'un business, typiquement, le résultat est une description d'un processus métier.
- Le deuxième type indique le modèle d'interaction d'une collection de Services Web, dans ce cas, le résultat est une description de l'interaction globale associée.

II.5.4. WSCL (*Web Services Conversation Language*)

WSCL propose de décrire à l'aide de document XML, les Services Web en mettant l'accent sur les conversations de ceux-ci. En outre, les messages à échanger sont pris en compte. WSCL a été pensé pour s'employer conjointement avec WSDL. Les définitions WSDL peuvent être manipulées par WSCL pour décrire les opérations possibles ainsi que leur chorégraphie. En retour, WSDL fournit les concrétisations vers des définitions de messages et des détails techniques pour les éléments manipulés par WSCL.

II.5.5. WSCI (*Web Services Choreography Interface*).

WSCI est un langage reposant sur XML. Il propose de se focaliser sur la représentation des Services Web en tant qu'interfaces décrivant le flux de messages échangés (la chorégraphie de messages). Il propose ainsi de décrire le comportement d'un Service Web, et plus précisément la manière dont ce dernier se doit de réagir au regard des messages qu'il réception en provenance d'autre Service Web. Des réactions qu'il exprime en termes de tâches à effectuer, de règles de séquencement et de corrélation de ces tâches…ou encore de traitement d'exceptions.

On remarque que WSDL et ses définitions abstraites sont réutilisées afin de pouvoir également décrire par la suite les modalités de concrétisation des éléments manipules pour modéliser un service.

II.5.6. BPEL4WS (*Business Process Execution Language for Web Services*)

BPEL est un langage pour les processus métier base sur XML. Il a été conçu pour permettre de charger/partager les données distribuées, même à travers des organismes multiples, en employant une combinaison de Service Web.

BPEL décrit l'interaction des processus métier, à la fois au sein des entreprises et entre elles. Les entreprises utilisatrices du langage BPEL pourront ainsi définir leurs processus métiers et en garantir l'interopérabilité non seulement à l'échelle de l'entreprise, mais également avec leurs partenaires commerciaux, au sein d'un environnement de Service Web. BPEL rend possible l'interopérabilité entre des activités commerciales basées sur des technologies différentes.

Ecrit par des créateurs des systèmes de BEA, IBM, et Microsoft, la spécification BPEL combine et remplace le WSFL d'IBM et le XLANG de Microsoft. Parfois, BPEL est appelé comme BPELWS ou BPEL4WS.

II.5.7. WS-CDL (*Web Services Choreography Description Language*)

Spécification du consortium W3C. WS-CDL est un langage de description de chorégraphie. C'est une initiative du consortium W3C qui vise à coordonner les interactions entre les services web et leurs utilisateurs ; il définit une chorégraphie comme un contrat multipartenaires qui décrit, d'un point de vue global, le comportement commun observable des services participants à une collaboration. Une description de chorégraphie en WS-CDL est un conteneur d'un ensemble d'activités qui peuvent être effectuées par un ou plusieurs participants. WS-CDL distingue trois classes d'activités : « *unitaire* », « *flot de contrôle* », et « *basiques* ».

Une activité « *unitaire* » décrit une exécution conditionnelle et éventuellement répétitive. La classe « *flot de contrôle* » inclut trois types d'activités, « séquence », « parallèle » et « choix », qui permet de spécifier les constructeurs basiques de flot de contrôle. La troisième classe « *basique* » inclut les activités « NoAction », « SilentAction », « Assign », « Perform » et « Interaction ». Les activités « NoAction » et « SilentAction » décrivent des points dans une chorégraphie ou un rôle n'effectue aucune action ou exécute une action cachée qui n'influe pas sur Le reste de la chorégraphie. L'activité « Assign » est pour affecter la valeur d'une variable à une autre. L'activité « Perform » est utilisée pour réaliser une autre chorégraphie dans le contexte de la chorégraphie courante.

Un élément important de WS-CDL est l'activité « Interaction ». Cette activité décrit un échange d'information entre les parties. Trois types d'interactions sont définis : envoyer une requête, envoyer une réponse, ou envoyer une requête qui nécessite une réponse. Une description d'une activité «Interaction » comprend trois parties correspondantes aux :

(1) participants impliqués.
(2) information échangées.
(3) au canal (Channel) pour l'échange des informations.

II.6. Etude comparative

Après des travaux menés par **[Antonio, 2006]** sur la comparaison entre différents langages de composition. Cinq langages, à savoir BPEL4WS, BMPL, WSCI, WSCDL et DAML-S, ont été choisis et comparés à ceux de huit exigences qu'un langage de composition devrait soutenir pour faciliter le processus de la composition de l'entreprise basés sur le Web.

Nous citons la comparaison entre BPEL4WS, BPML, WSCI, et WS-CDL selon les exigences suivantes :

- **Modélisation de la collaboration :** La possibilité d'effectuer à long terme, la collaboration peer-to peer entre les services participants. La collaboration doit être modélisée en termes d'interactions d'échanges de messagerie.

- **Modélisation du contrôle d'exécution :** La capacité à assembler et intégrer des Web services dans le cadre de l'exécution des processus métier est vitale pour toute Services Web dans le langage de composition.

- **Représentation de Rôle :** Les parties impliquées dans les processus d'affaires jouent différents rôles dans différentes étapes du processus. La représentation des rôles est nécessaire dans la responsabilité et le comportement que les parties sont supposées dans divers scénarios.

- **Les transactions et compensations** : Les processus d'affaires sont généralement de longue durée processus qui peut prendre des heures, voire des semaines, et donc la capacité à gérer les transactions et compensations sur les services invocation est critique pour Composition de services Web. Compensations sont nécessaires pour rollback les effets de transactions terminée quand il y'a une panne dans le cadre de l'opération ci-joint.

- **La gestion des exceptions :** La composition des services Web utilise des Services Web externes qui sont purement sous le contrôle du propriétaire de services Web. Ils doivent prendre en compte la gestion des exceptions dans le processus d'invocation quand services Web externes ne réagissent pas.
- **Prise en charge de l'accord d'entreprise :** Il est important dans un scénario business to-business de définir un accord entre les parties concernées. Accord d'entreprise définit le contrat entre deux ou plusieurs parties sur la qualité des services (QoS). Il est nécessaire de représenter la qualité de service requise dans les services Web composés.

- **Support logiciel du fournisseur :** Si le langage a un support du logiciel.

La comparaison est listée dans le tableau suivant :

	BPEL4WS	BMPL	WS-CDL	WSCI
Modélisation de la collaboration	Appui Solide	Appui indirect	Appui Solide	Appui Solide
Modélisation du contrôle d'exécution	Appui Solide	Appui Solide	Non	Non
Représentation de Rôle	Appui faible	Non	Appui Solide	Appui Solide
Les transactions et compensations	Appui indirect	Appui Solide	Appui indirect	Appui Solide
La gestion des exceptions	Appui Solide	Appui	Appui	Appui Solide
Prise en charge de l'accord d'entreprise	Non	Non	Non	Non
Support logiciel du fournisseur	Beaucoup	Peu	Non	Peu

Tableau II.1 : « Comparaison entre BPEL4WS, BMPL, WS-CDL, WSCI ». [Antonio, 2006]

Dans le tableau de comparaison (Tableau II.1), la valeur "indirect" signifie que les exigences ne sont pas directement prises en charge par le langage. Par exemple, dans BPEL4WS les transactions sont réalisées grâce à des défauts gestionnaires et compensations. En résumé tous ces langages fournissent les mécanismes de modélisation des collaborations. Sauf que BPML besoin d'un soutien indirect. BPML facilite la modélisation de l'exécution des processus, et BPEL4WS tente de couvrir les deux aspects. Enfin WS-CDL est plus naturel pour la modélisation des collaborations B2B.

II.7. Synthèse

Dans cette section, nous avons présenté un ensemble de standards pour la composition de services web. Alors que BPEL se concentre sur la création de procédés métiers exécutables, WSCI essaye de traiter le problème d'échange de messages entre services web. WSCI est plutôt une approche collaborative (chorégraphie), ou chaque participant doit définir une interface WSCI pour l'échange de messages. BPEL se focalise plus sur la logique interne et décrit le procédé exécutable de point de vue d'un seul partenaire. Tous les langages (Tableau II.1) prennent en charge la partie impérative de la composition des services, à savoir la gestion des exceptions et des compensations

et possèdent tous la capacité de composer des structures et des activités plus complexes. Les collaborations exigent la mise en place et l'application d'un accord d'entreprises sur la qualité de service. Quant aux outils, BPEL4WS a gagné un large soutien de l'industrie. La plupart des grands éditeurs de logiciels ont promis un soutien pour BPEL4WS dans leurs produits.

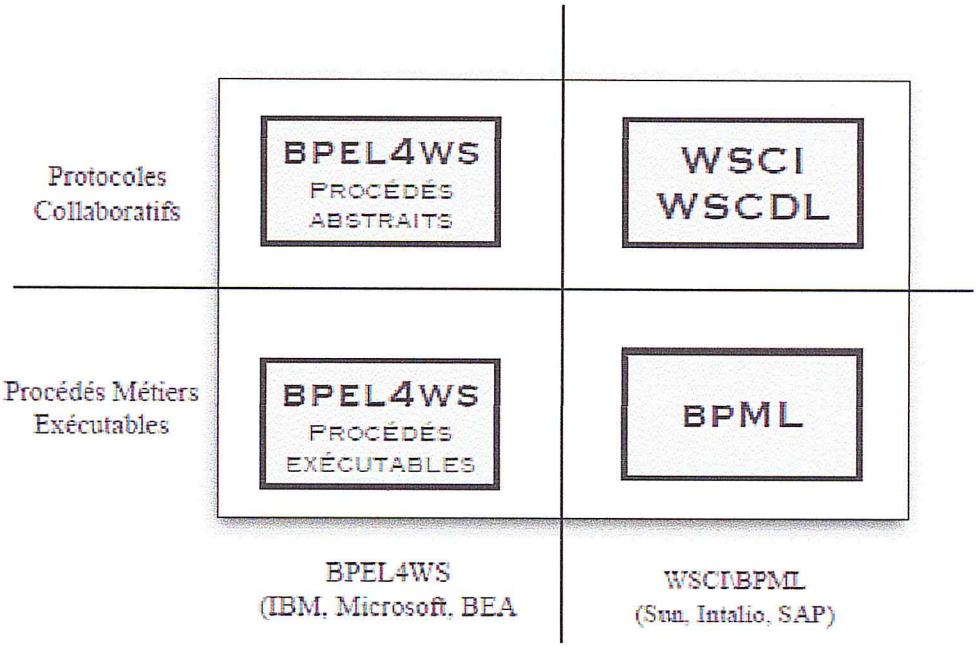


Figure II.7 :« Synthèse sur les standards. » **[Peltz, 2003]**

La figure II.7 présente une synthèse sur les standards cités, par rapport à leurs supports des procédés collaboratifs ou exécutables. WSCI et WS-SDL sont plus de nature collaboratifs, alors que BPML est plutôt centralisé. BPEL peut supporter les deux aspects. Dans notre travail, on a choisis BPEL4SW, comme langage de composition de nos services web. Ce langage permet aux programmeurs de mettre en œuvre un ensemble complexe de services distribués et des procédés pour les composer de manière générale. Il permet d'invoquer en mode synchrone ou asynchrone et d'interagir entre les Services Web.

Le grand avantage de BPEL4WS est la possibilité d'écrire les interactions entre les logiques métiers des différentes entreprises à travers le Service Web.

II.8 BPEL

BPEL définit un modèle de coordination et d'orchestration, il est né au moment de l'avènement des services web, considérés comme la seule réponse possible aux exigences d'intégration, et c'est pourquoi BPEL est entièrement conçu pour interagir avec des Services Web. Les programmes BPEL sont d'ailleurs eux-mêmes des WS dont l'interface est décrite simplement par un fichier WSDL classique. BPEL a été conçu spécifiquement comme un langage pour la définition des processus métier. Il est plus soutenu industriellement et le mieux accepté par les développeurs. En plus, il existe plusieurs outils de développement qui facilite la mise en œuvre de la composition des Services Web.**[Velasco, 2008]**

II.8.1. Définition

BPEL (Business Process Execution Language) est un langage de description de processus normalisé par OASIS (*Organization for the Advancemnent of Structured Informations Standards*). Il permet aux entreprises de normaliser la définition ainsi que l'exécution des processus métier. D'un point de vue technique, BPEL est un langage basé sur XML qui définit :

- L'invocation en mode synchrone ou asynchrone de service web.
- Des enchaînements de traitements.
- La manipulation des structures de données XML.
- La gestion des exceptions et des événements.
- Des branchements conditionnels et des boucles d'exécution.
- Des séquences parallèles.
- Des Timers.

BPEL repose sur les normes XML, SOAP et WSDL. Ce langage distingue les processus abstraits des processus exécutables **[Hubert, 2003]** :

- **Le processus abstrait :** Ce processus spécifie les messages échangés entre les différentes parties (*Services Web composants*) sans indiquer le comportement de chacune d'elles. Il s'agit 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 (*Figure II.8)*.
- **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. **[Peltz, 2003]**

La Figure II.8 illustre la mise en œuvre des deux types de processus (exécutable et abstrait) par BPEL4WS. La définition du processus métier est donnée par le processus exécutable. Les processus abstraits gèrent les invocations entre les différents Services Web (*WS*) permettant l'exécution de la composition de services définie dans le processus exécutable. Trois éléments permettent à BPEL4WS 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*) (Figure II.8).

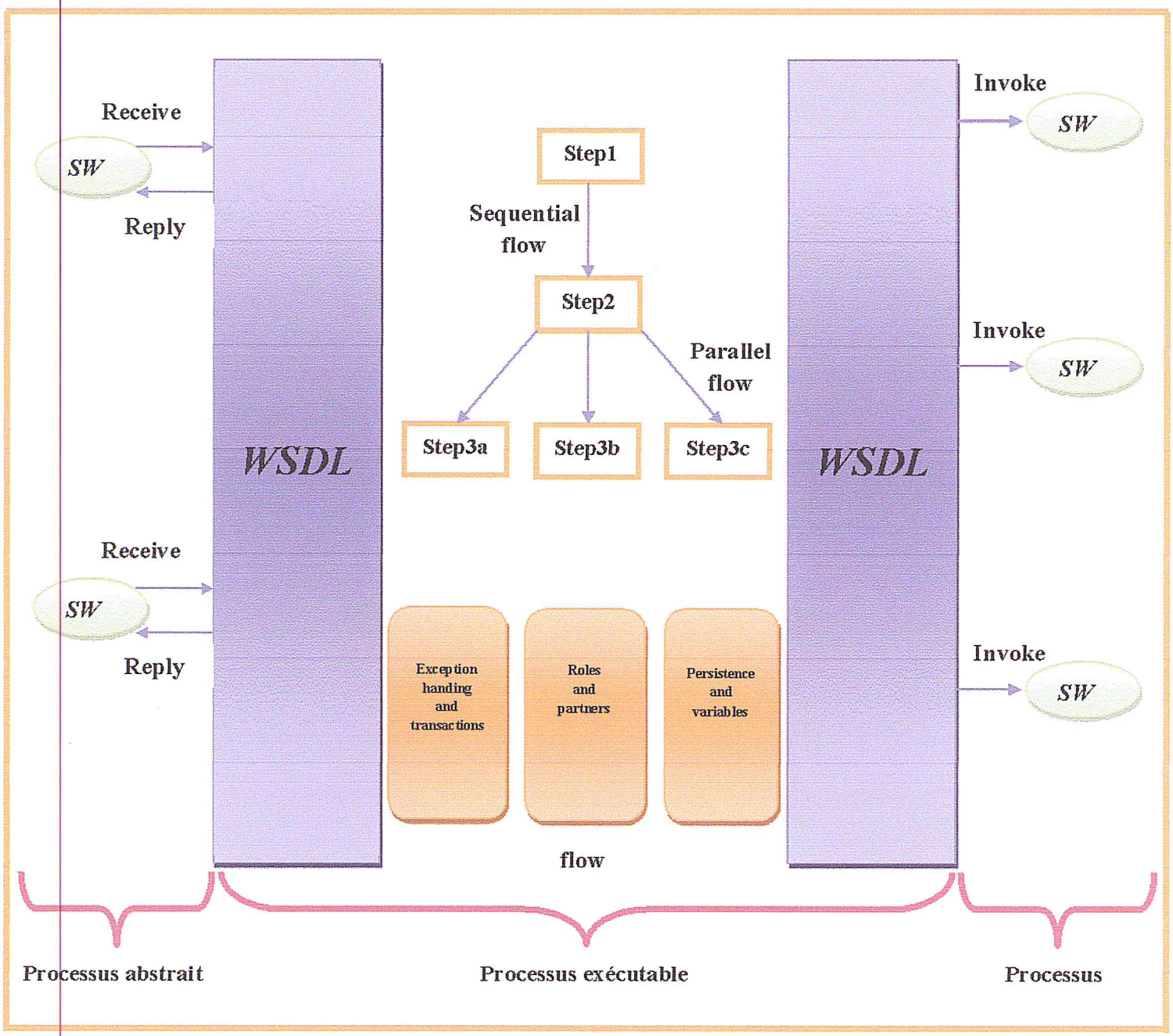


Figure II.8 : « Le flot de processus avec BPEL4WS. »

Les transactions : Les transactions sont utilisées dans BPEL4WS pour gérer les erreurs et les appels d'autres services si le service appelé est indisponible ou défaillant.

Les partenaires : 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. Dans la Figure II.8, présentant un exemple de la définition d'un processus exécutable en BPEL4WS, Les deux partenaires sont

définies (des lignes 2 à 11). Le fait de décrire deux niveaux de rôle permet à chaque partenaire d'avoir une vie indépendante des compositions dans lesquelles il intervient.

Les espaces de stockage : Les espaces de stockage permettent la transmission des données. Le flot de processus BPEL4WS 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.

À partir de la Figure II.9, On peut mettre en relief la syntaxe de BPEL4WS en décrivant un exemple de processus myMeteo (élément process, élément racine du document BPEL4WS, ligne 1). Les différents partenaires (geoIPService et globalWeather) sont en premier lieu définis (élément partners, lignes 2 à 11). Ces deux acteurs interagissent avec le service qui orchestre la composition par l'intermédiaire de deux messages nommés getLocalisation et getWeather (élément containers, lignes 12 à 15). L'élément containers référence le type de message (attribut messageType, lignes 13 et 14) contenus dans les descriptions WSDL des *Services Web composants*. Ainsi, le Service Web qui orchestre connaît les types de données manipulées. Le flot de processus (élément flow, lignes 16 à 25) se compose tout d'abord de l'appel du partenaire geoIPService (élément invoke, lignes 17 à 20) puis de l'appel du partenaire globalWeather (élément invoke, lignes 21 à 24). L'élément invoke comprend les accès aux ports (attribut portType, lignes 18 et 22) et les opérations (attribut opération, lignes 19 et 23) référençant la description WSDL des *Services Web composants*. La Figure II.9 n'illustre qu'un extrait de la description du processus exécutable *myMeteo*. BPEL4WS intègre aussi les flots de données dans la description du processus en reliant les sorties d'un service avec les entrées du service lui succédant.

```
<process name="myMeteo">
    <partners>
        <partner name="geoIPService"
                 serviceLinkType="geoIPLink"
                 myRole="geoIP"
                 partneRole="geoIPContext"/>
        <partner name="globalWeather"
                 serviceLinkType="globalWeatherLink"
                 myRole="weather"
                 partnerRole="globalWeatherService"/>
    </partners>
    <containers>
        <container name="getLocalisation" messageType="getGeoIPContextSoapIn"/>
        <container name="getWeatherContainer" messageType="getWeatherSoapIn"/>
    </containers>
    <flow>
        <invoke partner="geoIPService"
                portType="GeoIPServiceSoap"
                operation="GetGeoIPContext"
                inputContainer="getLocalisation" />
        <invoke partner="globalWeather"
                portType="GlobalWeatherSoap"
                operation="GetWeather"
                inputContainer="getWeatherContainer" />
    </flow>
</process>
```

***Figure II.9 : « Définition du processus exécutable de la demande du Service de météorologie myMeteo à l'aide de BPEL4WS. »*[Velasco, 2008]**

II.8.2. Fonctionnalités de BPEL

Avec BPEL, on peut définir des processus métiers aussi bien simples que complexes. Jusqu'à un certain degré, BPEL est similaire aux langages de programmation traditionnels. Il offre des constructions comme des boucles, des branchements, des variables, des assignements, etc. qui nous permettent de définir des processus métiers de manière algorithmique, par conséquent cette définition est relativement simplifiée, d'autre part il est moins compliqué que les langages de programmation traditionnels **[Hubert, 2003]**.

Les constructions les plus importantes sont liées aux invocations des web services. BPEL facilite l'invocation des opérations des web services que ce soit des opérations synchrones ou asynchrones [01]. On peut invoquer les opérations de manière séquentielle ou parallèles, comme on peut attendre les callbacks. BPEL fournit un vocabulaire riche pour le traitement des erreurs (*fault handling*). Ci-dessous sont décrites les fonctionnalités les plus importantes qu'offre BPEL **[Hubert, 2003]** :

- Décrire la logique du processus métier à travers la composition de services.
- Composer des processus métiers complexes à partir de processus et de services plus simples.
- Manipuler des invocations synchrones et asynchrones des opérations des services, et gérer les callbacks qui viendront après.
- Invoquer les opérations des services en séquence ou en parallèle.
- Maintenir des activités longues qui sont interruptibles.
- Reprendre des activités interrompues ou qui ont échoué pour minimiser le travail à refaire.

- Router les messages entrants à l'activité ou au processus destinataire.
- Organiser les activités en se basant sur leur temps d'exécution et définir leur ordre d'exécution.
- Exécuter les activités en parallèles.

BPEL4WS 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 du processus exécutable. L'inconvénient principal de BPEL4WS est que la définition du processus est rigide. Si une activité du processus échoue, le processus dans son intégralité échoue. Aucun retour en arrière et aucune alternative au processus ne sont possibles. De même, lors de la description du processus exécutable, la définition des flots de données ne permet pas de connecter des services dont les entrées/sorties ne correspondent pas exactement. BPEL4WS ne prévoit pas de mécanisme de transformation de données.

II.9. Conclusion

La Composition de Services Web a pour but d'utiliser les compétences de plusieurs services afin de résoudre un problème qu'aucun ne saurait résoudre individuellement. Le résultat de cette Composition est un enchaînement des Services Web qui permet de définir la façon dont les données produites par les uns sont consommées par les autres.

Dans ce chapitre, Nous avons défini les types de composition des Services Web et les différents langages utilisés comme BPEL. BPEL est un langage basé sur XML pour la définition des processus métiers. Chaque processus permet de définir des partenaires (servant de canaux de communication avec les Services Web et les clients) ainsi que des variables, et expose des activités simples et complexes. Bien que la mise en œuvre de la composition avec BPEL soit relativement simple, elle présente un inconvenant majeur concernant le mode d'invocation des Services Web en BPEL qui est figé et s'appuie uniquement sur une déclaration en XML : utilisation de l'interface décrite par le WSDL (portType) avec lesquels le processus interagi.

Chapitre III

Optimisation combinatoire : Algorithmes Génétiques pour la génération des emplois du temps

- **Définition**

Un POC peut être décrit en général sous la forme suivante :

$$\min_{x \in X} F(x)$$

Où F désigne une fonction à valeurs réelles (fonction objectif) évaluant les solutions désignées par x. Il s'agit de minimiser la fonction F sur l'ensemble de toutes les solutions réalisables noté X. Les solutions $x \in X$ peuvent être des objets mathématiques de nature très diverse ; souvent x représentera un vecteur d'un espace à n dimensions et F désignera une fonction de cet espace dans IR. L'ensemble des solutions réalisables X est généralement déterminé par des contraintes souvent exprimées comme des inégalités.

Remarque : Suivant la nature de l'ensemble X et de la fonction F, le POC peut être facile (résoluble en temps polynomial) ou difficile. Par exemple, si X est un polyèdre convexe $\subseteq$IR et F une fonction linéaire, on retrouve un PL résoluble efficacement par le simplexe ou même par d'autres algorithmes polynomiaux. Par contre si $X \subseteq Z$, on retrouve des programmes en nombre entiers qui sont des problèmes difficiles ; aucun algorithme polynomial n'a été trouvé à l'heure actuelle pour les résoudre.

III.2.2. Problème d'affectation sous contraintes

La définition générale de l'optimisation ne précise ni la forme des solutions de S, ni la façon de générer ces solutions (admissibles ou non admissibles). Nous nous intéressons en particulier à une classe importante de problèmes dont une solution peut être décrite explicitement par une affectation de valeurs à l'ensemble des variables du problème. Etant donné un ensemble fini $V = \{V1,..., Vn\}$ de variables et un ensemble $D = \{D1,..., Dn\}$ de domaines finis associés, une solution potentielle du problème (affectation) consiste à choisir pour chaque variable Vi ($1 \leq i \leq n$) une valeur choisie dans son domaine Di. L'ensemble S des solutions potentielles est donc représenté par le produit cartésien $D1 \times D2 ... \times Dn$ des domaines. On dispose en outre d'un ensemble $C = \{C1,..., Cp\}$ de contraintes, où chaque contrainte Cj ($1 \leq j \leq p$) est une relation sur un sous ensemble V'j de V qui spécifie quelles combinaisons de valeurs sont compatibles pour les variables de V'j. **[BEN 08]**

Les problèmes d'affectation sous contraintes possèdent de très nombreuses applications pratiques concernant l'affectation de ressources, le groupement, la classification, la planification, l'emploi du temps et l'ordonnancement, dans des domaines très variés. Les PASC permettent également de modéliser facilement des problèmes de référence comme par exemple la k-coloration et la satisfiabilité.

III.3. Théorie de la Complexité

Un problème est dit polynomial s'il existe un algorithme permettant de trouver une solution optimale pour toutes ses instances en un temps polynomial par rapport à la taille de l'instance **[Saka, 1984]** Un tel algorithme est dit efficace pour le problème en question. C'est notamment le cas de certains problèmes de plus court chemin dans un graphe valué, du recouvrement d'un graphe valué par un arbre de poids minimum, des problèmes classiques de flots. Cependant, pour la majorité des problèmes d'optimisation combinatoire, aucun algorithme polynomial n'est connu actuellement. La difficulté intrinsèque de ces problèmes est bien caractérisée par la théorie de la NP-complétude **[Saka, 1984]**. De nombreux problèmes d'optimisation combinatoire (la plupart de ceux qui sont vraiment intéressants dans les applications !) ont été prouvés NPdifficiles. Cette difficulté n'est pas seulement théorique et se confirme hélas dans la pratique. Il arrive que des algorithmes exacts de complexité exponentielle se comportent efficacement face à de très grosses instances - pour certains problèmes et certaines classes d'instances. Mais c'est très souvent l'inverse qui se produit ; pour de nombreux problèmes, les meilleures méthodes exactes peuvent être mises en échec par des instances de taille modeste, parfois à partir de quelques dizaines de variables seulement. Par exemple, on ne connaît aucune méthode exacte qui soit capable de colorier de façon optimale un graphe aléatoire de densité 1/2 lorsque le nombre de sommets dépasse 90 **[Goldberg, 1989]**. Pour traiter les grosses instances de ce type de problèmes, on se contente de solutions approchées obtenues avec une méthode heuristique.

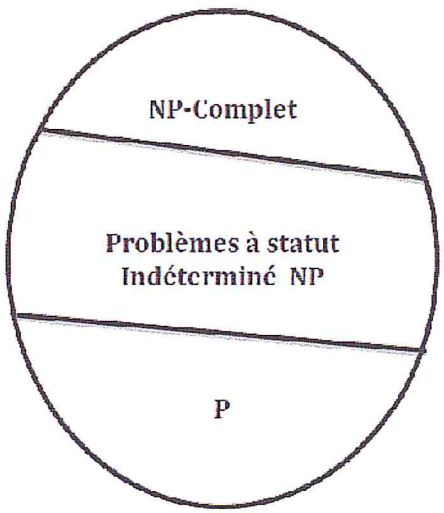


Figure III.1 : « Type de complexité »

La figure III.1 représente les différents types de complexité. Les problèmes NP-complets représentent le noyau dur de NP, car si on trouvait un algorithme polynomial pour un seul problème NP-complet X, on pourrait en déduire un autre pour tout autre problème difficile Y de NP. En effet, ce dernier algorithme consisterait à transformer polynomialement les données pour Y en données pour X, puis à exécuter l'algorithme pour X.

III.4. Méthodes de résolution

Un très grand nombre de méthodes de résolution existent en RO et en IA pour l'optimisation combinatoire et l'affectation sous contraintes. La figure met en parallèle les méthodes représentatives développées en RO et en IA, avec à titre indicatif la date approximative d'apparition de chaque méthode.

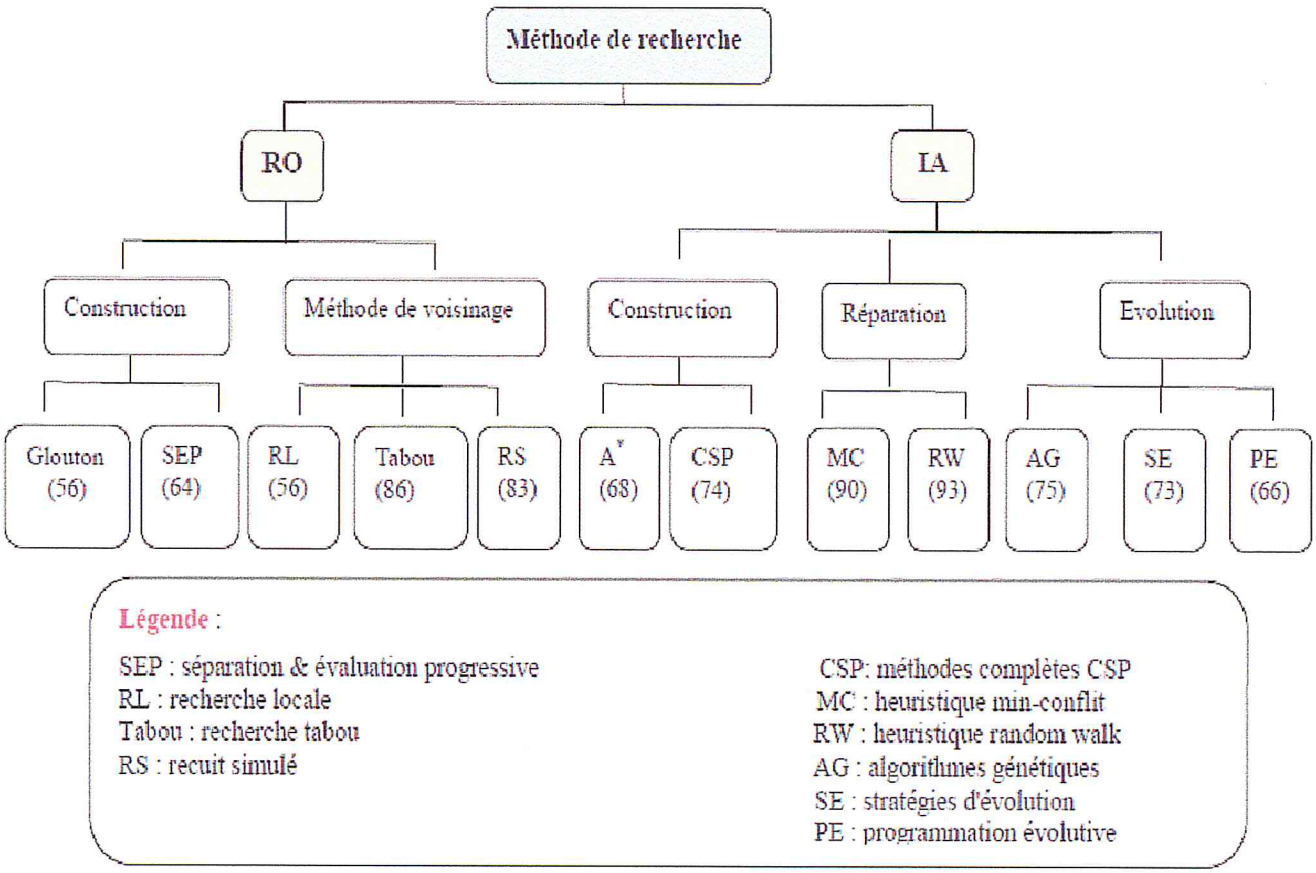


Figure III.2 : « Classement des méthodes de résolution en RO et IA ». **[12]**

D'une manière très générale (Figure III.2), les méthodes de résolution suivent quatre approches différentes pour la recherche d'une solution : l'approche de construction, l'approche de relaxation, l'approche de voisinage et l'approche d'évolution.
Ces méthodes font partie de deux groupes de nature différente **[Goldberg, 1989]**:

- Le premier groupe comprend les méthodes exactes d'arborescence qui garantissent la complétude de la résolution : c'est le cas de SEP, A* et CSP. Le temps de calcul nécessaire d'une telle méthode augmente en général exponentiellement avec la taille du problème à résoudre (dans le pire des cas). Pour améliorer l'efficacité de la recherche, on utilise des techniques variées pour calculer des bornes permettant d'élaguer le plus tôt possible des branches conduisant à un échec. Parmi ces techniques, on peut citer les trois types de relaxations **[Goldberg, 1989]**: la relaxation de base en programmation linéaire, la relaxation lagrangienne, la relaxation agrégée et la décomposition lagrangienne. De plus, on emploie des heuristiques pour guider les choix de variables et de valeurs durant l'exploration de l'arborescence.

- Le second groupe comprend les méthodes approchées dont le but est de trouver une solution de bonne qualité en un temps de calcul raisonnable sans garantir l'optimalité de la solution obtenue. Les méthodes approchées sont fondées principalement sur diverses heuristiques, souvent spécifiques à un type de problème **[Goldberg, 1989]**. Les métaheuristiques constituent une autre partie importante des méthodes approchées et ouvrent des voies très intéressantes en matière de conception de méthodes heuristiques pour l'optimisation combinatoire

III.4.1. Approche de construction

L'approche de construction est probablement la plus ancienne et occupe traditionnellement une place très importante en optimisation combinatoire et en intelligence artificielle **[Goldberg, 1989]**.Une méthode de construction construit pas à pas une solution de la forme s = (<V1, v1><V2, v2>...<Vn,vn>). Partant d'une solution partielle initialement vide s = (), elle cherche à étendre à chaque étape la solution partielle s = (<V1, v1>...<Vi-1, vi-1>) (i n) de l'étape précédente. Pour cela, elle détermine la prochaine variable Vi, choisit une valeur vi dans Di et ajoute <Vi,vi> dans s pour obtenir une nouvelle solution partielle s = (<V1,v1>...<Vi-1,vi-1> <Vi,vi>). Ce processus se répète jusqu'à ce que l'on obtienne une solution complète. Durant la recherche d'une solution, une méthode de construction fait intervenir des heuristiques pour effectuer chacun des deux choix : le choix de la variable suivante et le choix de la valeur pour la variable. Les méthodes de cette classe diffèrent entre elles selon les heuristiques utilisées. En général, les heuristiques portent plus souvent sur le choix de variables que sur le choix de valeurs car les informations disponibles concernant le premier choix semblent souvent plus riches.

La performance de ces méthodes dépend largement de la pertinence des heuristiques employées, i.e., de leur capacité d'exploiter les connaissances du problème **[Goldberg, 1989]**.

Par exemple, une heuristique gloutonne bien connue pour la coloration introduite est la suivante **[Goldberg, 1989]**: pour choisir le noeud suivant à colorier, prendre celui dont les noeuds adjacents sont déjà coloriés avec le plus grand nombre de couleurs différentes et lui assigner la couleur autorisée de plus petit rang possible. Les méthodes gloutonnes sont généralement rapides, mais fournissent le plus souvent des solutions de qualité médiocre **[Goldberg, 1989]**.

Un deuxième type de méthode de construction est représenté par les méthodes avec retour arrière. Une méthode de retour arrière avec une stratégie de recherche en profondeur d'abord consiste à fixer à chaque étape la valeur d'une variable. Aussitôt qu'un échec est détecté, un retour arrière est effectué, i.e., une ou plusieurs instanciations déjà effectuées sont annulées et de nouvelles valeurs recherchées **[Goldberg, 1989]**.Par exemple, un algorithme typique avec retour arrière pour la résolution d'un problème de satisfaction de contraintes cherche à prolonger à chaque étape l'assignation courante de manière consistante. En cas d'échec, un retour arrière est effectué sur la dernière variable instanciée possédant encore des valeurs non essayées. Les méthodes avec retour arrière sont en général complètes et de complexité exponentielle **[Goldberg, 1989]**. Pour réduire le nombre de retour arrière (et le temps de recherche), on utilise des techniques de filtrage afin d'anticiper le plus tôt possible les échecs.

Un troisième type de méthode de construction concerne de nombreux algorithmes basés sur le principe de séparation et évaluation progressive **[Goldberg, 1989]**. Un exemple typique est l'algorithme A* avec une stratégie « meilleur d'abord » pour la recherche d'un plus court chemin dans un graphe valué. Cet algorithme débute avec un nœud initial et prolonge ensuite parmi l'ensemble de chemins partiels développés le chemin dont la longueur est la plus faible, en utilisant une estimation de la longueur restante pour calculer la borne inférieure. Lorsqu'un chemin complet est trouvé, les chemins partiels dont la longueur est supérieure à celle du chemin complet sont éliminés. De manière générale, on utilise des techniques de relaxations pour obtenir des bornes aussi serrées que possible.

III.4.2. Approche séquentielle

Les méthodes séquentielles (de voisinage) sont des algorithmes de recherche itératifs qui explorent l'espace **S** en se déplaçant pas à pas d'une solution à une autre. Deux solution S et S'sont

dites voisines si l'on peut obtenir s'en modifiant S selon une règle bien définit. Nous dirons que le passage de S a S' se fait par mouvement. Le voisinage V(s) d'une solution s appartient à S est l'ensemble des solutions voisines de s.

Voici quelques méthodes séquentielles :

III.4.2.1 La méthode de la recherche locale

La recherche locale, appelée aussi la descente ou l'amélioration itérative, représente une classe de méthodes heuristiques très anciennes. Traditionnellement, la recherche locale constitue une arme redoutable pour attaquer des problèmes réputés très difficiles tels que le voyageur de commerce et la partition de graphes. Contrairement à l'approche de construction, la recherche locale manipule des configurations complètes durant la recherche. Une méthode de recherche locale est un processus itératif fondé sur deux éléments essentiels : un voisinage $N : X \rightarrow 2X$ et une procédure exploitant le voisinage. Plus précisément, elle consiste à :

1- débuter avec une configuration quelconque s de X,
2- choisir un voisin s' de s tel que $f(s') < f(s)$ et remplacer s par s'
3- répéter (2) pour tout voisin s' de s, jusqu'à ce que $f(s') \geq f(s)$.

Cette procédure fait intervenir à chaque itération le choix d'un voisin qui améliore la configuration courante. Plusieurs possibilités peuvent être envisagées pour effectuer ce choix. Il est possible d'énumérer les voisins jusqu'à ce qu'on en découvre un qui améliore strictement (première amélioration). On peut également rechercher le meilleur voisin (meilleure amélioration). Cette dernière solution peut sembler plus coûteuse, mais le voisin découvert sera en général de meilleure qualité. De plus, l'utilisation d'une structure de données appropriée peut souvent permettre de trouver directement ce meilleur voisin. Comme l'espace des solutions X est fini, cette procédure de descente s'arrête toujours, et la dernière configuration trouvée ne possède pas de voisin strictement meilleur qu'elle-même. Autrement dit, la recherche locale retourne toujours un optimum local. L'avantage principal de cette méthode réside dans sa grande simplicité et sa rapidité. Mais les solutions produites sont souvent de qualité médiocre et de coût très supérieur au coût optimal. Pour remédier à ce problème, la solution la plus simple est la *méthode de relance aléatoire* qui consiste à générer une nouvelle configuration de départ de façon aléatoire et à recommencer une descente.

En général, l'efficacité des méthodes de recherche locale simples (descente, ou plus grande descente) est très peu satisfaisante. D'abord, par définition, la recherche s'arrête au premier minimum local rencontré, c'est là leur principal défaut. Pour améliorer les résultats, on peut lancer plusieurs fois l'algorithme en partant d'un jeu de solutions initiales différentes, mais la performance de cette technique décroît rapidement. En revanche, autoriser de temps à autre une certaine dégradation des solutions trouvées, afin de mieux explorer tout l'espace des configurations, a conduit au développement des deux méthodes que nous explorons aux paragraphes suivants, à savoir le recuit simulé et la méthode Tabou.

Le principal avantage de la recherche locale simple est évidemment sa grande simplicité de mise en œuvre : la plupart du temps, elle ne fait que calculer f(s+i)-f(s), où i correspond à un déplacement élémentaire, et si cette expression peut se simplifier algébriquement, alors on pourra évaluer très rapidement cette différence. Il est important de remarquer également l'importance du choix de la fonction de voisinage N : un minimum local pour une certaine structure de voisinage ne l'est pas forcément pour une autre. C'est d'ailleurs ce constat qui est à l'origine de la méthode dite de recherche par voisinage variable, qui repose sur la construction de solutions s parmi plusieurs voisinages Ni, plutôt que dans un seul.

III.4.2.2. Le recuit simulé

La méthode de recuit simulé **(RS)** s'inspire du processus de recuit physique. Les origines du recuit simulé remontent aux expériences réalisées dans les années 50 par Metropolis et al **[Hao et al, 1999]** pour simuler l'évolution d'un tel processus de recuit physique. Metropolis et al ont utilisé une méthode stochastique pour générer une suite d'états successifs du système en partant d'un état initial donné. Tout nouvel état est obtenu en faisant subir un déplacement (une perturbation) aléatoire à un atome quelconque. Soit ΔE la différence d'énergie occasionnée par une telle perturbation. Le nouvel état est accepté si l'énergie du système diminue ($\Delta E \leq 0$), sinon, il est accepté avec une probabilité définie par :

$p(\Delta E, T) = \exp(-\Delta E / (Cb \times T))$ où T est la température du système et Cb une constante physique connue sous le nom de *constante de Boltzmann*.

A chaque étape, l'acceptation ou non d'un nouvel état dont l'énergie est supérieure à celle de l'état courant est déterminée de manière probabiliste : un réel θ est tiré aléatoirement de l'intervalle [0, 1] et il est ensuite comparé avec $p(\Delta E, T)$. Si $\theta \leq p(\Delta E, T)$ alors le nouvel état est accepté pour remplacer l'état courant, sinon l'état courant est maintenu. Après un grand nombre de perturbations, un tel processus fait évoluer le système vers un état d'équilibre thermodynamique selon la *distribution de Boltzmann* qui est définie par la probabilité de se trouver dans un état d'énergie E suivante :

$Pr(E) = c(T) \times \exp(-E/(Cb \times T))$ où $c(T)$ est un facteur de normalisation.

Un tel processus du RS est appelé pour résoudre des problèmes d'optimisation combinatoire. Le RS peut être vu comme une version étendue de la méthode de descente. Le processus du recuit simulé répète une procédure itérative qui cherche des configurations de coût plus faible tout en acceptant de manière contrôlée des configurations qui dégradent la fonction de coût. A chaque nouvelle itération, un voisin $s' \in N(s)$ de la configuration courante s est généré de manière aléatoire. Selon les cas, ce voisin sera soit retenu pour remplacer celle-ci, soit rejeté. Si ce voisin est de performance supérieure ou égale à celle de la configuration courante, *i.e.*, $f(s') \leq f(s)$, il est systématiquement retenu.

Dans le cas contraire, s' est pris avec une probabilité $p(\Delta f, T)$ qui dépend de deux facteurs : d'une part l'importance de la dégradation $\Delta f = f(s') - f(s)$ (les dégradations plus faibles sont plus facilement acceptées), et d'autre part un paramètre de contrôle T, la température (une température élevée correspon à une probabilité plus grande d'accepter des dégradations). La température est contrôlée par une fonction décroissante qui définit un schéma de refroidissement. Les deux paramètres de la méthode définissent la longueur des paliers et la fonction permettant de calculer la suite décroissante des températures. En pratique, l'algorithme s'arrête et retourne la meilleure configuration trouvée lorsqu'aucune configuration voisine n'a été acceptée pendant un certain nombre d'itérations à une température ou lorsque la température atteint la valeur zéro.

III.4.2.3. Les algorithmes d'acceptation avec seuil

Ces algorithmes sont des variantes du recuit simulé, la différence entre eux se situe au niveau de l'acceptation de dégradation à chaque étape. Dans le RS, cette décision est prise selon le critère de Metropolis. Dans un algorithme d'acceptation avec seuil, une telle décision est prise de manière déterministe. A chaque itération k, l'acceptation d'un voisin $s' \in N(s)$ se base uniquement sur une fonction auxiliaire $r(s',s)$ et un seuil Tk. A cet effet, s' est accepté si $r(s',s) < Tk$. La fonction $r(s',s)$ et le seuil Tk peuvent être définis de nombreuses manières. Ainsi, on peut définir la fonction $r(s',s)$

par $\Delta f = f(s') - f(s)$, et le paramètre de seuil Tk de manière analogue à la température T du RS. En effet, le seuil Tk est initialisé à une valeur élevée puis décroît progressivement à chaque fois qu'un nombre prédéterminé d'itérations est effectué. Les seuils ainsi générés correspondent à une suite de valeurs positives décroissantes $T1 \geq T2 \geq ... \geq Tk-1 \geq Tk \geq 0$ et donc Tk tends vers 0. L'idée est de diminuer petit à petit la chance d'accepter des configurations qui dégradent la fonction de coût. Quand Tk tend vers 0, l'algorithme réalise une recherche de descente aléatoire. Ces algorithmes ont été utilisés pour résoudre des problèmes d'optimisation et ont obtenu des résultats intéressants. La difficulté essentielle de cette approche se situe au niveau de la détermination des seuils pour une application donnée.

III.4.2.4. La méthode Tabou

La méthode Tabou a été développée par Glover et indépendamment par Hansen **[Ben 2008].** Cette méthode fait appel à un ensemble de règles et de mécanismes généraux pour guider la recherche de manière intelligente à travers l'espace des solutions. Contrairement au recuit simulé qui génère aléatoirement une seule solution voisine $s' \in N(s)$ à chaque itération, la méthode Tabou examine un échantillon de solutions de $N(s)$ et retient la meilleure s', même si s' est plus mauvaise que s. La recherche Tabou ne s'arrête donc pas au premier optimum trouvé, mais elle peut entraîner des cycles, par exemple un cycle de longueur 2 : $s \rightarrow s' \rightarrow s \rightarrow s' ...$ etc. Pour empêcher ce type de cycle, on mémorise les k dernières configurations visitées dans une mémoire à court terme et on interdit tout mouvement qui conduit à une de ces configurations. Cette mémoire est appelée *la liste tabou* qui est une des composantes essentielles de cette méthode. Elle permet d'éviter tous les cycles de longueur inférieure ou égale à k. La valeur de k dépend du problème à résoudre et peut éventuellement évoluer au cours de la recherche.

III.4.3.La méthode évolutive

Le terme « algorithmes évolutifs » englobe une autre classe assez large de métaheuristiques. Ces algorithmes sont basés sur le principe du processus d'évolution naturelle. Les algorithmes évolutifs doivent leur nom à l'analogie avec les mécanismes d'évolution des espèces vivantes. Un algorithme évolutif typique est composé de trois éléments essentiels :

1- une population constituée de plusieurs individus représentant des solutions potentielles (configurations) du problème donné.
2- un mécanisme d'évaluation de l'adaptation de chaque individu de la population à l'égard de son environnement extérieur.
3- un mécanisme d'évolution composé d'opérateurs permettant d'éliminer certains individus et de produire de nouveaux individus à partir des individus sélectionnés.

Du point de vue opérationnel, un algorithme évolutif typique débute avec une population initiale souvent générée aléatoirement et répète ensuite un cycle d'évolution composé de trois étapes séquentielles :
1- mesurer l'adaptation (la qualité) de chaque individu de la population par le mécanisme d'évaluation,
2- sélectionner une partie des individus,
3- produire de nouveaux individus par des recombinaisons d'individus sélectionnés. Ce processus se termine quand la condition d'arrêt est vérifiée, par exemple, quand un nombre maximum de cycles (générations) ou un nombre maximum d'évaluations est atteint. Par analogie à l'évolution naturelle, la qualité des individus de la population devrait tendre à s'améliorer au fur et à mesure du processus.

Les algorithmes évolutifs ont été appliqués à de très nombreux problèmes complexes et difficiles, y compris des problèmes d'optimisation (continue ou combinatoire).

D'une manière générale, on peut distinguer trois grandes familles d'algorithmes évolutifs : les algorithmes génétiques, la programmation évolutive et les stratégies d'évolution. Ces méthodes se différencient par leur manière de représenter les données et par leur façon de faire évoluer la population d'une génération à l'autre. **[Hao et al, 1999]**

III.4.3.1. Algorithmes génétiques

Les algorithmes génétiques classiques introduits par Holland s'appuient fortement sur un codage universel sous forme de chaînes 0/1 de longueur fixe et un ensemble d'opérateurs génétiques : la mutation, l'inversion et le croisement **[Erben 1995]**. Un individu sous ce codage, appelé un chromosome, représente une configuration du problème. Les opérateurs « génétiques » sont définis de manière à opérer aléatoirement sur un ou deux individus sans aucune connaissance sur le problème. Le croisement permet de produire deux nouveaux individus (enfants) à partir de deux individus (parents). Par exemple, le croisement bi-points consiste à choisir aléatoirement deux points de croisement et à échanger les segments des deux parents déterminés par ces deux points. Le croisement réalise donc uniquement des recombinaisons de valeurs (gènes) existantes entre deux parents et ne permet pas d'introduire de nouvelles valeurs dans les individus enfants. Pour cela, on applique la mutation. L'opérateur de mutation consiste à changer aléatoirement la valeur de certaines variables dans un individu. Dans les algorithmes génétiques la mutation est considérée comme un opérateur secondaire par rapport au croisement. Un cycle d'évolution complet d'un algorithme génétique est formé par l'application des opérateurs de sélection, croisement et mutation sur une population d'individus.

Bien que les algorithmes génétiques soient considérés aujourd'hui comme une méthode d'optimisation, l'objectif initial consistait à concevoir des systèmes d'apprentissage généraux, robustes et adaptatifs, applicables à une large classe de problèmes. En particulier, Holland s'est intéressé à l'élaboration d'une technique de programmation permettant l'évolution des programmes informatiques par reproduction, croisement et mutation. Cette motivation est à l'origine de l'universalité des algorithmes génétiques : ni le codage ni les opérateurs génétiques ne demandent des connaissances spécifiques du problème. Selon ce principe, pour un problème donné, il suffit de trouver une transformation des paramètres du problème en chaînes 0/1 et d'appliquer les opérateurs génétiques sur une population de solutions potentielles ainsi codées. C'est grâce à cette universalité que Holland a pu réaliser des analyses théoriques sur les algorithmes génétiques conduisant à la théorie des schémas et à la caractérisation du rôle et de l'importance du croisement. L'universalité d'un tel algorithme pose évidemment des problèmes d'efficacité en pratique. En effet, en tant que méthode d'optimisation, un algorithme génétique classique se base uniquement sur des opérateurs « aveugles » et est donc rarement en mesure de produire des résultats comparables à ceux d'une méthode de voisinage **[Goldberg, 1989]**. Une technique pour remédier à ce problème consiste à spécialiser l'algorithme génétique au problème donnée. Plus précisément, à la place des opérateurs aléatoires, la mutation et le croisement sont adaptés en se basant sur des connaissances spécifiques du problème. De cette manière, la recherche est mieux guidée et donc plus efficace. Il est désormais établi que pour être efficace en optimisation, il est indispensable d'intégrer des connaissances du problème **[Goldberg, 1989]**.

Une autre voie intéressante pour améliorer l'efficacité des algorithmes génétiques consiste à combiner le cadre génétique avec d'autres méthodes de résolution. Ce point sera développé ultérieurement dans la section sur les méthodes hybrides. Les algorithmes génétiques spécialisés ou hybrides ont été appliqués à de nombreuses applications dans des domaines très variés. Quant aux

algorithmes génétiques purs, leurs résultats en optimisation combinatoire sont en général faibles **[Goldberg, 1989]**.

III.4.3.2. Programmation évolutive

La programmation évolutive s'appuie sur un codage approprié du problème à résoudre et sur les opérations de mutation adaptées au codage **[Goldberg, 1989]**. Le codage d'un tel algorithme dépend du problème à résoudre. Par exemple, pour un problème d'optimisation dans le domaine des réels, les individus d'une population seraient des vecteurs de réels. De manière similaire, une permutation serait utilisée pour représenter un tour dans le problème du voyageur de commerce. A partir d'un codage donné pour un problème, la mutation ou opérateur d'évolution spécifique sera définie. Par exemple, dans le cas du problème du voyageur de commerce, la mutation basée sur le codage de permutation pourrait être l'opération d'inversion : prendre deux villes dans le tour et inverser l'ordre du segment défini par les deux villes. Ainsi l'analogie est forte avec les méthodes de voisinage : une mutation correspond à un mouvement dans un algorithme de voisinage. Un cycle d'évolution typique pour la programmation évolutive est le suivant : chaque configuration de la population courante est copiée dans une nouvelle population. Les configurations sont ensuite mutées, conduisant à de nouvelles configurations. L'ensemble des configurations entre ensuite dans une étape de compétition pour survivre dans la génération suivante.

La programmation évolutive a été initialement introduite pour simuler l'intelligence qui est définie sur l'hypothèse suivante : la caractéristique principale de l'intelligence est la capacité d'adaptation comportementale d'un organisme à son environnement **[Goldberg, 1989]**. Selon un modèle très simpliste, cette tâche de simulation revient à prédire une séquence de symboles appartenant à un alphabet fini à partir des séquences déjà observées. Dans ce but, des automates d'états finis sont choisis pour représenter les individus d'une population. Ainsi, à chaque automate de la population est donnée une série de symboles pris dans une séquence déjà observée et la sortie de l'automate est mesurée par rapport au résultat déjà connu. Cette mesure constitue l'adaptation de l'individu. L'étape suivante consiste à créer, pour chaque individu, un individu enfant par une mutation aléatoire de l'individu parent. La mutation est assurée par une des 5 opérations suivantes : changer le symbole d'une sortie, changer un état de transition, ajouter ou supprimer un état et changer l'état initial.

III.4.3.3. Stratégies d'évolution

Les stratégies d'évolution sont conçues dès le départ pour résoudre des problèmes d'optimisation continus **[Goldberg, 1989]**. Dans un algorithme SE, les individus sont des points (vecteurs de réels). Comme la programmation évolutive, les SE n'utilisent que la mutation et la sélection. L'algorithme le plus simple, noté (1+1)-ES, manipule un seul individu. A chaque génération (itération), l'algorithme génère par mutation un individu enfant à partir de l'individu parent et sélectionne l'un ou l'autre pour le conserver dans la population (selon l'adaptation de chaque individu). Le processus s'arrête quand la condition d'arrêt est vérifiée, définie souvent par le nombre d'itérations, le temps de calcul réalisé ou l'écart entre deux individus de deux itérations successives. La mutation dans un tel algorithme est aléatoirement appliquée à tous les composants de l'individu pour produire un enfant. Cette mutation fonctionne selon les principes suivants : un enfant ressemble à ses parents, et plus (moins) un changement est important, moins (plus) la fréquence de changement est élevée.

Cet algorithme (1+1)-ES se généralise en un algorithme (m+l)-ES qui signifie que m parents génèrent l enfants à chaque génération et qu'une sélection ramène ensuite la population de m+l individus à m individus. De plus, la recombinaison a été également introduite dans ces algorithmes **[Goldberg, 1989]**.

III.4.4. Les Méthodes hybrides

Le mode d'hybridation qui semble le plus fécond concerne la combinaison entre les méthodes de voisinage et l'approche d'évolution. L'idée essentielle de cette hybridation consiste à exploiter pleinement la puissance de recherche de méthodes de voisinage et de recombinaison des algorithmes évolutifs sur une population de solutions. Un tel algorithme utilise une ou plusieurs méthodes de voisinage sur les individus de la population pendant un certain nombre d'itérations ou jusqu'à la découverte d'un ensemble d'optimum locaux et invoque ensuite un mécanisme de recombinaison pour créer de nouveaux individus. La recombinaison doit impérativement être adaptée au problème traité.

Les algorithmes hybrides sont sans doute parmi les méthodes les plus puissantes. Malheureusement, le temps de calcul nécessaire peut devenir prohibitif à cause du nombre d'individus manipulés dans la population. A cet effet, une solution pour ce problème est la parallélisation de ces algorithmes sur des machines parallèles ou sur des systèmes distribués. Nous venons d'évoquer l'hybridation d'un algorithme évolutionniste avec des méthodes de voisinage. Il est également possible d'hybrider d'autres types d'approches, par exemple une heuristique gloutonne et une méthode de voisinage ou un algorithme évolutif.

III.5. Analyse

Dans cette section, nous présenterons quelques éléments d'analyse des métaheuristiques qui nous faciliterons le choix de la méthode de résolution pour le problème de la génération automatique des emplois du temps.

- **Exploitation Vs exploration :**

Les inventeurs des algorithmes génétiques ont introduit les notions de l'exploitation et de l'exploration **[Goldberg, 1989]**. L'exploitation insiste sur la capacité d'examiner en profondeur par une méthode des zones de recherche particulières alors que l'exploration met en avant la capacité de découvrir des zones de recherche prometteuses. Ces deux notions complémentations concernent au fait l'ensemble des méthodes de recherche. Il est donc pertinent d'analyser l'ensemble des métaheuristiques en fonction de ces deux notions.

Les méthodes de voisinage entendent exploiter les bonnes propriétés de la fonction de voisinage pour découvrir rapidement des configurations de bonne qualité. Elles reposent sur l'hypothèse que les zones les plus prometteuses sont situées à proximité des configurations (déjà visitées) qui sont les plus performantes. Le principe d'exploitation consiste à examiner en priorité ces zones. Dans les algorithmes génétiques, la sélection a pour effet de concentrer la recherche autour des configurations de meilleure performance. Et dans les méthodes de voisinage, l'exploitation est principalement assurée par la préférence accordée à une configuration de bonne performance dans la procédure de réparation.

En plus, les heuristiques emploient principalement deux autres stratégies dans le but d'explorer : la première consiste à perturber aléatoirement et la seconde à caractériser les régions visitées pour pouvoir ensuite s'en éloigner. La première stratégie qui est aussi la plus simple consiste à introduire des perturbations aléatoires : c'est le cas pour les mutations aléatoires dans les algorithmes génétiques ainsi que pour la génération aléatoire d'un voisin dans le recuit simulé.
La deuxième stratégie pour explorer consiste à mémoriser au cours de la recherche des caractéristiques des régions visitées et à introduire un mécanisme permettant de s'éloigner de ces

zones. C'est ce que fait la méthode tabou avec la liste tabou (court terme) et avec la diversification (long terme).

Nous venons de voir qu'il existe différents modes d'exploration. Les métaheuristiques se différencient également selon la manière dont elles font varier l'intensité de l'exploration au cours de la recherche. Le recuit simulé présente une manière spécifique de procéder : le niveau des perturbations aléatoires (responsables de l'exploration), lié au paramètre de température, est initialement fixé à un niveau élevé et abaissé progressivement au cours de la recherche.

- **Choix de la metaheuristique :**

Il est possible de caractériser les métaheuristiques selon quelques critères généraux, ce qui pourrait faciliter ce choix.

D'après l'analyse réalisée par J. K. Hao **[Hao et al, 1999]** une synthèse décrite par le tableau (III.1) met en relation cinq critères avec cinq métaheuristiques parmi les plus représentatives. Les critères de comparaisons retenus sont les suivants :

- Simplicité de la métaheuristique, *i.e.* simplicité de la méthode elle-même,
- Facilité d'adaptation au problème.
- Possibilité d'intégrer des connaissances spécifiques du problème,
- Qualité des meilleures solutions trouvées,
- Rapidité, *i.e.*, le temps de calcul nécessaire pour trouver une telle solution (sur une machine séquentielle).
-

	AI	RS	TABOU	AG	AH
Simplicité	0	-	-	-	--
Faciliter d'adaptation	0	0	-	-	-
Connaissance	0	0	+	+	+
Qualité	0	+	++	+	++(+++)
Rapidité	0	-	-	--	--

Tableau III.1 : « Comparaison générale des principales métaheuristiques. » **[Hao et al, 1999]**

Dans ce tableau, les cinq métaheuristiques comparées sont les suivantes :

- AI : amélioration itérative (descente).
- RS : recuit simulé.
- Tabou : méthode tabou.

- AG : algorithme génétique adapté.
- AH : algorithme hybride.

La méthode d'amélioration itérative est utilisée comme point de référence pour l'ensemble des méthodes : les signes -, 0, + indiquent des performances respectivement inférieures, égales, supérieures à celles obtenues par l'amélioration itérative.

Le critère de qualité utilisé dans le tableau correspond à la meilleure qualité qu'il est possible d'obtenir par une exécution prolongée. Le critère de rapidité représente le temps de calcul typiquement nécessaire pour obtenir une telle solution.

- **Bilan :**

Les métaheuristiques présentent de nombreux avantages mais aussi un certain nombre de limitations. Nous résumons ci-dessous les différentes caractéristiques de ces méthodes citées dans **[Hao et al, 1999]** :

- Généralité et application possible à une large classe de problèmes.
- Efficacité pour de nombreux problèmes.
- possibilité de compromis entre qualité des solutions et temps de calcul.
- nécessité d'adaptation de la méthode au problème traité.
- nécessité de réglage des paramètres.
- difficulté de prévoir la performance.

Afin de résoudre des instances de taille et de difficulté croissantes, il faut mettre au point des méthodes toujours plus puissantes. Pour atteindre cet objectif, au moins deux voies privilégiées se développent : l'hybridation de méthodes et les algorithmes génétiques. Les algorithmes génétiques répondent par performances supérieurs dans la possibilité d'intégrer des connaissances spécifiques au problème ainsi qu'à la meilleure qualité qu'il est possible d'obtenir par une exécution prolongée.

Dans cette partie nous avons dans un premier temps défini des notions de base sur les problèmes d'optimisation combinatoire et pour finir les différentes méthodes utilisées pour les résoudre.
Nous avons constaté que les métaheuristiques constituent une classe de méthodes approchées adaptables à un très grand nombre de problèmes. Elles ont révélé leur grande efficacité pour fournir des solutions approchées de bonne qualité pour un grand nombre de problèmes d'optimisation classiques et d'applications réelles de grande taille. C'est pourquoi l'étude de ces méthodes est actuellement en plein développement.

III.6 Problème de la génération des emplois du temps

Cette partie constitue le problème de la génération automatique d'emploi du temps. Nous présenterons les différentes formulations du problème trouvées dans la littérature. La problématique de la génération automatique d'emploi du temps sera présentée comme un problème de recherche dans un premier temps et comme un problème d'optimisation dans un second. Nous verrons ensuite les différentes variantes du problème d'emploi du temps trouvées dans la littérature. Et pour finir nous étudierons les différentes méthodes et techniques trouvées pour le résoudre. Parmi l'ensemble

des méthodes et techniques nous nous intéresserons plus particulièrement aux algorithmes génétiques.

III.6.1. Introduction

La littérature contient un grand nombre de variantes du problème calendrier. Ces variantes diffèrent les unes des autres par le type d'institution concernée (université ou haute école) et par le type des contraintes. Les annotations bibliographiques trouvées dans les articles étudiés du problème de calendrier dressent la liste de nombreux documents qui ont apparu avant 1980 **[Schaerf 1999]** dans une étude donne un aperçu des différentes formulations du problème de calendrier et les classifie en trois grandes classes : calendrier pour école, calendrier pour examen et calendrier pour cours universitaires. Bien entendu, cette classification est brut, et il y'a beaucoup de problèmes de calendrier dans le monde réel qui n'appartient pas ces trois catégories.

III.6.2. Différents problèmes et formulations

Le problème du calendrier scolaire de base est également connu sous le nom du modèle classe-enseignant (class-teacher model). Une classe est un groupe d'étudiants qui ont le même ensemble de cours et généralement demeurent ensemble tout au long de la semaine. Le problème le plus simple consiste à l'assignation de séances (leçons) à des périodes de telles manière qu'aucun enseignant ou classe ne soit impliqué dans plus d'une séance à la fois.

La problématique majeure dans cette version simplifier du problème de calendrier scolaire est quand les enseignants ou les classes ont plus de séances que de créneaux disponibles. D'un autre côté, on a les calendriers universitaires qui peuvent être classifié en deux grandes catégories. Calendrier pour examen et calendrier pour cours. La différence principale entre les deux types de calendrier est qu'on peut organiser plusieurs examens dans une même salle. **[Schaerf 1999]** donne une formulation par programmation linéaire au problème du calendrier pour examen et décrit des variantes alternatives.

Le problème du calendrier universitaire consiste en la planification d'un ensemble de cours dans un certain nombre de salles et de périodes. La principale différence entre le problème de calendrier universitaire et celui de l'école est que l'université peut avoir des cours qui regroupent des étudiants appartenant à des groupes différents. Si deux cours ont des étudiants en communes, y a conflit, et ils ne peuvent pas se tenir à la même période. En outre, le problème à l'université est que la disponibilité d'une salle (et sa taille) joue un rôle important, alors que dans l'école ils sont souvent négligés parce que, dans la plupart des cas, nous pouvons supposer que chaque classe a sa propre salle.

III.6.3. Le problème d'emploi du temps vu comme un problème de recherche

Plusieurs formulations peuvent être trouvées dans la littérature, celle proposée ici est issu de l'étude réalisée par Scharef **[Schaerf 1999]**. Il y a q cours K1, …, Kq. pour chaque cours Ki, m séances sont organisées et R programmes son crées S1, …, SR où Sj est le programme pour les cours qui regroupe les mêmes étudiants. Ce qui veut dire que les cours programmés dans Si doivent être programmé à des périodes différentes. Le nombre de période est P, et lk est le nombre de séance qui peut être planifié à la période K (c.à.d. le nombre de salle disponible à la période K). La formulation du problème est la suivante :

$$y_{ik} \qquad (i = 1..q; k = 1..p)$$

$$\sum_{k=1}^{p} y_{ik} = k_i \qquad (i = 1..q)$$

$$\sum_{i=1}^{q} y_{ik} \leq l_k \qquad (k = 1..p)$$

$$\sum_{i \in S_j} y_{ik} \leq 1 \qquad (j = 1..r; k = 1..p)$$

$$y_{ik} = 0 \text{ ou } 1 \qquad (i = 1..q; k = 1..p)$$

Où si $Y_{i\,k}=1$ si la séance du cours Ki est planifiée à la période k, et $Y_{i\,k}=0$ sinon.

- La contrainte (1) impose qui chaque cours ai un nombre de séance correct.
- La contrainte (2) vérifie pour chaque période, que le nombre de séance programmé ne dépasse pas le nombre de salle.
- La contrainte (3) évite que les séances qui sont en conflit ne soient organisées au même moment.

III.6.4. Le problème d'emploi du temps vu comme un problème d'optimisation

Sharef **[Schaerf 1999]** explique dans son étude que le même problème de recherche vu dans la section précédente peut être transformé en un problème d'optimisation en modifiant la fonction objectif de la façon suivante :

$$\max \sum_{i=1}^{q} \sum_{k=1}^{p} d_{ik} y_{ik}$$

Où $d_{i\,k}$ est la désirabilité de voir la séance du cours Ki planifie à la période k.

On peut passer d'un problème de recherche à un problème d'optimisation ou vis vers ça on choisissant d'intégré les contraintes dans la fonction objectif ou pas. Aussi, on peut intégrer une partie des contraintes et optimiser une autre partie. Dans la littérature, les contraintes d'un problème d'optimisation sont le plus souvent divisées en deux ensembles, les contraintes dures et les contraintes relâchées. Les contraintes dures sont celles qui doivent absolument être satisfaites par la méthode utilisée. Pour cette ensemble de contraintes le problème ressemble un problème de recherche. Les contraintes relâchées sont celles qui de préférence doivent être respectées. Pour ce deuxième ensemble de contrainte, le problème est un problème d'optimisation.

III.6.5. Variantes du problème d'emploi du temps universitaire

Maintenant nous allons essayer de voir brièvement quelques variantes du problème d'emploi du temps. Sharef dans son étude **[Schaerf 1999]** décrit un ensemble de variante qui diffère les institutions les unes des autres :

a) L'indisponibilité : En plus du problème de base, une contrainte qui concerne la disponibilité des enseignants peut venir s'ajouter au problème de base. Certains intervenants peuvent ne pas être disponibles durant des journées bien spécifiques au cours d'un semestre d'études et que l'emploi du temps doit prendre en charge.

b) Les sections : Dans certaines universités, certains cours se répètent plus d'une fois au cours de la semaine. En particulier, ces cours impliquant un grand nombre d'étudiants et appartenant à plusieurs programmes sont divisés en plusieurs sections. La création de différentes sections d'un cours peut contribuer à réduire le nombre de conflits dans un calendrier. Par exemple, supposons que le programme des études S1 comporte les cours K1 et K2 et le programme des études S2 comporte K1 et K3. Supposons également qu'une séance de K2 a lieu au moment p et une séance de K3 au moment q. Dans ce cas, la séance du cours K1 ne peut avoir lieu ni au temps p, ni au moment q. Toutefois, si K1 est donné pour deux sections, alors la séance d'une section peut avoir lieu au moment p, et la séance de l'autre, au moment q.

c) Répartition des étudiants : Une autre variante du problème traite le regroupement des étudiants comme faisant partie du problème d'emploi du temps. Le problème est résolu grâce à une procédure itérative qui essaye durant chaque itération de résoudre les deux problèmes. Dans un premier temps l'emploi du temps et dans un second les regroupements des étudiants.

d) Durée des séances : Une variante encore moins connu est celle ou la durée de la séance de même type est variable.

e) Affectation des salles : Concernant l'affectation des salles, deux choix sont envisageable : soit les groupes d'étudiant ne change jamais de salle durant l'année ou aucune salle n'est affecté par défaut.

Nous venons de voir ici les Cinque principales caractéristiques trouvées dans la littérature qui différent un calendrier universitaire d'une institution à une autre.

III.6.6. Différentes approches et techniques pour la résolution du problème

Un certain nombre de technique et approche ont été proposés dans la littérature pour résoudre le problème d'emplois du temps dans le domaine de l'éducation en général et l'emploi du temps universitaire en particulier. La plus part des première technique étaient basées sur la simulation de l'être humain pour la construction d'un emploi du temps **[Schaerf 1999]**. L'emploi du temps était construit pas à pas, on ajoute leçon âpres leçon à l'emploi du temps partiel en essayant de satisfaire l'ensemble de toutes les contraintes à chaque fois.

Par la suite, un ensemble d'autres méthodes ont été appliquées dans le domaine de la recherche pour résoudre le problème. Entre programmation linéaire en nombre entier, Réseaux de flux, Le problème a été même réduit à un simple problème de coloriage de graphe.

Plus récemment, des approches fondées sur les techniques de recherche utilisées également en Intelligence Artificielle apparu dans la littérature, nous avons recuit simulé, recherche tabou, algorithmes génétique **[Hao et al, 1999]**.

III.6.7. Algorithmes Génétiques appliqués aux emplois du temps universitaires

Comme nous venons de le voir ci-dessus, plusieurs formulations du problème d'emploi du temps universitaire ont été proposées et appliqué. Les approches récentes trouvées dans la littérature pour résoudre le problème des emplois du temps utilisent les algorithmes évolutionnaires et en particulier les algorithmes génétiques comme une puissante méthode pour résoudre le problème.

Différentes adaptations des algorithmes génétiques ont été appliquées pour la résolution du problème d'emploi du temps. Elles différent les unes des autres par :

1. Le type de codage utilisé pour les individus de la population.
2. Le choix de la fonction d'évaluation.
3. L'operateur de sélection.
4. Les opérateurs de croisement et de mutation spécifiques à la problématique et au codage défini.
5. La stratégie d'insertion des nouveaux individus générés grâce aux opérateurs de croisement et de mutation.
6. La sélection des paramètres pour l'exécution de l'AG (Taille population, taux de mutation, taux de croisement, etc…)

Néanmoins, le critère jugé pertinent pour différencier un algorithme génétique d'un autre pour la résolution du problème emploi du temps reste le codage. Les algorithmes génétiques peuvent être divisés en trois grands groupes sur la base de la représentation de leur gène direct, indirect et structuré. Une méthode directe encode directement tous les événements attributs (classe, cours, maître de conférences, salle, jour, etc), tandis que une méthode indirecte juste encode certaines parties de la solution. La phase de codage du gêne dans l'élaboration d'un algorithme génétique est un facteur très important pour la réussite final, le codage affecte la performance de l'algorithme, vitesse et qualité des résultats**[Karova, 2004]**.

Karova **[Karova, 2004]** explique brièvement les trois types de codage qui sont :

- **Le codage direct**

C'est le codage le plus simple, il ne tient compte d'aucune contrainte lors de la création de l'individu. L'évaluation et le respect des contraintes sont laissés entièrement à la fonction fitness. Pour cette raison, la représentation du chromosome ne nécessite aucun prétraitement. Adamidis et Arapakis **[Adamidis et al, 1999]** proposent une GA directe qui utilise une matrice NxM pour représenter le calendrier. Chaque élément de cette matrice est un gène. Un alphabet de caractères est utilisé pour représenter les attributs des événements. Un avantage de cette méthode est d'utiliser un algorithme de filtrage pour générer des solutions réalisables.

L'avantage de cette représentation est que le décryptage du chromosome est directe est ne nécessite aucune procédure particulière. L'inconvénient majeur est que la fonction fitness doit prendre en compte toutes les contraintes dures et relâchées ce qui nous donne une fonction fitness très compliqué et une convergence plus lente pour notre AG vu le nombre élevé de contrainte.

- **Le codage indirect**

Le codage direct n'est pas toujours possible. Quand l'espace de recherche est fortement contraint, le besoin de réduire cette espace est plus que nécessaire. C'est pour cette raison qu'on essaye de satisfaire un ensemble de contrainte dans la génération de la solution initiale.

Dans [1] [22] un codage indirect pour la résolution du problème d'emploi du temps a été utilisé, la solution est représentée telle une chaîne de caractères. Deux configurations sont possible, soit les indices du vecteur représentent les événements à planifier et le contenu est le créneau au quel est programmé l'événement ou l'inverse, les indices du vecteur représentent tous les créneaux possible est leur contenu représente les événements à planifier.

- **Le codage structuré**

Le codage structuré est le codage le moins utilisé, plusieurs chromosomes sont utilisés pour coder plusieurs niveaux de la structure. Ueda et al **[Mili 2010]** nt opté pour un codage structuré

pour la génération de la solution. Un premier codage est crée pour optimiser le problème à un premier niveau puis la solution trouvé est utilisée dans le codage de la solution globale à un deuxième niveau (figure 3).

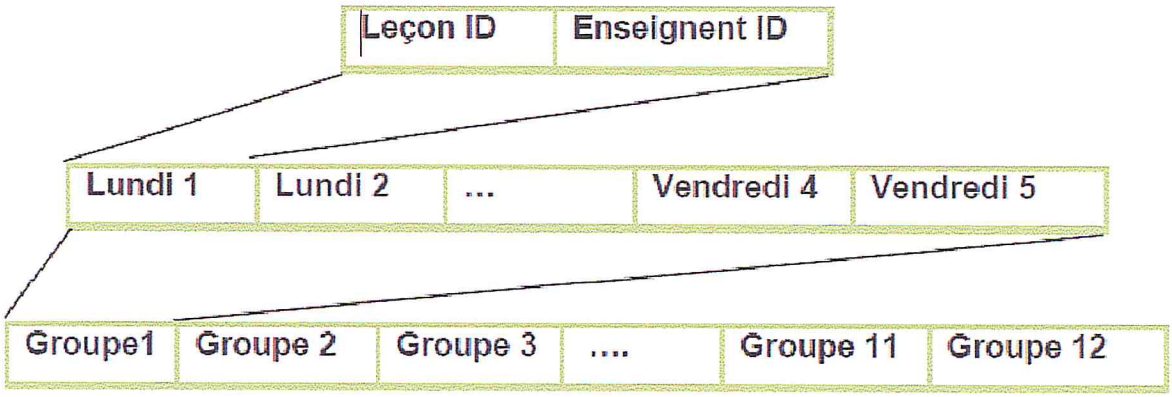


Figure III.3 : « Codage structuré pour la planification d'un groupe » **[Mili 2010]**

III.7. Conclusion

Parmi les trois codages présentés ci-dessus, le codage direct reste le plus utilisé au vu des articles traités.

Dans ce chapitre nous avons eu un aperçu général des différentes méthodes trouvé dans la littérature qui ont été utilisées pour la résolution du problème d'emplois du temps et en particulier les différentes adaptations des algorithmes génétiques au problème. Dans le chapitre suivant nous verrons une adaptation d'un algorithme génétique pour la résolution de notre problématique qui est la génération de l'emploi du temps de l'IAESB de Blida.

Chapitre IV

Conception

IV.1. Introduction

L'objectif de notre travail est de réaliser des services web pour la génération automatique des emplois du temps destinée à la pédagogie de l'institut d'aéronautique et des études spatiales de Blida. Le premier service web a pour but la génération automatique des emplois du temps et la gestion des mises à jour, tandis que le deuxième permettra aux enseignants de l'institut de consulter et de suggérer des modifications des séances de leur emploi du temps, et le dernier service permettra aux étudiants de l'institut de consulter les emplois de temps affectés selon leurs niveaux.

Ce chapitre abordera la conception de notre méthode d'adaptation des algorithmes génétiques pour la génération automatique des emplois du temps, et en second, la conception de nos services web en utilisant la modélisation VSoaML qui est une conception orientée service.

IV.2. Conception des services web

L'objectif de cette partie du chapitre est de décrire nos services web à l'aide du formalisme VSoaML **[Kenzi et al., 2009]** qui est un profil UML2.0 pour la modélisation des systèmes orientés services adaptables aux différents types d'utilisateurs. Ce profil est une extension d'UML2.0. Il définit les différents stéréotypes étendant les métaclasses UML2.0 dans l'objectif de spécifier et modéliser des systèmes orientés services adaptables.

Notre système comporte trois services web : administration, enseignant et étudiant.

- **Le service administration :** Ce service est responsable de la création et la gestion des emplois du temps de l'institut. Il est chargé de réaliser les opérations complexes suivantes :

✓ Gestion des enseignants.
✓ Gestion des sections/groupes.
✓ Gestion des salles.
✓ Gestion et génération automatique des emplois du temps.

- **Le service enseignant :** Ce service est dédié aux enseignants. Un enseignant peut effectuer les opérations complexes suivantes :

✓ Consultation des emplois du temps de ses enseignements.
✓ Contacter le service administration pour une éventuelle modification de ces séances.

- **Le service étudiant :** Ce service permet aux étudiants de l'institut de consulter leurs emplois du temps.

Nous commençons tout d'abord par définir notre conception orientée service en utilisant le formalisme VSoaML. Ensuite, nous présentons les différents stéréotypes du VSoaML ainsi que leurs métaclasses de base UML2.0. Nous décrivons chaque stéréotype en présentant sa description, ses attributs et ses associations.

IV.2.1. VSoaML

VSoaML est une extension du formalisme UML 2.0 adaptée pour les architectures orientées services. C'est une spécification de l'Object Management Group (OMG), qui fournit un Métamodèle pour la conception de services au sein d'une architecture orientée services. **[Adil, 2010]**

SoaML se base sur un ensemble de composants qui s'adaptent au niveau métier de l'entreprise :

- **Les participants** : sont utilisés pour définir les fournisseurs et les consommateurs de services dans un système, ils peuvent jouer le rôle d'un fournisseur de service, d'un consommateur de service ou les deux.
- **L'architecture de services** : est utilisée pour définir la manière dont les participants s'échangent les services.
- **Les contrats de services** : définis par des interfaces. Les contrats sont utilisés pour décrire les modèles d'interaction entre les participants. Un contrat est utilisé pour modéliser un accord entre deux ou plusieurs parties.

IV.2.1.1 VSoaML : Les stéréotypes

L'objectif de cette section est de présenter chaque stéréotype constituant le VSoaML. Le tableau IV.1 présente les différents stéréotypes du VSoaML. Dans la première colonne, nous présentons chaque stéréotype du VSoaML. Dans la deuxième, nous présentons la métaclasse de base de ce stéréotype. La troisième colonne présente une description succincte de chaque stéréotype.

Stéréotype (VSoaML)	Metaclasse (UML)	Description
Message	***Class***	<<Message>> est un élément de modélisation qui définit les structures de données échangées. Il spécifie aussi la propriété "Encoding" pour spécifier l'"encodage SOAP ou RPC.
Service	**Port**	<<Service>> est un élément de modélisation qui permet de représenter un point final de communication. Un service n'"identifie pas seulement ses interfaces fournies mais peut aussi identifier ses interfaces requises.
Channel	**Connector**	<<Channel>> est un élément de modélisation permettant décrire une connexion entre services. Il spécifie les liaisons a les protocoles de transport (SOAP, HTTP, JMS, etc.)

ServiceOpération	**Operation**	<<*ServiceOperation*>>permet de représenter une opération d'un service modélisant une fonctionnalité fournie d'un service donné. La spécificité de <<*ServiceOperation*>> est qu'elle manipule des données de grosse granularité ayant une valeur dans le cadre d'un processus métier. <<*ServiceOperation*>> manipule en entrée, en sortie ou en erreur des messages.
ServiceSpecification	**Artifact**	<<ServiceSpecification>> est un élément de modélisation permettant la spécification des opérations et leurs messages associés qui définissent le service.
Provider	**Class**	<<Provider>> est un élément de modélisation permettant la représentation d'un fournisseur d'un ou de plusieurs services.

Tableau IV.1 : « Stéréotypes du VSoaML »

- **Le stéréotype Service :**

Le stéréotype Service est un élément de modélisation permettant de représenter un point final de communication. Un service n'identifie pas seulement ses interfaces fournies mais peut aussi identifier ses interfaces requises. Ce stéréotype ne peut être utilisé qu'avec des éléments (classes, composants, systèmes, etc.) stéréotypés comme <<service Provider>>. Un service possède un attribut binding qui permet de spécifier la liaison à utiliser telle que SOAP-HTTP ou SOAP-JMS. Le stéréotype Service étend la métaclasse Port comme métaclasse de base d'UML2.0.

La figure IV.1 illustre le service Administrateur. Le service fournit une ou plusieurs interfaces. Chaque interface permet de représenter les fonctionnalités fournies par le service à ses clients.

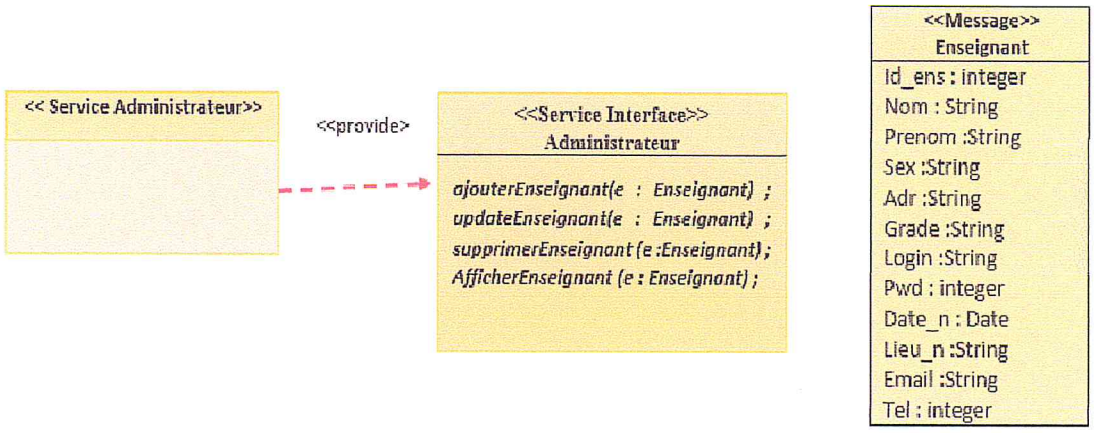


Figure IV.1 : « Service Administrateur »

Chaque opération du service possède un message en entrée et en sortie. Un message d'entrée permet de représenter les données manipulées par une opération de service. Un message de sortie représente les données résultat de l'exécution d'une opération donné. Le service fournit une interface de service Administrateur qui se compose d'opérations : ajouterEnseignant, updateEnseignant, supprimerEnseignant. Il permet à l'acteur Administrateur de gérer la gestion des enseignants. Ainsi, le service permet de créer des enseignants, de les modifier ou de les supprimer.

- **Le stéréotype Specification**

Le stéréotype Specification permet de définir la description des interfaces fournies par un service. Un service peut fournir bien évidemment plus d'une interface. Pour chaque service, nous pouvons associer un protocole (machine à états, diagramme d'activité, diagramme de séquence) pour spécifier l'ordre d'exécutions des opérations de services. Ce protocole permet de valider chaque service et tient compte ses aspects comportementaux. Ce stéréotype possède une propriété « published » qui permet d'indiquer si le service doit être publié dans un annuaire de service. La spécification d'un service peut contenir aussi des informations des propriétés non fonctionnelles telles que les propriétés de la qualité de service, de sécurité ou de performance. La spécification de service ne possède pas de propriétés et toutes les opérations doivent être publiques.

La figure IV.2 montre la spécification du service administrateur. Dans le cas des services web, une spécification de service correspond à une description WSDL. La description WSDL permet de décrire les interfaces de service, ses opérations d'entrée et de sortie, les ports et les formats de messages.

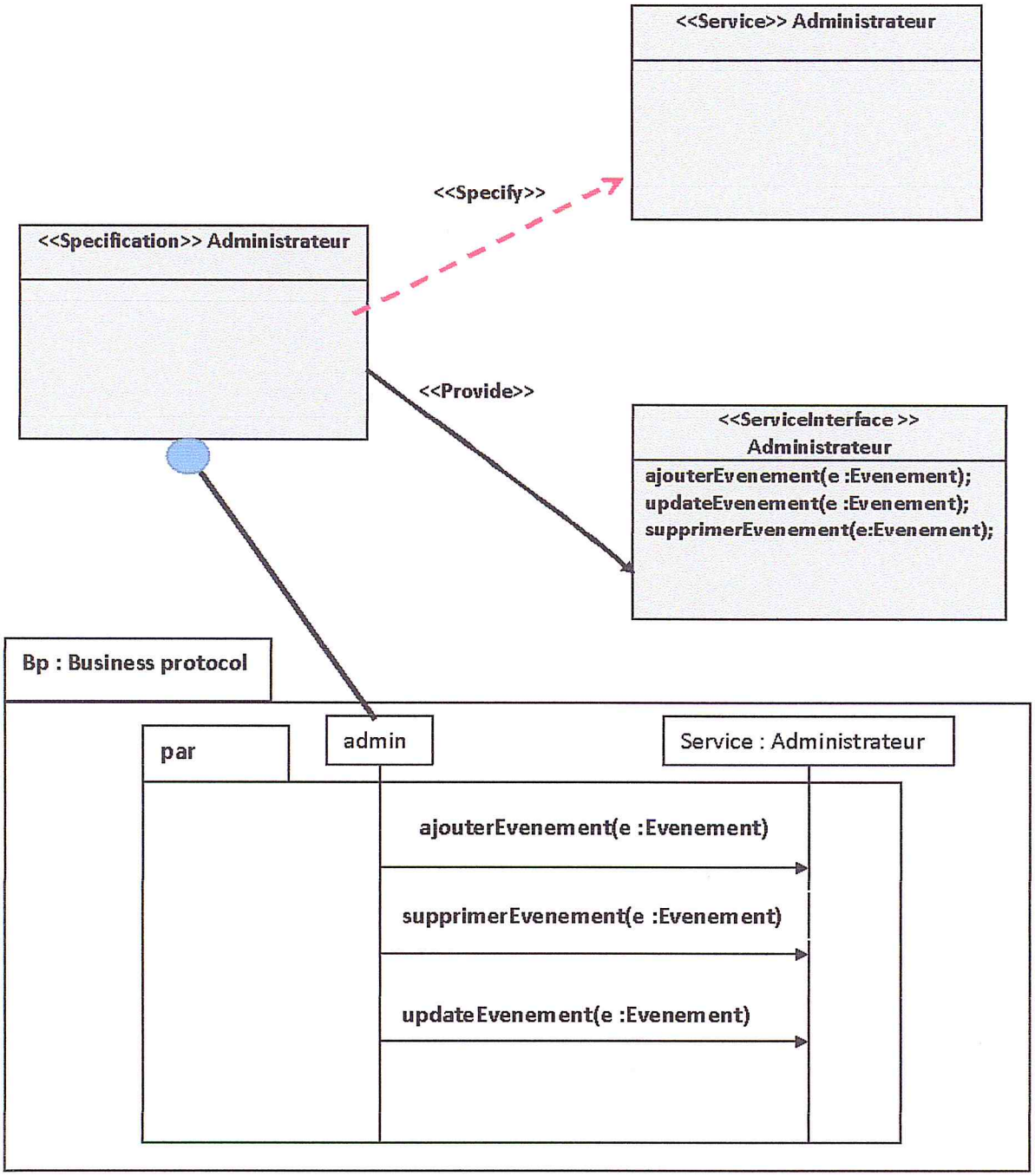


Figure IV.2 : « Specification du service Administrateur»

- **Le stéréotype Requester**

Le stéréotype Requester permet de représenter chaque classifier (Class, Component, etc.) qui peut être un consommateur de service. Ce stéréotype est utilisé pour spécifier les éléments d'un modèle qui ne sont pas des services mais des clients de services. L'utilisation de ce stéréotype est optionnelle. La figure IV.3 illustre la métaclasse Requester ainsi que ses associations.

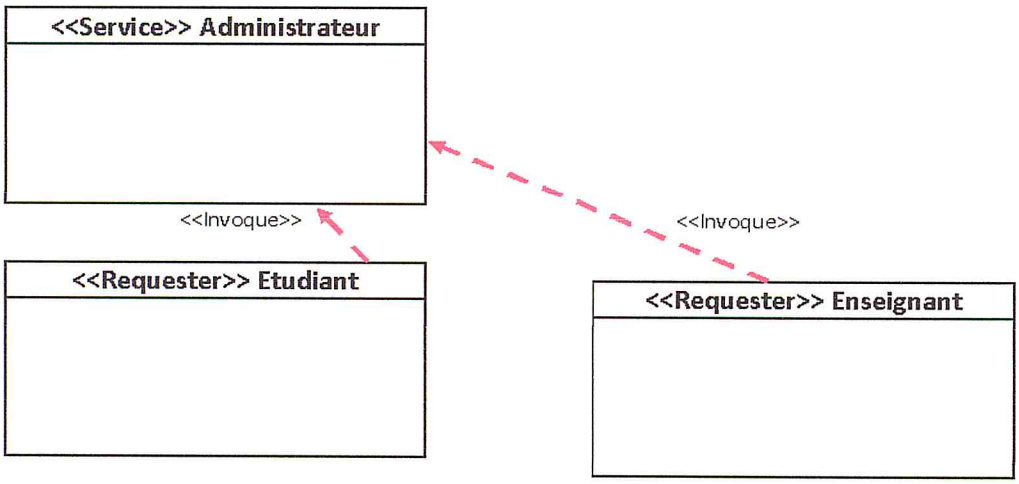


Figure IV.3 : « Specification de Service Administrateur »

De nos jours, le formalisme VSoaML en particulier et les modèles de la conception orienté services en général n'arrivent pas à définir tous les mécanismes et les processus métiers des services vu que ce sont des nouvelles technologies qui sont en cours de standarisation. On a constaté que le VSoaML est un langage UML particulier. C'est dans ce contexte, que nous avons décidé, dans notre travail, de compléter la conception orienté service par le standard UML. UML est le plus utilisé pour décrire un modèle de données. On dessine alors le diagramme de classe ainsi que le diagramme de séquence pour définir les propriétés et les évènements se déroulant dans le système.

IV.2.2 Diagramme de cas d'utilisation

Notre système est en interaction potentiellement avec trois acteurs (Figure IV.4) : l'étudiant, l'enseignant et l'administrateur. Pour chaque acteur, il fournit les fonctionnalités qui correspondent à ses besoins. Ainsi, le système fournit les fonctionnalités nécessaires à l'étudiant pour consulter, après authentification, les emplois du temps générés automatiquement par l'administrateur. Quant à l'enseignant, il utilise le système pour consulter après authentification les emplois du temps qui lui sont affectés, et pour suggérer des modifications auprès de l'administrateur.

Le système permet à l'administrateur d'effectuer la gestion et la génération automatique des emplois du temps en choisissant le nombre de section et de groupe à générer. Il lui permet aussi l'affichage des emplois du temps.

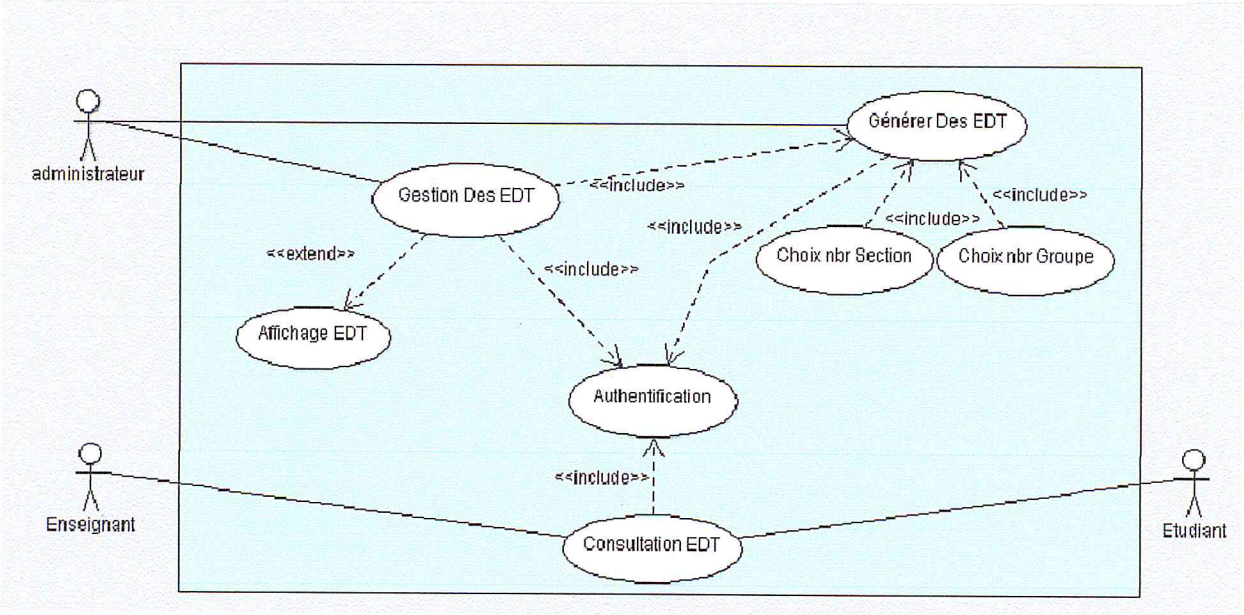
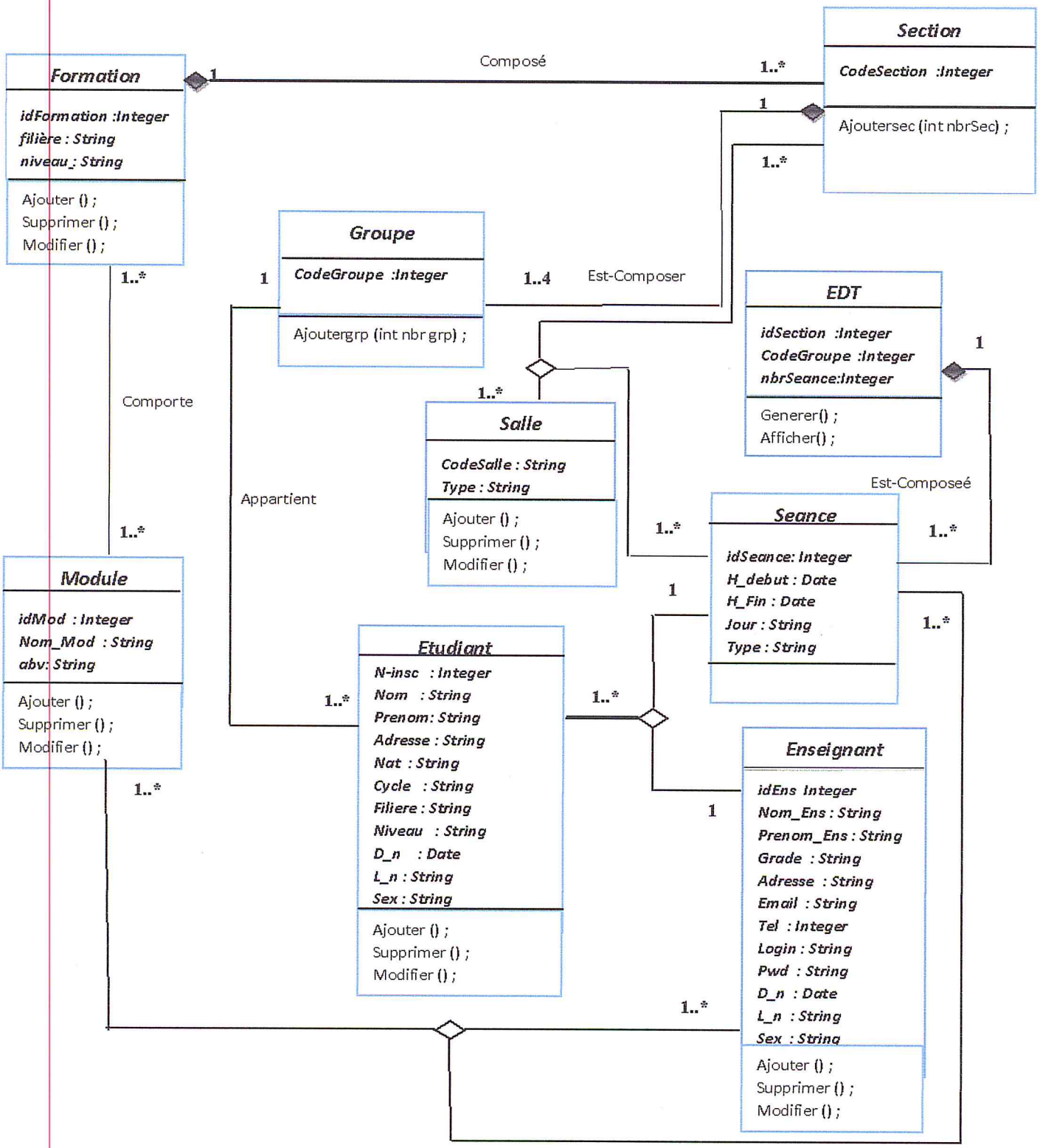


Figure IV.4 : « Diagramme de cas d'utilisation »

IV.2.3. Diagramme de classes

Un diagramme de classes permet de représenter les classes intervenant dans le système.il constitue un élément très important de la modélisation et permettre de définir quelle seront les composantes du système final. Il permet de structuré le travail de développement de manière très efficace. La figure IV.5 suivante représente le diagramme de classes de notre application.



« Diagramme De Classes »

IV.2.4. Diagramme de Séquence

La figure IV.6 représente un diagramme de séquence « génération emplois du temps » pour la génération automatique des emplois du temps :

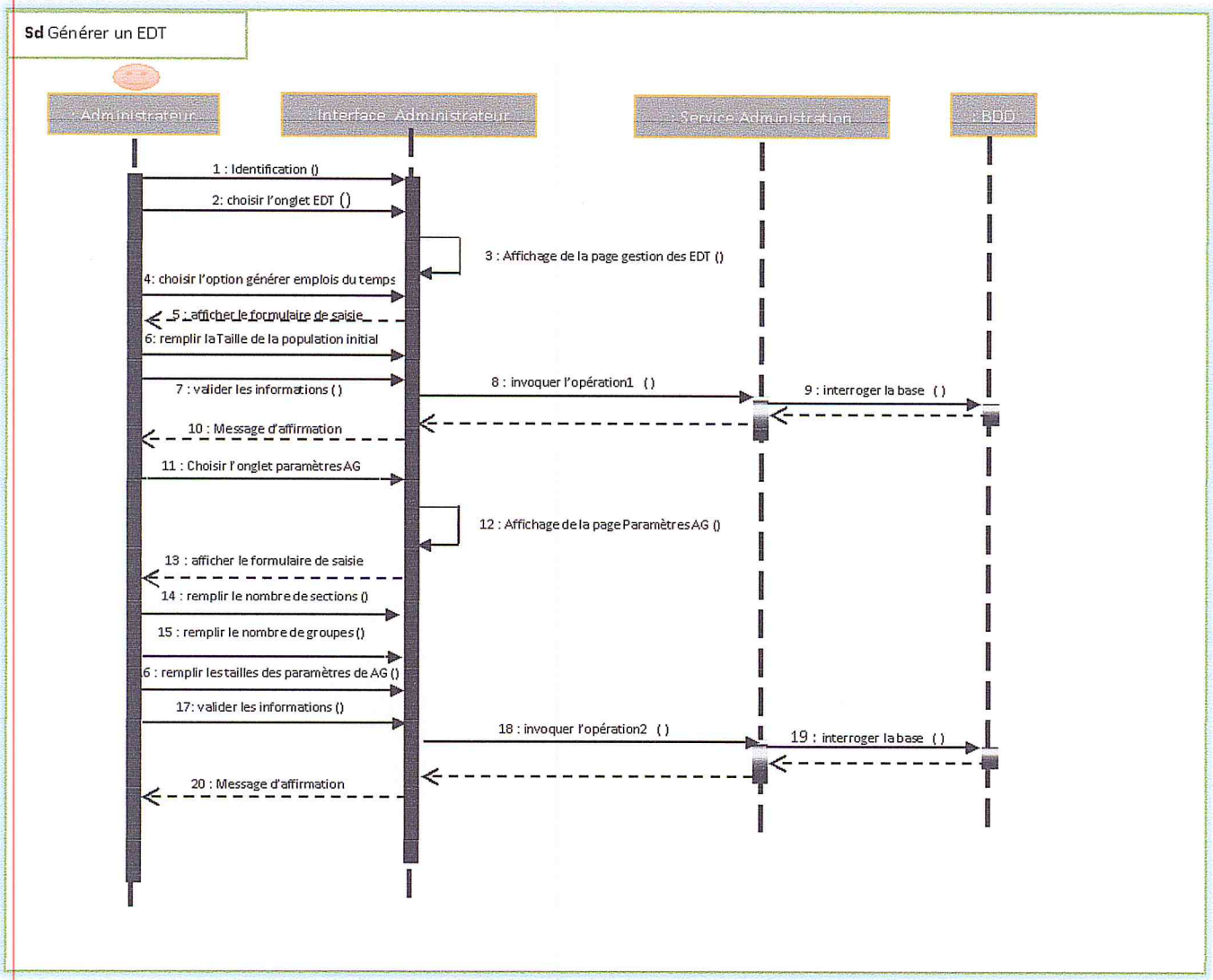


Figure IV.6 :« Diagramme de séquence de génération d'un EDT »

Pour générer des emplois du temps, l'Administrateur doit effectuer les principales tâches suivantes (Figure IV.6) :

1- Authentification.
2- Choix de l'option « Générer EDT ».
3- Le service lui renvois le formulaire de la saisie de la taille de la population initiale.

4- L'Administrateur invoque l'opération de la génération automatique de la population initiale en effectuant la saisie de la taille de la population initiale souhaitée, et le service interroge la base de données puis, il lui renvois un message de confirmation du succès de la génération de la population initiale.
5- L'Administrateur clique sur les paramètres de l'algorithme génétique pour saisir les paramètres d'entrées, et le service lui renvoie la page des paramètres de l'AG.
6- L'Administrateur choisi le nombre de section et le nombre de groupes pour lesquels, il générera des emplois du temps d'une manière automatique. Il remplit aussi la taille des chromosomes et le taux de mutation et de croisement désiré.
7- Il invoque l'opération de génération dans le service Administrateur, qui par la suite, ce dernier interrogera la base de données puis renvois un message de confirmation de la génération automatique des emplois du temps avec succès pour le nombre de groupes et le nombre de sections désirés dans l'étape (6).

IV.2.5. Interaction entre le service Administration et le service Enseignant

L'échange des messages entre le service Administration et le service Enseignant suit un ordre bien précis. Selon la figure suivante (Figure IV.7) :

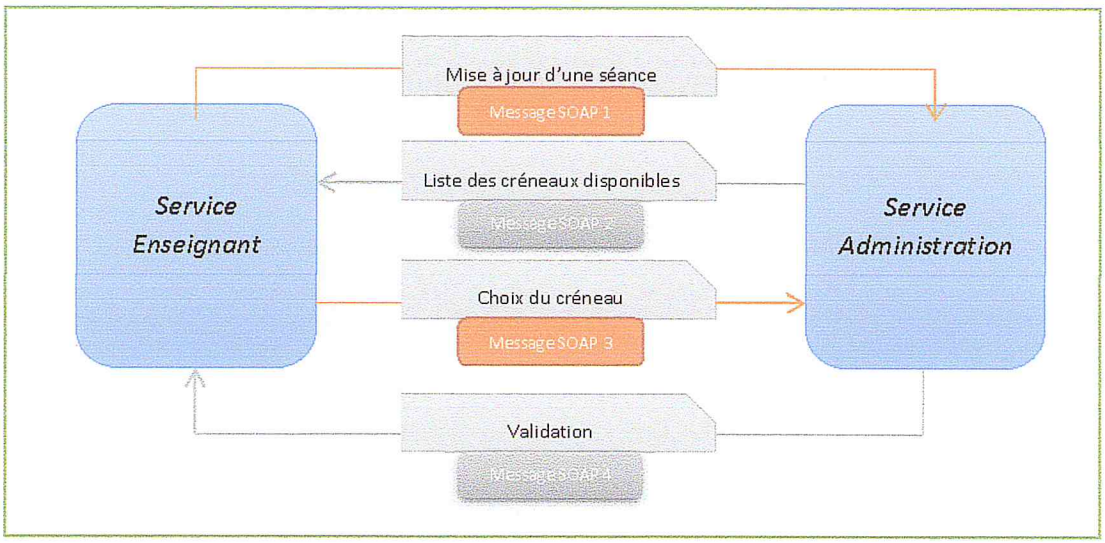


Figure IV.7 « Interaction entre service administration et service enseignant »

Message Soap	méthodes	Paramètres	Description
Message Soap 1	EnvoiMsgAdmin	Séance, jour, créneau	Le service enseignant envoie un message au service administration contenant la demande de modification d'une séance.
Message Soap 2	Verifcreneau	Séance, jour, créneau	Le service administration vérifie la disponibilité du nouveau créneau, s'il n'est pas disponible alors il renvoi un message au service enseignant contenant la liste des créneaux libres.
	EnvoiMsgEns	Créneaux libres	
Message Soap 3	EnvoiMsg Admin	Créneau choisi	Le service enseignant choisit un créneau et envoi son choix.
Message Soap 4	UpdateSeance	Séance, jour, créneau choisi	Le service administration valide le choix du service enseignant et effectue la mise à

Tableau IV.2 « Description des messages SOAP »

IV.3. Conception de la méthode d'adaptation des algorithmes génétique pour la génération automatique des emplois du temps

IV.3.1. Introduction

La confection d'emploi du temps est une tache fastidieuse et répétitive qui fait généralement intervenir de nombreux éléments d'information. La difficulté du problème qui en résulte est liée à la nature des contraintes qu'il s'agit de satisfaire. A celle-ci s'ajoute la crainte que quelques incompatibilités ne se manifestent après la publication de l'emploi du temps définitif. Les techniques manuelles utilisées actuellement par la plupart des responsables horaires sont portées à disparaitre. Ces derniers sont prêts à faire appel à des outils informatiques qui les soulageraient dans leur activité en leur fournissant un horaire complet.

Dans cette partie, nous détaillerons la conception d'une méthode d'adaptation. Nous proposerons une adaptation d'un algorithme génétique sous contraintes pour résoudre le problème de l'emploi du temps.

IV.3.2. Présentation des données

Les modèles mathématiques proposés dans la littérature sont souvent trop restrictif pour prendre en compte toutes les contraintes qui interviennent dans le processus de construction d'un emploi du temps. L'algorithme d'optimisation qu'il s'agit de concevoir doit être suffisamment flexible pour tenir en compte des particularités propres de l'établissement et des différents changements qui peuvent surgir par la suite.

Les éléments d'information ci-dessous sont pris en considération lors de l'établissement d'un emploi du temps :

(a) Un ensemble de périodes P= {p1, …, p_n} pendant lesquelles différentes leçons peuvent être placées.

(b) Un ensemble d'enseignants Ens = {ens1, …, ens_n}

(c) Un ensemble de Groupes GR= {gr1, …, g_n}. Chaque groupe regroupe un ensemble d'étudiants.

(e) Un ensemble de salles S= {s1, …, s_n}. Il y a plusieurs types de salles, salles de cours (Amphi), salles de Travaux dirigés (TD), et salles de Travaux pratiques TP (Laboratoire).

(g) Un ensemble de modules Mod = {mod1, …, mod_n} : Un module doit contenir :

- Un ensemble d'enseignants qui lui sont affectés.
- Le nombre de séance de cours durant la semaine.
- Le nombre de séance de TD durant la semaine.
- Le nombre de séance de TP durant la semaine.

IV.3.3. Formulation des contraintes

Un horaire peut être considéré comme satisfaisant s'il convient aux professeurs, et s'il respecte la disponibilité des salles. Le souci d'assurer un certain confort aux professeurs rend particulièrement difficile la confection d'un emploi du temps. Nous mentionnons ci-dessous les contraintes les plus fréquentes et celle dont nous allons tenir compte par la suite :

(1) Contrainte Enseignant :

- Un enseignant ne peut enseigner que les modules concernés.
- Le nombre de séances pour un enseignant est limité par 06 séances par semaine.
- Un enseignant peut enseigner un cours, un Td et un Tp.
- L'emploi du temps essaie de respecter la journée de disponibilité des enseignants.
- Un enseignant ne peut pas enseigner deux cours dans un même horaire.

(2) Contrainte Séance :

- Les séances de cours se feront durant les trois premières périodes (la matinée).
- Les séances de Td et Tp peuvent se faire la matinée ou l'après-midi.

(3) Contrainte d'affectation de salles :

- Une salle (Amphi, Salle TD, Laboratoire) n'est affectée qu'une seule fois dans un même horaire de la même journée.
- Choix de salles : les affectations des salles doivent respecter les types des modules (Cours dans un Amphi, TD dans une salle et un TP dans un laboratoire).

(4) Contrainte Groupe :

- Les groupes d'une même section partagent les même cours, dans une même salle (Amphi) avec le même enseignant et dans un même horaire (jour et heure).

L'existence d'un horaire respectant à la lettre toutes les contraintes n'est pas garantie dans le cas général. En réalité, il s'agit de trouver un horaire aussi bon que possible dans un temps de calcul raisonnable tout en tolérant la présence de conflits ; le nombre de ces conflits doit être minimisé pour que l'horaire soit jugé satisfaisant par les différents parties concernées (direction de l'université, enseignant, responsables...etc.)

IV.3.4. Modélisation du problème

Soit C l'ensemble des contrainte (1) à (4) décrites dans la section précédente. Une contrainte est dite Dure si :

• Un emploi du temps est acceptable si toutes les contraintes énoncées dans la section précédente sont satisfaite.

• Un emploi du temps est admissible si toutes les contraintes dures sont satisfaites.

Selon la définition précédente, un emploi du temps est acceptable s'il est admissible. La modélisation de la confection d'un emploi du temps en termes de problème d'optimisation combinatoire découle des définitions précédentes.

IV.3.5. Adaptation des algorithmes génétiques

Les algorithmes génétiques tentent de simuler le processus d'évolution naturelle suivant le modèle darwinien dans un environnement donné. Ils utilisent un vocabulaire similaire à celui de la génétique naturelle. Cependant, les processus naturels auxquels ils font référence sont beaucoup plus complexes. On parlera ainsi d'individu dans une population. L'individu est composé d'un chromosome lui-même constitue de gènes qui contiennent les caractères héréditaires de l'individu. Les principes de *sélection*, de *croisement* et de *mutation* s'inspirent des processus naturels de même nom.

Pour un problème d'optimisation donné, un individu représente un point de l'espace d'états. On lui associe la valeur du critère à optimiser, sa Fitness. On génère en suite de façon itérative des populations d'individus sur lesquelles on applique des processus de sélection, de croisement et de mutation. La sélection a pour but de favoriser les meilleurs éléments de la population, le croisement et la mutation assurent une exploration efficace de l'espace d'états. On commence par générer une population d'individus. Pour passer d'une génération K à la génération K+1, les trois opérations suivantes sont répétées pour tous les éléments de la population K.

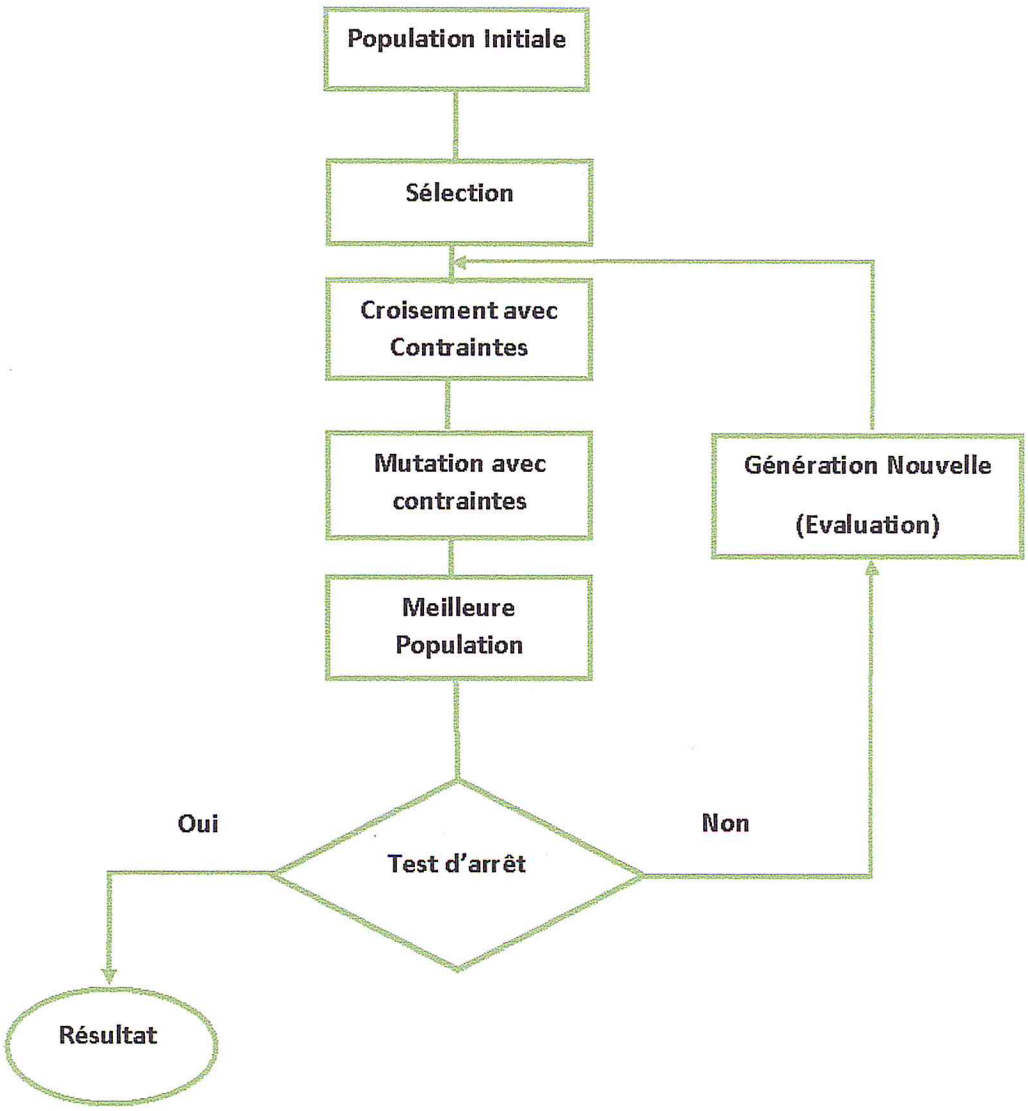


Figure IV.8 : « Schéma fonctionnel de l'algorithme »

La figure IV.8 représente le schéma fonctionnel de notre algorithme génétique d'adaptation pour le problème de la génération des emplois du temps. Au départ, une population initiale est

générée en satisfaisant au maximum les contraintes. Des couples de parents P1 et P2 sont sélectionnées en fonction de leur fitness. L'operateur de croisement leur est appliqué avec une probabilité P_c (entre 0.7 et 0.95) et génère des couples d'enfants C1 et C2. D'autres éléments P sont sélectionnés en fonction de leur adaptation, l'operateur de mutation leur est appliqué avec la probabilité P_m (est généralement inférieur à 0.1) et génère des individus mutés P'. La Fitness des enfants (C1, C2) et des individus mutés est ensuite avalée avant insertion dans la nouvelle population.

Plusieurs critères d'arrêt de l'algorithme sont possibles :

- le nombre de générations peut être fixé à priori.
- le nombre de solutions à générer est trop grand par rapport aux ressources (salles, enseignants,...)
- l'algorithme peut être arrêté lorsque la population n'évolue plus suffisamment rapidement.

Pour utiliser un algorithme génétique sur un problème d'optimisation on doit donc disposer d'un principe de codage des individus, d'un mécanisme de génération de la population initiale et d'opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace de recherche.

IV.3.5.1. Génération de la population initiale (Solution)

Elle est usuellement aléatoire (diverses stratégies sont d'ailleurs envisageables pour échantillonner correctement un espace de recherche complexe ou de grande dimensionnalité). C'est là que l'on peut injecter la ou les solutions initiales au problème que l'on a pu obtenir par ailleurs (par exemple à l'aide d'autres techniques de résolution). Si l'initialisation a théoriquement peu d'importance (on converge en limite toujours vers l'optimum global), on constate expérimentalement que cette étape influence énormément la variance des résultats (et aussi la rapidité d'obtention de ceux-ci !). Il est ainsi souvent très avantageux d'injecter le maximum de connaissances à priori sur le problème par le biais de l'initialisation : placer dans la population initiale des individus que l'on sait proches par différents aspects de l'optimum recherché ne peut qu'accélérer la convergence de l'algorithme.

Le tableau IV.3 suivant présente la codification des enseignants et les modules, ainsi que leurs significations :

Id_Ens	Ensignant	Id_Module	Module
1	Amtout	1	Math 2
2	Chikh	2	Physique 2
3	Merarzia	3	Chimie 2
4	Lagha	4	Français
5	.Amzal	5	Science d'expression
.	.	6	Informatique
.	.		
N		7	Science Aéronautique

Tableau IV.3 : « Données codifiées »

Dans notre cas, une méthode constructive est utilisée pour établir un emploi du temps initial. Tout d'abord deux structures de données sont créés (Voir Tableau IV.4 et IV.5).

La première structure trie les données modules, type, et nombre de séances. Elle est représentée par le tableau suivant :

Id_Mod	Type	Nbr_Seance
1	C	2
2	Td	1
3	C	2
4	C	1
.	.	.
.	.	.
.	.	.
N	Tp	2

Tableau IV.4 : « Données Module-Type-Nombre de Séance »

La deuxième structure trie les données enseignant, module et type. Elle est représentée par le tableau suivant :

Id_Ens	Id_Mod	Type
1	1	C
2	5	C
3	2	C
4	6	C
.	.	.
.	.	.
.	.	.
14	3	Td
.	.	.
.	.	.
M	6	Tp

Tableau IV.5 : « Données Enseignant-Module-Type »

- **Individu :**

Notre population initiale est composée de plusieurs individus, et chaque individu est représenté par 6-tuplets de nombre de séance, dans notre cas, de 17 séances. Un tuplet est défini par un type, module, enseignant, jour, heure et salle comme l'illustre le tableau (IV.6). Il représente une séance parmi celle comprise dans un emploi du temps.

Type	Module	Enseignant	Jour	Heure	Salle

Tableau IV.6 : « Définition d'un 6-tuplet »

Un individu est alors constitué d'un tableau codifié qui contient exactement Mx6 cases. Tel que les indexes du tableau représentent les événements et leurs contenu les créneaux auxquels ils sont affectés (voir Figure IV.9).

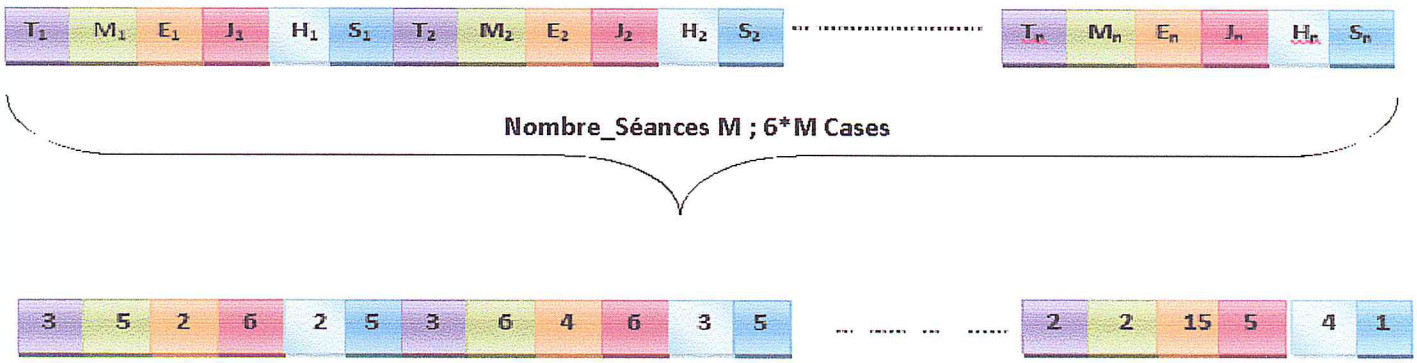


Figure IV.9 : « Individu »

Dans cette étape de la construction de la population initiale, nous avons essayé de satisfaire au maximum les contraintes définies auparavant dans le but d'accélérer le processus d'évolution des nouvelles populations.

IV.3.5.2. Operateur de sélection

La méthode utilisée est la sélection par rapport à la fonction objective Fitness ; elle consiste à associer à un chromosome i de la population une portion égale à :

$$F_i = \sum_{j=1}^{m} P_j$$

Où Fi est la valeur de la fonction objective de l'individu i, et Pj représente le poids affecté pour chaque case (donnée). Ainsi les individus qui sont triés à l'ordre décroissant en fonction de leurs Fitness peuvent avoir une chance d'être choisis lors des opérations de croisement et de mutation.

IV.3.5.3. Operateur de croisement

L'operateur de croisement qu'on a développé pour notre type de codage manipule deux parents choisis en fonction de la Fitness de chaque individu parmi les individus sélectionnés pour le croisement. Le processus passe deux étapes (voir figure IV.10) qui sont :

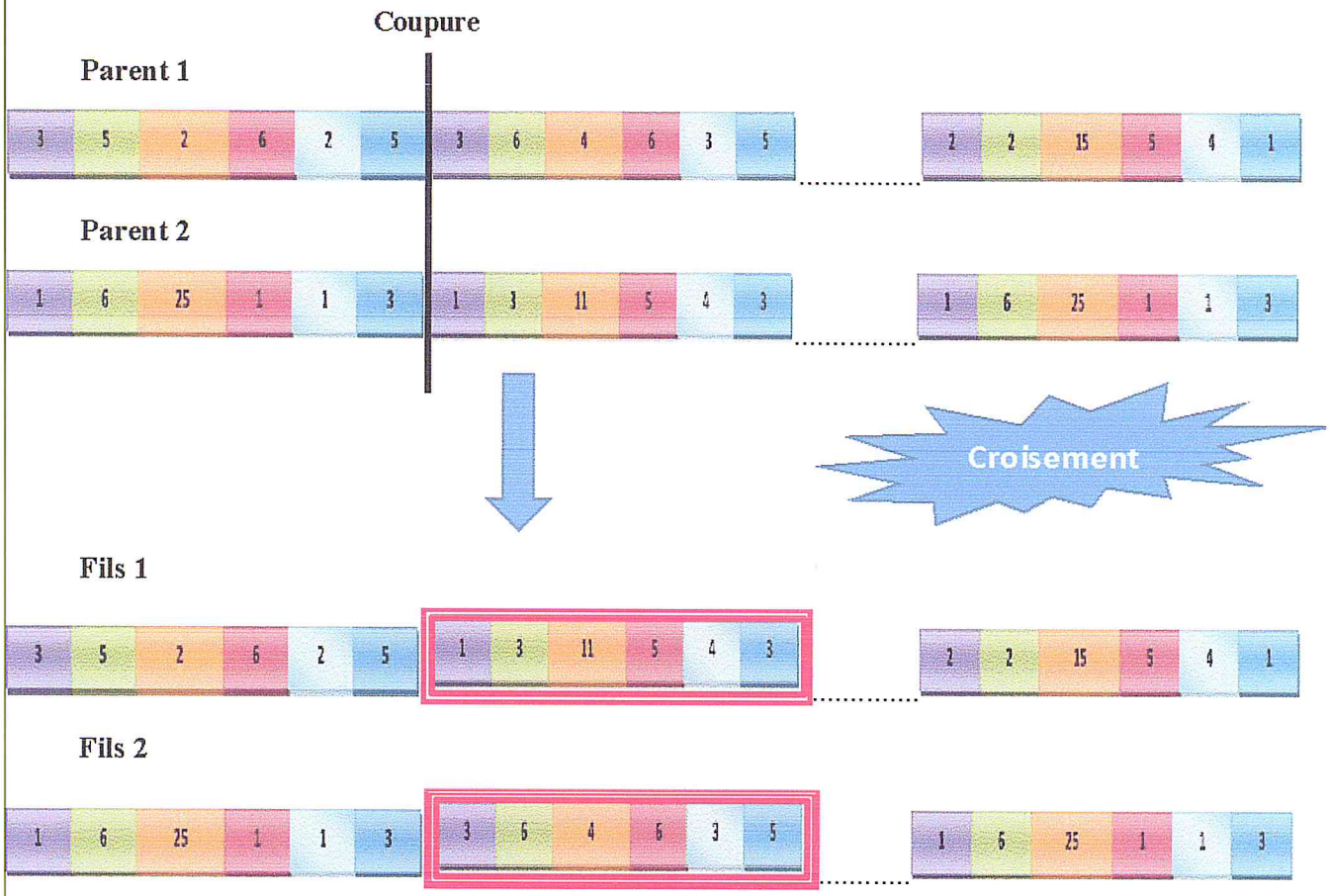


Figure IV.10 : « Opérateur de croisement »

- **Etape 1** : La coupure de croisement pour les deux parents P1 et P2 est effectuée est donnée par une fonction aléatoire (random()). Cette fonction choisis un nombre aléatoire y compris entre 0 et Mx6 qui nous permettra de fixer la coupure ou le croisement serais sensé s'effectuer.

- **Etape 2** : On effectue un croisement uniforme entre les deux parents choisi avec une probabilité de croisement P_c à notre choix y comprise entre 0.7 et 0.95 pour les parents.

Une procédure de vérification des contraintes est lancée au moment du croisement pour vérifier si ce croisement sera possible ou non, comme dans le cas où on obtient des fils avec des créneaux doubles, ou dans le cas d'un dépassement des contraintes, le croisement serait annulé pour ces gènes.

IV.3.5.4. Operateur de mutation

Les opérateurs de mutation qu'on a développés évitent d'établir des populations uniformes incapables d'évoluer. Ils permettent de modifier les valeurs des gènes de chromosomes. Nous avons défini les opérateurs de mutation utilisés entre des gènes d'un même niveau (même chromosome),

et de même type, c'est-à-dire qu'on ne peut pas faire de mutation à partir d'une permutation entre un enseignant et une salle, mais par contre, la mutation à partir de la permutation entre enseignants est décrite faisable dans cette étape. Bien sûr, différemment des opérateurs de croisement, ceux de la mutation n'ont besoin que d'un seul chromosome.

- **Mutation par inversion (permutation) :**

Elle consiste à choisir deux gènes de même type aléatoirement et à échanger leur contenu (Créneaux, enseignants, salles, ... etc.) avec une probabilité de mutation Pm entre 0.01 et 0.1.

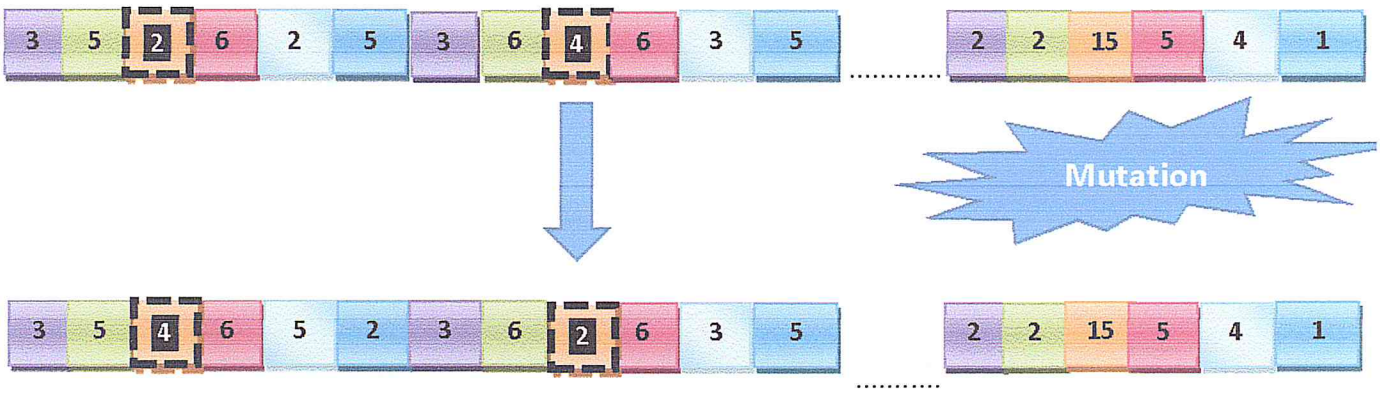


Figure IV.11 : « Opérateur de mutation par Inversion »

La figure IV.11 illustre un exemple de mutation par inversion entre deux gènes du même type qui est enseignant, ces deux enseignants peuvent se permuter entre eux à condition qu'il y'aurait pas de doublures et qu'à chaque fois en vérifie toutes les contraintes qui doivent être satisfaites dans tous les cas de mutation.

- **Mutation par changement de créneau :**

Elle consiste à choisir un gêné aléatoirement et à le remplacé par un créneau non utilisé par l'individu avec une probabilité de mutation Pm.

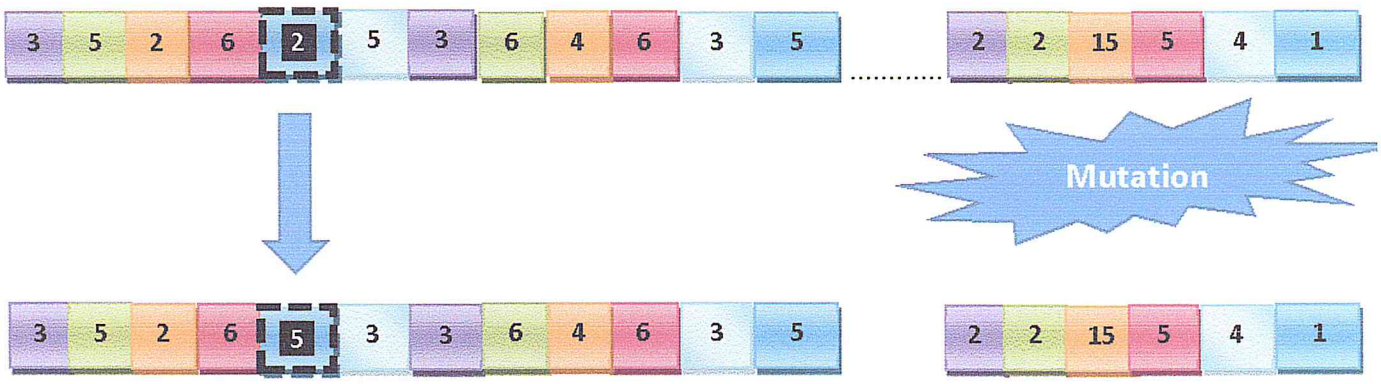


Figure IV.12 : « Opérateur de mutation par changement de Créneaux »

La figure I.V.12 illustre un exemple de mutation par changement de créneau dans un même gène. Le gène (2) qui représente un créneau est remplacé par un autre créneau (5) à condition que ce dernier satisfasse les contraintes décrites auparavant (dans les sections précédentes). Un créneau n'est remplacé que par un créneau qui n'existe pas déjà dans le même chromosome (voir figure IV.12).

IV.4. Conclusion

Dans cette partie conception du mémoire, en premier temps, nous avons décrit nos services web à partir d'une conception orientée service d'une part à l'aide du formalisme VSoaML et une conception d'un modèle de donnée à l'aide du standard UML. En second, nous avons conçu une méthode d'adaptation des algorithmes génétiques avec contraintes pour la génération automatique de l'emploi du temps de l'IAESB.

Le prochain chapitre fera l'objet de la réalisation et l'implémentation de nos services web et de notre méthode d'adaptation des algorithmes génétique pour la génération automatique des emplois du temps feront l'objet du dernier chapitre de ce mémoire.

Chapitre V

Implémentation

V.1 Introduction

Apres avoir décrit dans le chapitre précédent les différents services web et la méthode d'adaptation que constituent notre système de gestion des emplois du temps, l'implémentation et la réalisation fera l'objet du dernier chapitre de notre mémoire.

Nous allons présenter d'abord l'interface de notre application et les différentes taches de nos services web, et ensuite, nous présenterons l'implémentation et la réalisation de notre méthode d'adaptation avec les résultats obtenus lors des expériences.

V.2 Environnement de travail

V.2.1 Environnement matériel

Pour l'implémentation de nos services web on a disposé d'un ordinateur de type Acer dont la configuration par le tableau V.1 :

Processeur	**Intel(R) Core (TM) i5-4200 CPU @ 2.50 GHz 2.50 GHz**
Mémoire	2 Go
Disque dur	320 Go

Tableau V.1 « Configuration de l'ordinateur de développement »

V.2.2 Environnement logiciel

- **Langages de programmation**

✓ **Java :** Est un langage de programmation à usage général, évolué et orienté objet dont la syntaxe est proche du Langage C. Ses caractéristique ainsi que la richesse de son écosystème et de sa communauté lui ont permis d'être très largement utilisé pour le développement d'applications de types très disparates. Java est notamment largement utilisé pour le développement des services web.

✓ **JavaScript :** Est un langage de script orienté objet utilisé pour le développement d'applications Internet. Ce langage a été développé par la société Netscape Corporation, qui l'a introduit, pour la première fois dans son Navigator 2.0 en 1996. L'intérêt d'un langage come JavaScript est de pouvoir contrôler dynamiquement le comportement d'une page Web. Aujourd'hui,

JavaScript est le langage le plus couramment utilisé par les créateurs de pages Web afin de les animer.

✓ **SGBD MySQL :** Les SGBD libres et gratuits sont nombreux. MySQL, MS Access, en sont des exemples. Si on a choisi MySQL, c'est plus pour des raisons de performances et pour les fonctionnalités offertes. Nous citons les principaux avantages :

- MySQL est beaucoup moins complexe à installer et à administrer que les autres systèmes.
- MySQL support le langage de requête SQL et l'accès se fait via le pilote JDBC.
- MySQL permet des connexions multiples et utilise simultanément différentes bases de données.

✓ **Logiciel NetBeans :** On a utilisé la version 7.2 de ce logiciel pour la création de nos services web. C'est un outil de développement puissant, qui se distingue par sa facilité d'utilisation due à son ergonomie qui utilise toutes les technologies possibles d'aide au développement.

✓ **Logiciel DreamWeaver :** DreamWeaver CS6, logiciel créé par MacroMedia (et géré maintenant par Adobe) fonctionne en mode création ou en mode code. D'après les statistiques DreamWeaver est le lead des environnements de développement web.

V.3 Les approches de développement des services web

Les services web peuvent être développés selon deux approches :

- **Un développement Bottom-up :**

Le développement « **Bottom-up** » consiste à créer le service web à partir d'une classe java en utilisant les annotations @WebService et @WebMéthode. Le fichier WSDL est généré automatiquement. L'approche « Bottom-up » est l'approche la plus rapide pour développer des services web.

- **Un développement Top-Down :**

Le développement « **Top-down** » consiste à créer manuellement le fichier WSDL ainsi les XML schémas associés à ce fichier. Après la création du fichier WSDL, l'IDE offre un outil pour la génération des classes java à partir de ce fichier.

Pour la réalisation, On a utilisé l'approche « **Bottom-up** » pour le développement de nos services web.

V.4 Présentation de l'application

Dans cette partie, on va représenter les différentes étapes effectuées pour réaliser cette application, et pour bien illustrer ces étapes on a choisi de présenter quelques exemples.

V.4.1 Page d'accueil

Le prototype de l'application qu'on a développé se compose de trois interfaces principales contenu dans l'interface d'accueil présenté dans la figure V.1 et Figure V.2 et qui sont respectivement : Espace administratif (1), Espace enseignant (2), Espace étudiant (3).

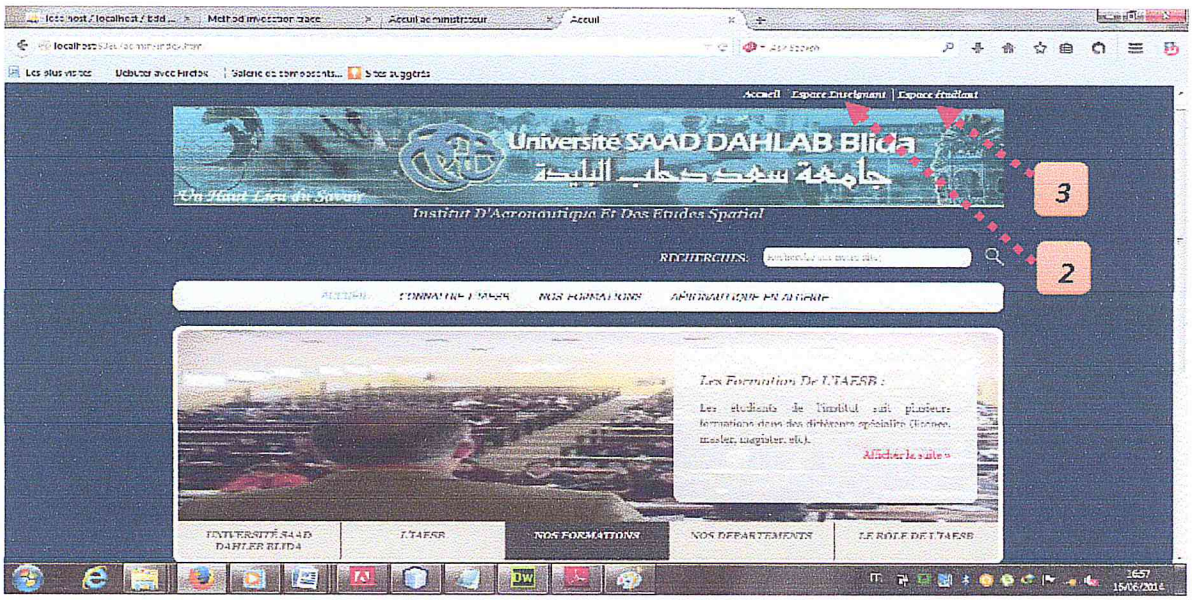


Figure V.1 « Interface d'accueil (A) »

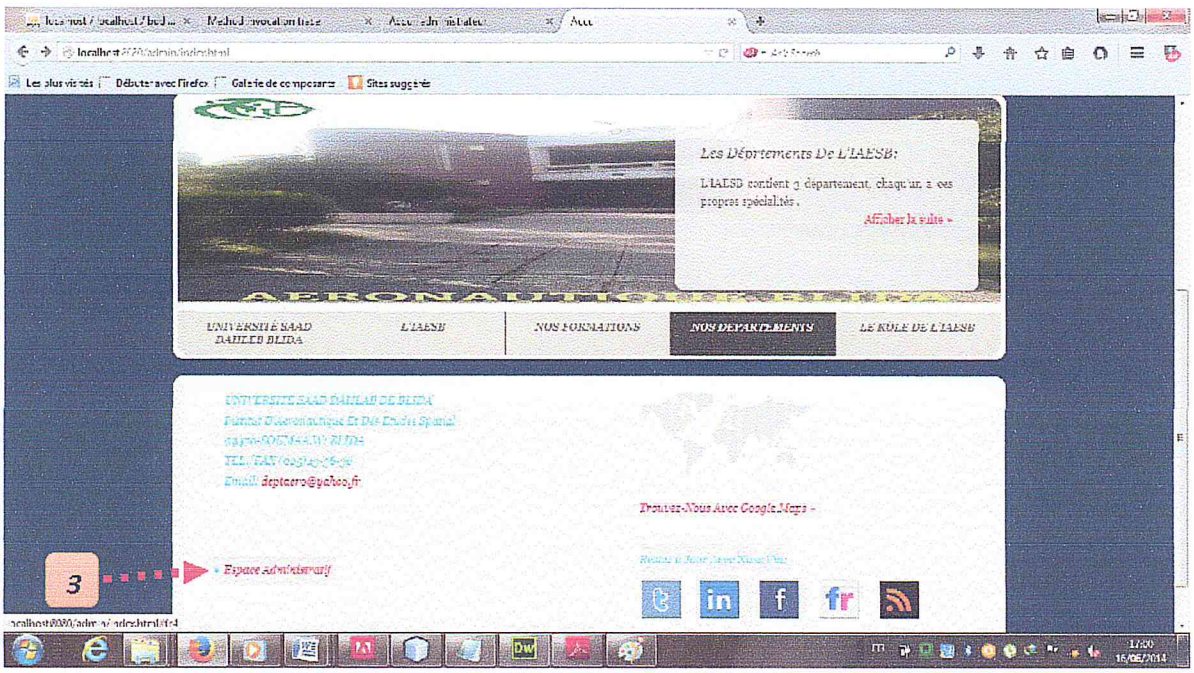


Figure V.2 « Interface d'accueil (B) »

V.4.2 Page d'Authentification

Les pages des trois interfaces développées constituent les portes d'accès à l'application. L'administrateur (respectivement l'enseignant et l'étudiant) est invité à introduire un mot de passe et un login pour s'identifier, si un problème d'identification surgit (erreur de mot de passe ou login) un message d'erreur est affiché.

La figure V.3 représente la page d'authentification de l'administrateur.

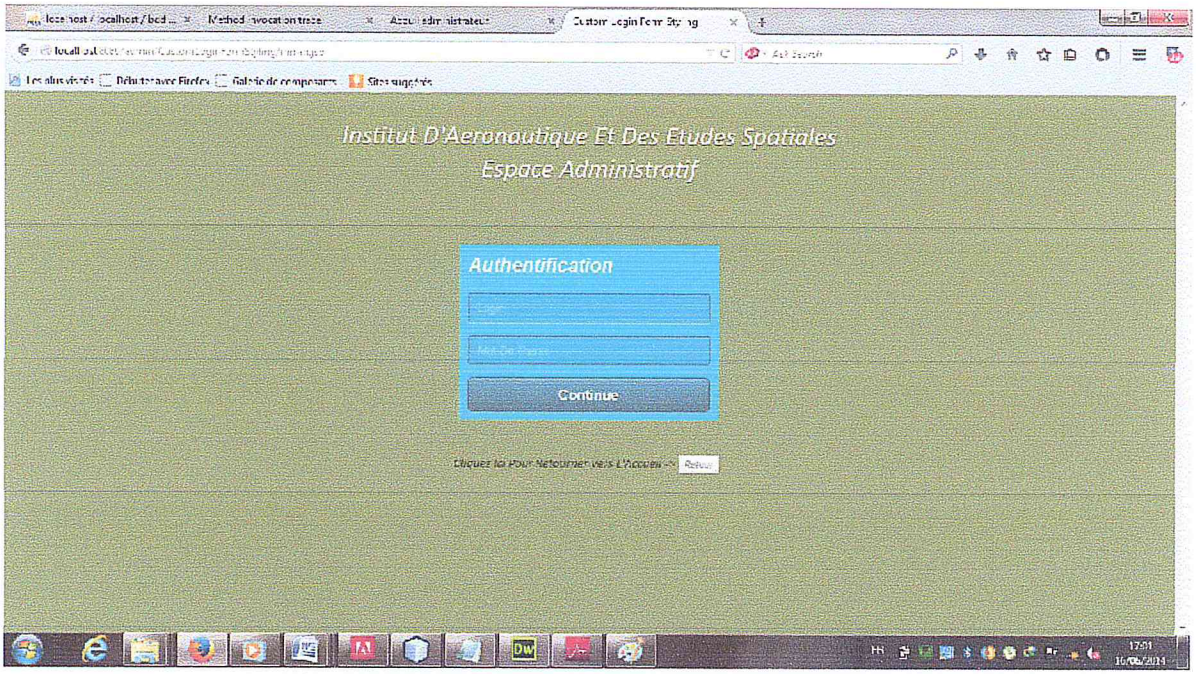


Figure V.3 « Page d'authentification »

V.4.3 Service Administration

La figure V.4 représente la fenêtre principale du service administration. L'administration peut effectuer les tâches figurant dans la fenêtre ci-dessous.

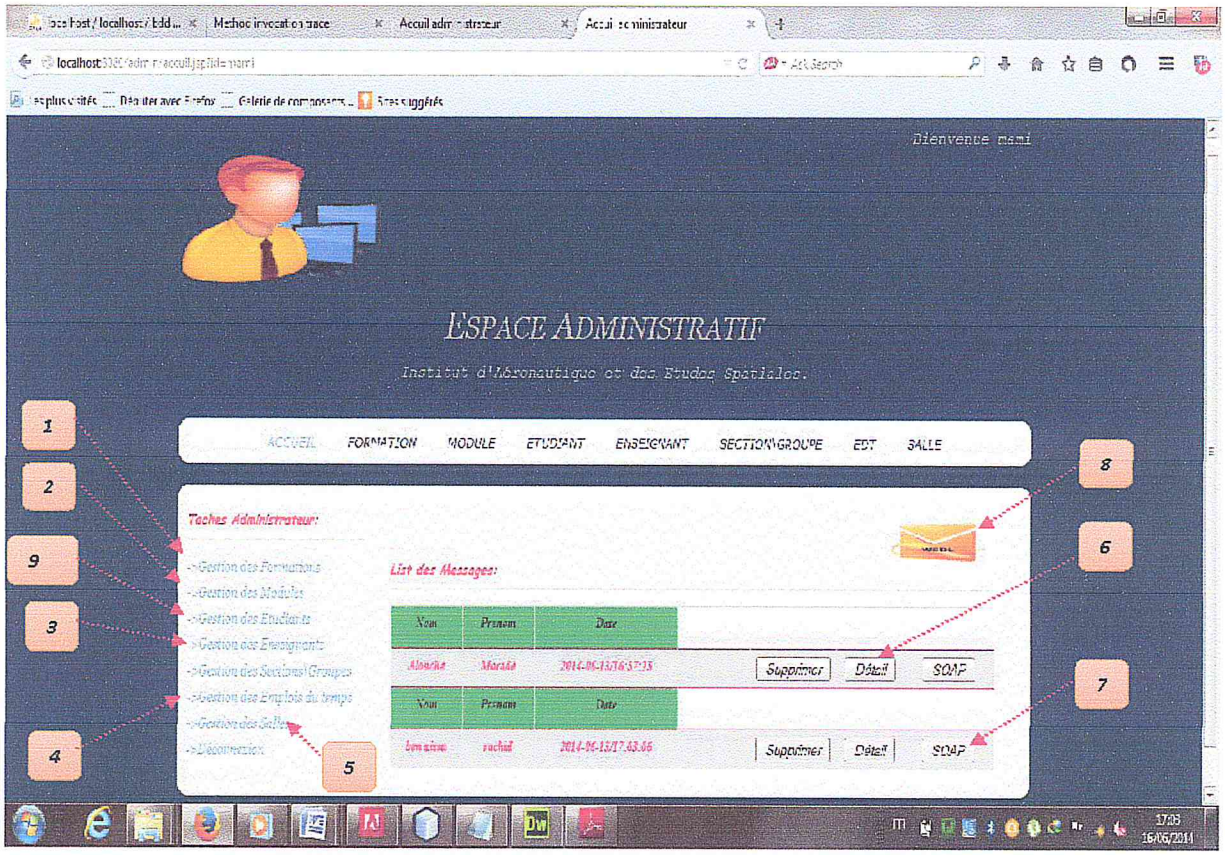


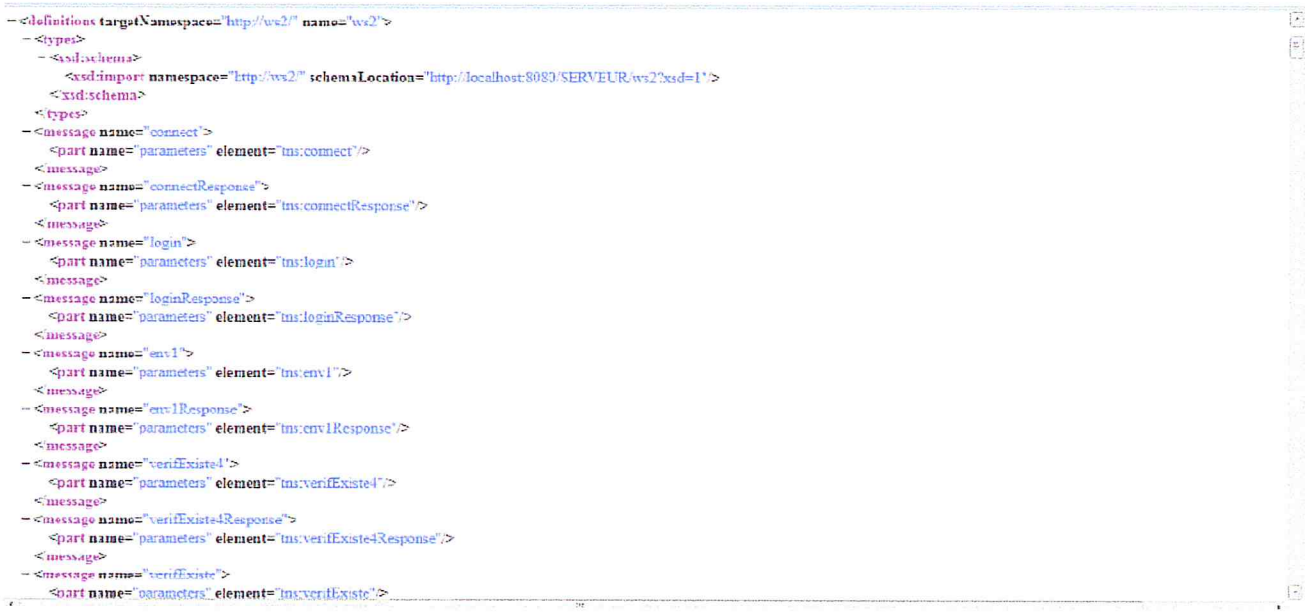
Figure V.4 « page principale du service administration »

Ces taches se résument en :

1. Gestion des formations.
2. Gestion des modules.
3. Gestion des enseignants.
4. Gestion et création des emplois du temps (EDT).
5. Gestion des salles.
6. Le détail des messages envoyés par les enseignants.
7. Le détail des messages envoyés par les enseignants sous forme enveloppe SOAP.
8. Le fichier WSDL du service administration.
9. Gestion des étudiants.

- **Fichier WSDL Service Administration**

La figure V.5 représente la structure du fichier WSDL du service Administration. Ce fichier décrit toutes les fonctionnalités et les opérations que peut effectuer le service administration ainsi que les paramètres de ce service. Il est le langage de description pour le service Web Administration.



```
<definitions targetNamespace="http://ws2/" name="ws2">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://ws2/" schemaLocation="http://localhost:8080/SERVEUR/ws2?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="connect">
    <part name="parameters" element="tns:connect"/>
  </message>
  <message name="connectResponse">
    <part name="parameters" element="tns:connectResponse"/>
  </message>
  <message name="login">
    <part name="parameters" element="tns:login"/>
  </message>
  <message name="loginResponse">
    <part name="parameters" element="tns:loginResponse"/>
  </message>
  <message name="env1">
    <part name="parameters" element="tns:env1"/>
  </message>
  <message name="env1Response">
    <part name="parameters" element="tns:env1Response"/>
  </message>
  <message name="verifExiste4">
    <part name="parameters" element="tns:verifExiste4"/>
  </message>
  <message name="verifExiste4Response">
    <part name="parameters" element="tns:verifExiste4Response"/>
  </message>
  <message name="verifExiste">
    <part name="parameters" element="tns:verifExiste"/>
```


Figure V.5 « Fichier WSDL Service Administration »

Le service Administration est accessible à travers l'URL :

« http://localhost:8080/Serveur/ws2 » définit par le port suivant :

```
<service name="ws2">
  <port name="ws2Port" binding="tns:ws2PortBinding">
    <soap:address location="http://localhost:8080/SERVEUR/ws2"/>
  </port>
</service>
```

- **Gestion et génération des emplois du temps**

Cependant, l'administrateur peut effectuer plusieurs taches (Figure V.4). Nous allons présenter une des plus principales taches de ce service qui se base sur la génération automatique des emplois du temps.

Pour effectuer la génération automatique des emplois du temps, l'administrateur clique sur l'onglet (4) « Gestion des emplois du temps » (voir figure V.4). Les opérations prévues sont comme suit :

1- Génération d'une population initiale :

On cliquant sur générer un nouveau emploi du temps, le service renvoi le formulaire représenté par la figure V.6. On saisit après la taille de la population initiale souhaitée et puis on clique sur le bouton générer (figure V.6).

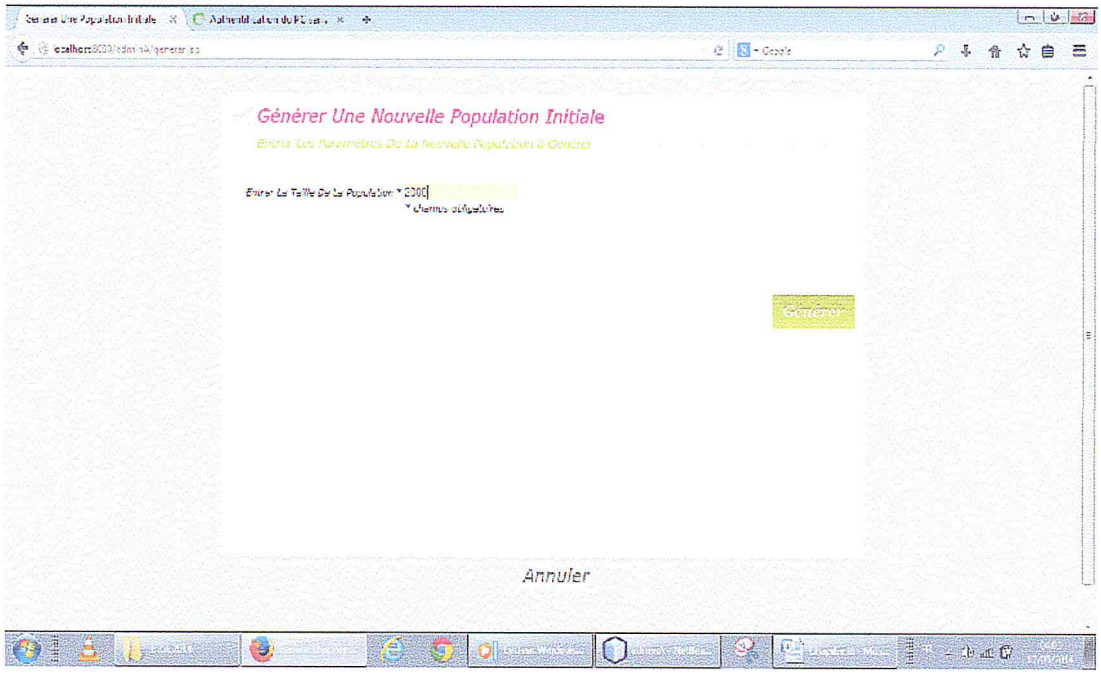


Figure V.6 « Génération d'une Population Initiale »

Pour générer la population initiale le service effectue une opération d'invocation « Invoquer » pour invoquer la méthode d'adaptation des algorithmes génétique pour générer cette population. Une fois que la population initiale est générée, le service renvoi un formulaire contenant un message de succès de la génération, le temps d'exécution, la population initiale brute (voir figure V.7) et codifiée ainsi que les paramètres AG pour la génération des EDT (emploi du temps).

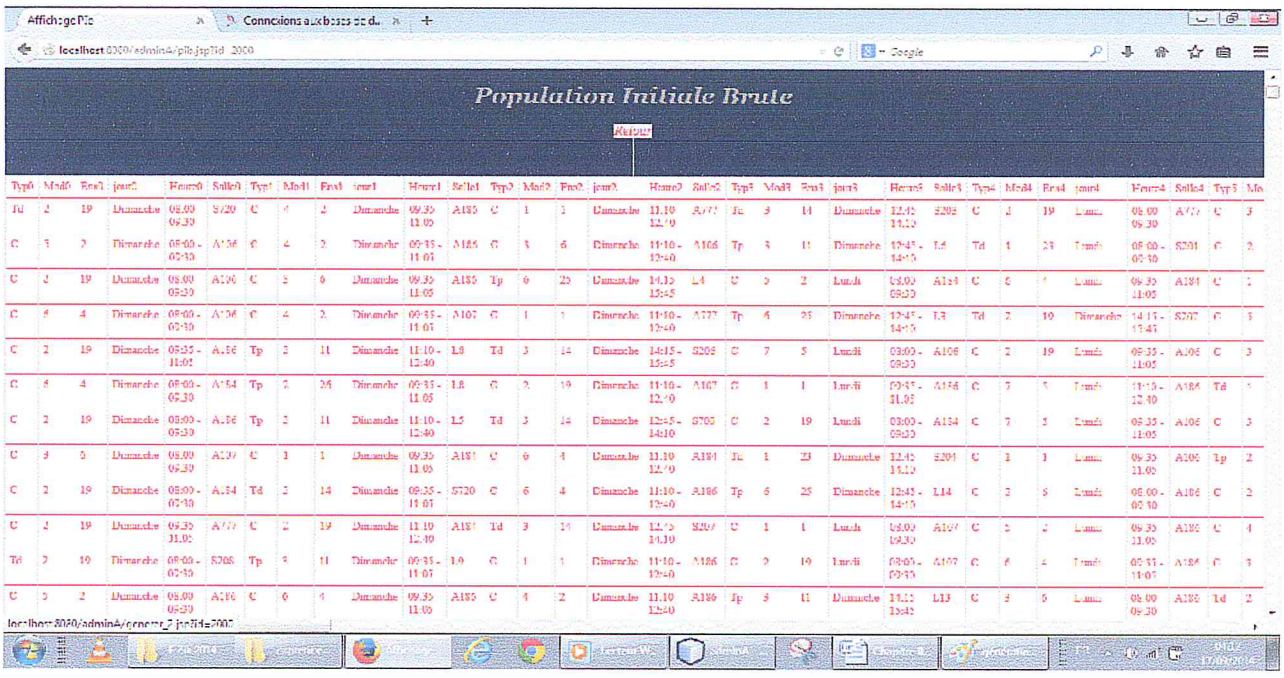


Figure V.7 « Exemple d'une Population Brute générée»

Une Classe java « gn » du service web Gn est crée pour la méthode d'adaptation de l'algorithme génétique. Ce service est définit par un ensemble d'opération web contenant les fonctionnalités de L'algorithme génétique.

```
@WebService(serviceName = "gn")
public class gn {


//*****************************************************************************
    //---------------------------POPULATION INITIALE----------------------------
    @WebMethod(operationName = "popGlobal")
    public void popGlobal(@WebParam(name = "name") String name)
    {
      //  System.out.println("BAKHTAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA");
        BDD.affichage_enseignants(AG.ENS);
        BDD.affichage_modules(AG.MOD);
        BDD.affichage_salles(AG.SAL);
        BDD.affichage_MTN(AG.MTN);
        BDD.affichage_EMT(AG.EMT);
        AG.TAILLEGLOBALE = Integer.parseInt(name);
        long T1,T2;

      T1 = System.currentTimeMillis();
     //  System.out.println("T1 : "+T1);
        AG.combinaison( AG.EMT, AG.MTN, AG.SAL, AG.ENS, AG.MOD );

         AG.Generation Population initiale( AG.PIBRUT, AG.PICODIF);
         BDD.SuppressionTable( AG.PIBRUT,"pib");
         BDD.creationRemplissage( AG.PIBRUT,"pib");
         BDD.InsertionLot( AG.PIBRUT,"pib");

         BDD.SuppressionTable( AG.PICODIF,"pic");
         BDD.creationRemplissage( AG.PICODIF,"pic");
         BDD.InsertionLot( AG.PICODIF,"pic");
```

2- Génération des emplois du temps :

A partir de la population initiale, l'administrateur peut générer des emplois du temps en invoquant le service Administration, qui lui à son tour, invoque la méthode et l'applique sur cette population pour générer le nombre désiré d'emplois du temps) d'une promotion comme l'illustre la figure V.8 suivante :

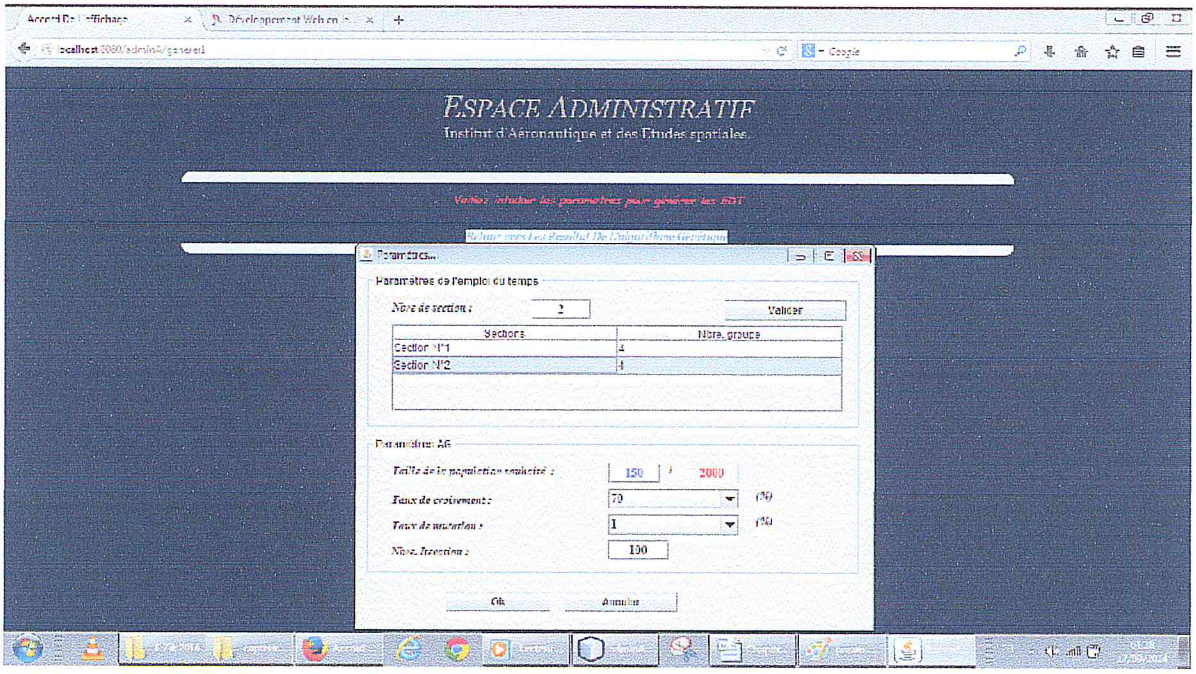


Figure V.8 « Paramètres de La méthode génétique»

En cliquant sur le bouton (OK), le service Administration effectue l'opération « Invoquer 2» pour invoquer la méthode d'adaptation afin de réaliser la génération des emplois du temps. Une fois que les emplois du temps sont générés, le service revoie à l'administrateur un formulaire représenté par la figure V.9 contenant le temps d'exécution de la génération automatique, le graphe de la fitness (voir l'exemple de la figure V.10) que la méthode calcule une fois qu'on obtient la solution optimale.

Le code source de l'opération « Invoquer 2» pour les paramètres de l'algorithme génétique est :

```
@WebMethod(operationName = "invoque2")
public int invoque2(@WebParam(name = "choix") boolean choix) {
    int a=0;
 if(choix==true)
         {
     try { a=1;
       java.lang.Runtime rt = java.lang.Runtime.getRuntime(); Process proc = rt.exec ("C:\\EDT2014\\param.bat");
        choix = false;
         }
     catch (IOException st) {st.printStackTrace();}
         }
 return a;
         }
```

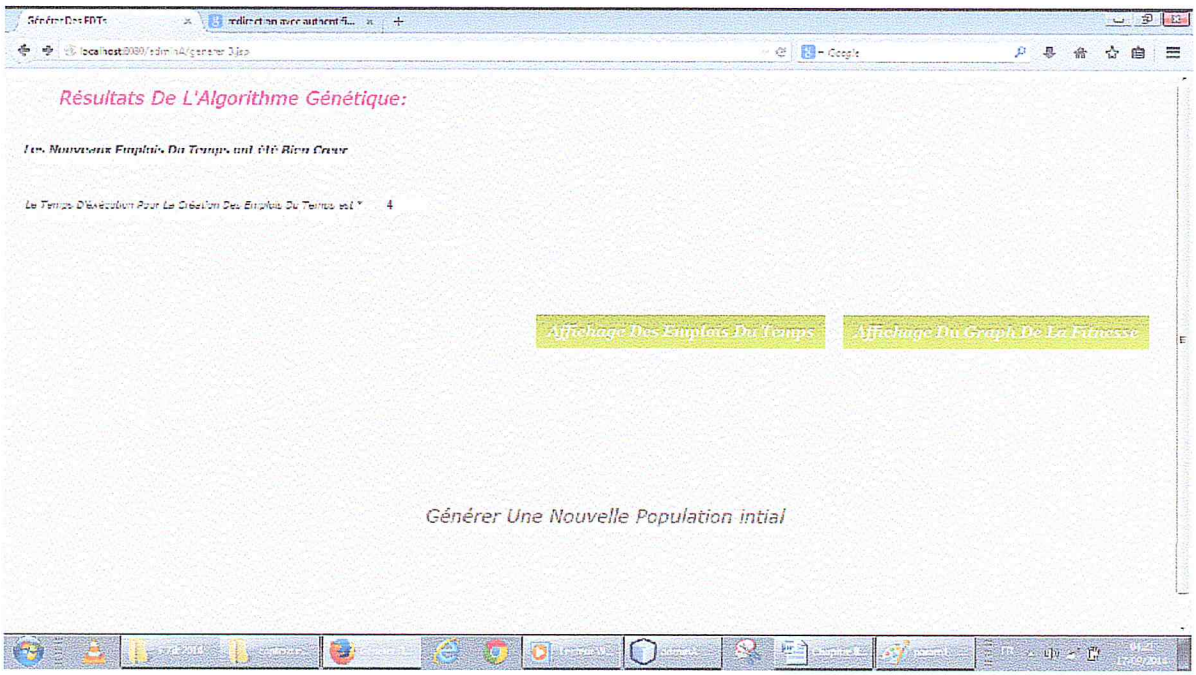


Figure V.9 « Résultats de l'Algorithme génétique»

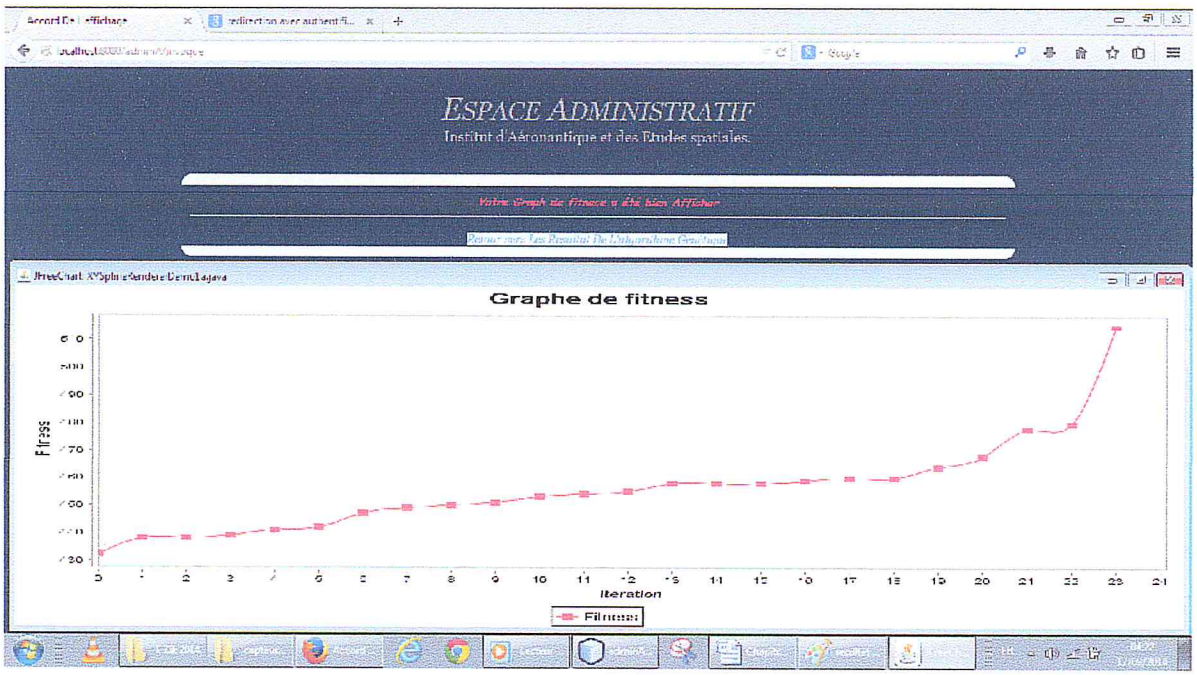


Figure V.10 « Graphe de Fitness»

3- Affichage des emplois du temps :

Le service Administration affiche les emplois du temps générés automatiquement comme l'illustre cette figure V.11 qui représente un exemple d'un emploi du temps généré automatiquement selon la section et le groupe choisi. On voit qu'on peut aussi afficher les cours seulement d'une section ou bien les séances de Td ou TP d'un groupe.

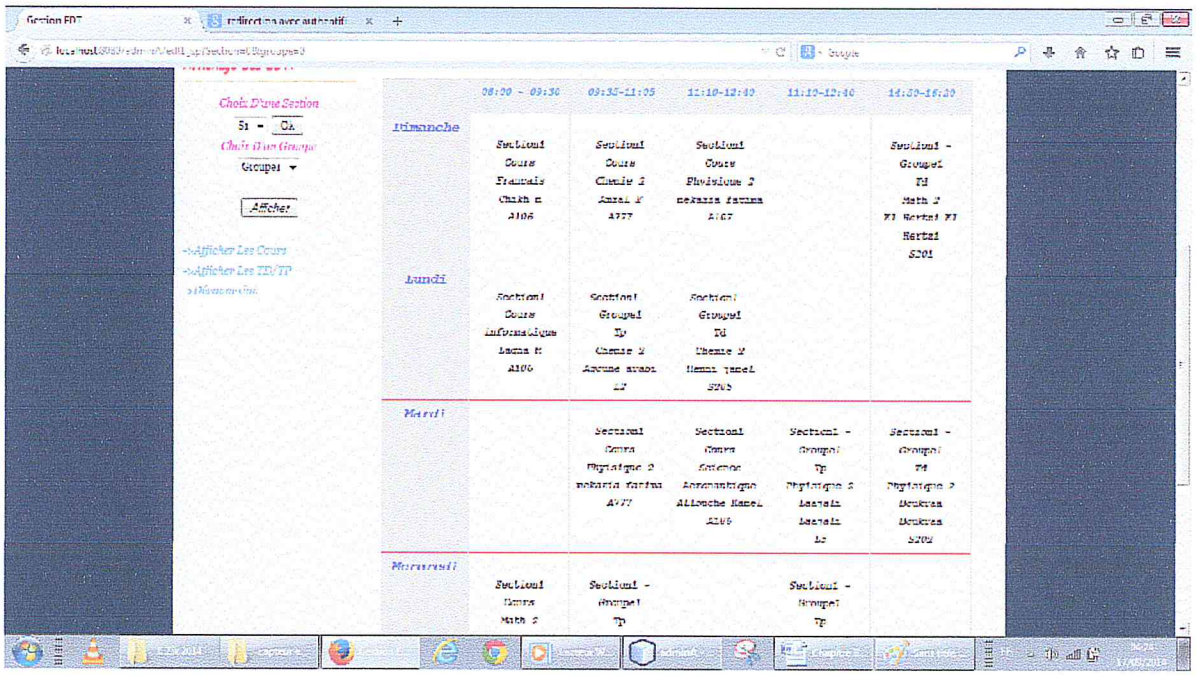


Figure V.11 : « Affichage des emplois du temps»

V.4.4 Service enseignant

Le service enseignant permet à l'enseignant d'effectuer les tâches suivantes représentées dans la figure (V.12) :

1. Consultation des emplois du temps des enseignants.

2. Messagerie : Boite de réception des messages envoyés par le service administration.

3. Contact : Le service administration pour une mise à jour des séances.

La figure ci-dessus représente l'affichage de l'emploi du temps d'un enseignant. L'enseignant invoque le service Enseignant pour consulter son emploi du temps généré automatiquement par le service Administrateur. Comme il peut aussi suggérer des modifications de ces séances en contact auprès du service Administrateur.

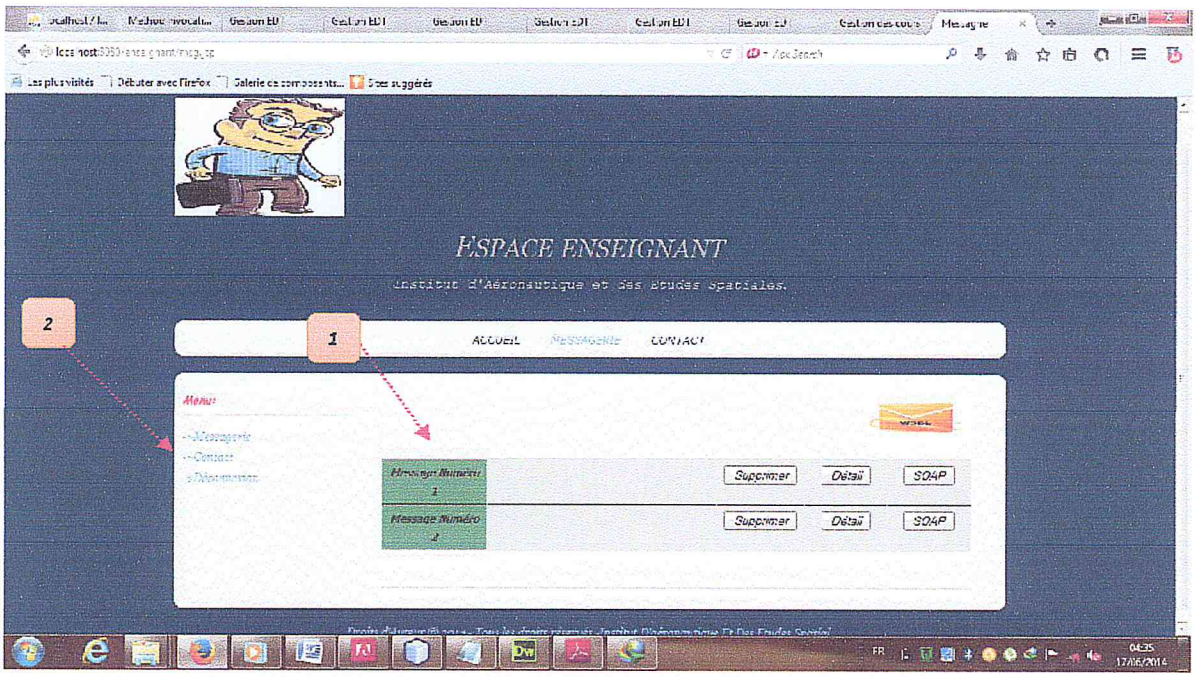


Figure V.12 « Emploi du temps Enseignant »

V.4.5 Service étudiant

Ce service permettre aux étudiant après avoir étaient authentifiés de consulter leurs emplois du temps selon leurs niveaux.

La figure V.13 représente le service étudiant de notre l'application avec l'emploi du temps affiché.

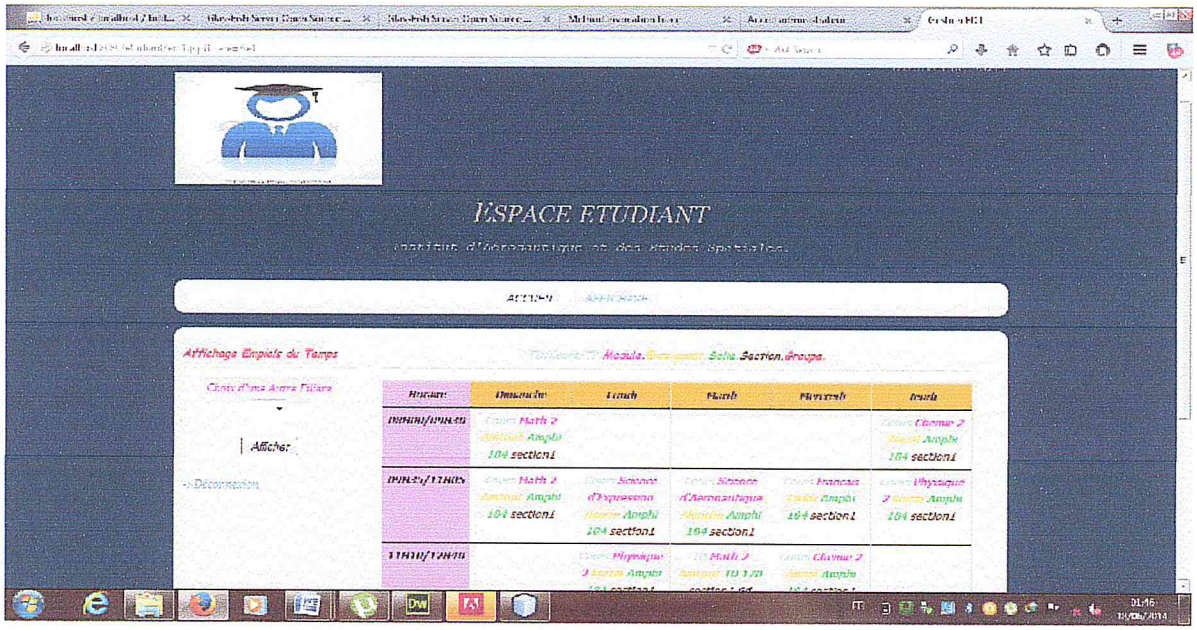


Figure V.13 « Page du service étudiant »

V.4.6. Invocation de la méthode d'adaptation des algorithmes génétiques

Dans l'étape de la génération automatique des emplois du temps, le service Administrateur invoque notre méthode d'adaptation des algorithmes génétique. Il utilise l'opération « Invoquer » contenu dans le fichier WSDL comme le présente la figure V.14 suivante

Method parameter(s)

Type	Value
java.lang.String	100

Method returned

void : "**null**"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:popGlobal xmlns:ns2="http://gn/">
            <name>100</name>
        </ns2:popGlobal>
    </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:popGlobalResponse xmlns:ns2="http://gn/"/>
    </S:Body>
</S:Envelope>
```

Figure V.14 « Soap Request/Response »

Dans cette première partie de l'implémentation de notre application, on a décrit les différents services Web qui constituent notre système de gestion des emplois du temps. On a détaillé leurs implémentations tout en mettant en évidence les fonctionnalités de chaque service.

Dans la partie Implémentation et réalisation qui ce suit, nous discuterons sur les expériences réalisées et les résultats obtenus pendant l'adaptation de notre méthode des algorithmes génétique pour la génération automatique des emplois du temps.

V.5. Résultats de la méthode d'adaptation des Algorithmes Génétiques

L'approche décrite dans la section précédente du deuxième chapitre de ce mémoire a été développée avec JavaNetBeans sous Windows avec une machine qui a un processeur Intel Core et 2Go de RAM. Nous présentons ici les meilleurs résultats obtenus parmi toutes les exécutions réalisées.

Plusieurs expériences ont été réalisé avec les donnée réelles pour une nouvelle promotion de l'IAES de Blida septembre 2013 pour ajusté les paramètres de l'algorithme génétique.

Dans un premier temps, nous avons examiné le comportement de notre AG avec des données fixes et un changement proportionnel au niveau de la probabilité de croisement et dans un autre temps, nous avons fixé les données avec un changement au niveau de la probabilité de mutation, et on suit de cette manière dans plusieurs expériences le comportement de notre AG ainsi que l'évolution de notre solution optimale.

- **Données utilisées**

L'algorithme mis en place a été testé sur les données réelles pour une promotion 2013 de l'Institut d'Aéronautique et des Etudes Spatiales de Blida.

Pour les résultats suivants, on a utilisé une taille même globale de la population pour toutes les expériences, et un nombre fixe d'itération qui est égale à 1000 itérations à chaque expérience.

N° Exp	Taille de la Population Souhaité	Taux de Croissement %	Taux de Mutation %	Temps de réponse	Solution Optimale
1	100	70	5	5	192
2	100	70	9	6	251
3	100	80	5	7	244
4	200	75	5	13	255
5	200	75	5	13	269
6	200	75	9	13	253
7	200	85	5	15	271

8	500	90	5	39	296
9	500	80	5	35	287
10	700	70	5	41	307
11	1000	70	5	59	294
12	1000	90	10	75	306

Tableau V.5 : « Données utilisées et résultats »

Le tableau V.5 représente les données et les résultats obtenus après 12 expériences en modifiant les différents paramètres pour voir en clair le comportement de notre AG dans le but d'améliorer la solution optimale et le temps d'exécution. Les résultats de ce tableau sont représentés par des figures explicatives qui nous permettent de bien expliquer le comportement de l'AG et l'évolution de notre solution optimale (voir figures V.17 et figure V.18).

- **Représentation des résultats**

Voici un exemple (figure V.16) .d'un emploi du temps généré par la méthode des algorithmes génétiques

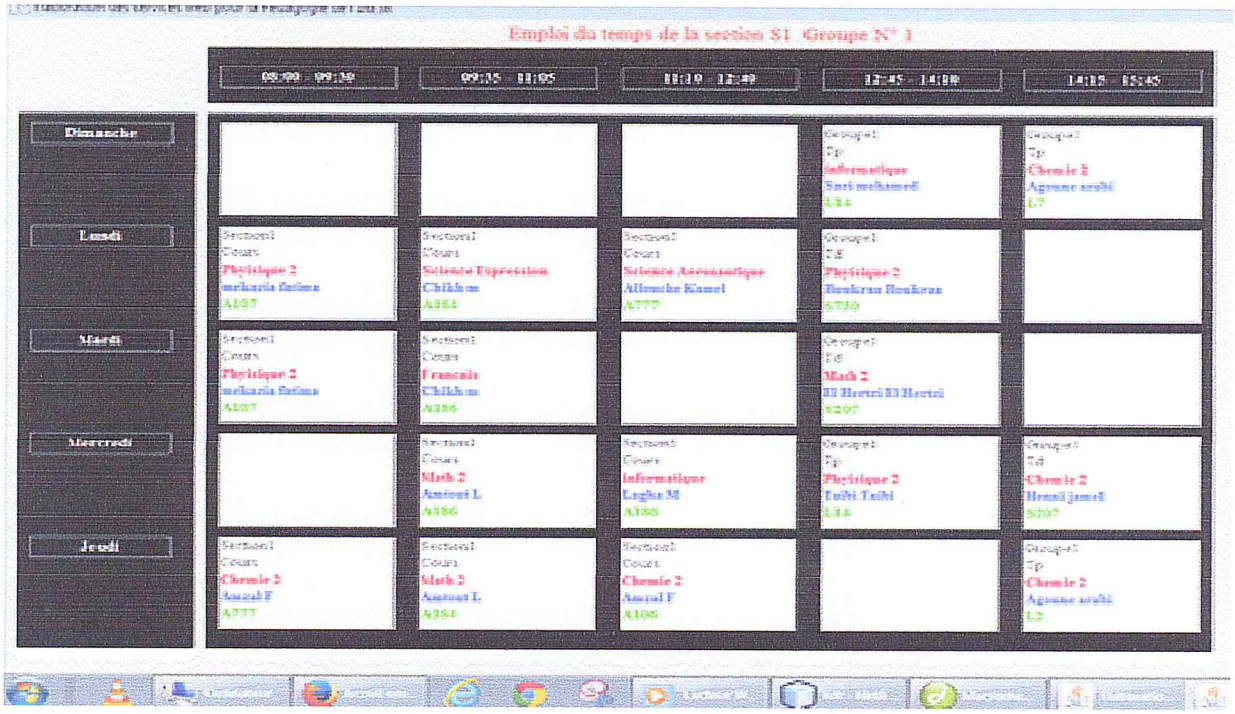


Figure V.16 : « Interface de la Méthode d'adaptation »

La figure V.16 représente un exemple d'un emploi du temps généré automatiquement par notre méthode d'adaptation, on voit que les contraintes durs sont satisfaire à partir de cet exemple et que c'est valable pour tous les autres emplois du temps.

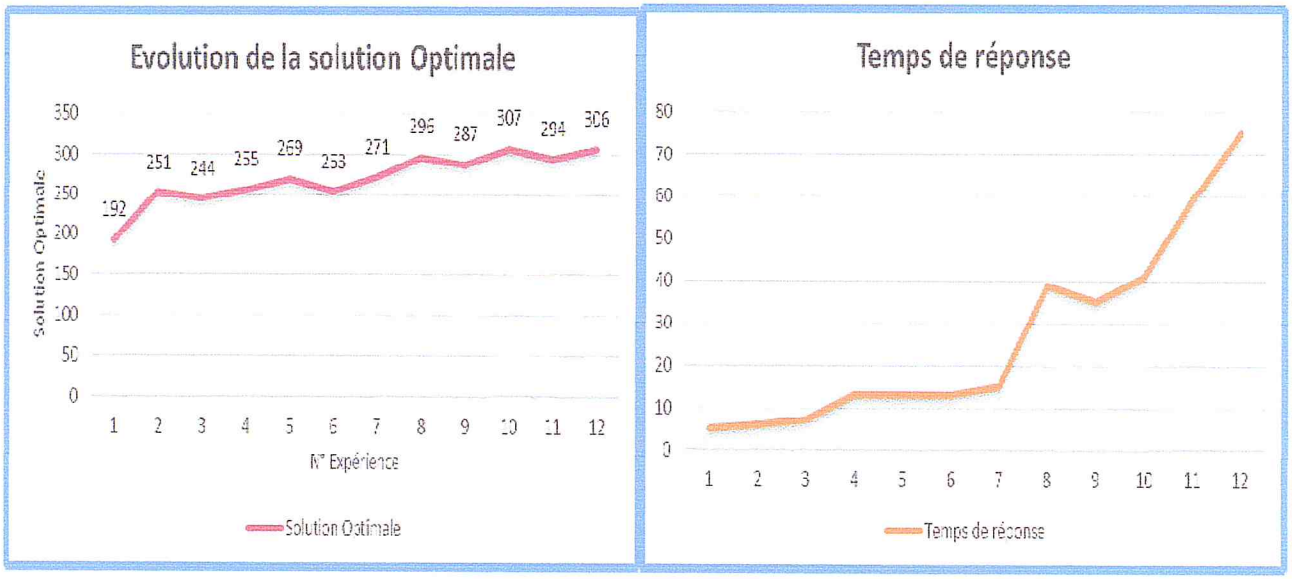


Figure V.17 : « Temps de réponse / Solution Optimale »

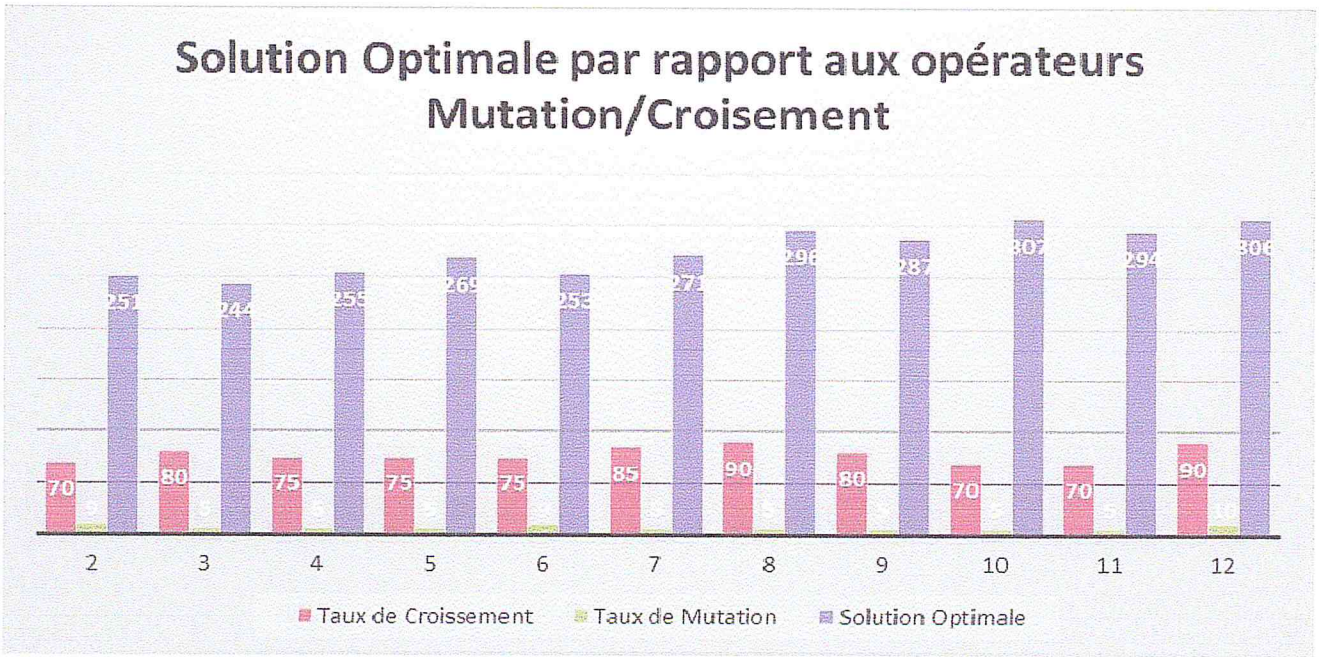


Figure V.18 : « Solution Optimale/Mutation/Croisement »

La solution optimale peut être amélioré à chaque fois qu'on renforce notre algorithmes par un taux de croisement et taux de mutation important, mais qui prend un temps de réponse assez important surtout qu'on on manque de ressources (salles, enseignants, Amphi, etc...).On note aussi que, les algorithmes génétiques sont des méthodes évolutives et non exactes, et ça été prouvé par les différentes expériences, en testant notre algorithme génétique avec les même entrées et on ressort à chaque fois avec une solution et un résultat différent (voir figure V.18).

V.6 Conclusion

Les problèmes liés aux emplois du temps sont à la fois intéressants, difficiles, et surtout différents d'une institution à une autre. Cependant, Il est rare d'aboutir à un emploi du temps qui fasse l'unanimité auprès de toutes les personnes concernées. Nous avons remarqué qu'il est relativement difficile d'établir un horaire qui garantisse à la fois un programme compact pour les enseignants et de satisfaire des contraintes qui varient d'un établissement à un autre.

Il est délicats de conclure une étude de ce type, les résultats obtenus sont très encourageants et nous prouvent encore une fois que les algorithmes génétiques sont aptes à résoudre des problèmes très complexes (NP Complet) comme les emplois du temps.

Dans ce chapitre nous avons décrit l'implémentation d'une part de nos services web et de l'autre, 14implémentation de l'algorithme génétique avec des contraintes pour résoudre le problème de la génération des emplois du temps.

Conclusion générale & perspectives

L'objectif de notre travail a été de réaliser des services web pour la génération automatique des emplois du temps destinée à la pédagogie de l'Institut d'Aéronautique et des Etudes Spatiales de BLIDA. Le premier service web a pour but la création et la génération des emplois du temps et la gestion des mises à jour, tandis que le deuxième permettra aux enseignants de l'institut de consulter et de suggérer des modifications des séances de leur emploi du temps et le dernier permet aux étudiants de l'institut de consulter les différents emplois du temps de l'institut.

Afin d'aboutir à cet objectif, on a présenté dans la première partie les trois chapitre de notre état de l'art ; le premier sur les concepts de base des services web ainsi que les principaux standards. Le deuxième chapitre, on a abordé la composition des services web. Et dans le dernier, sur l'optimisation combinatoire pour le problème de la génération automatique des emplois du temps.

Dans la deuxième partie de notre mémoire, on a abordé le volé conception et implémentation ; Dans le premier chapitre conception, on a décrit les différents services web qui constituent notre système de gestion des emplois du temps, et la conception de notre méthode d'adaptation des algorithmes génétique pour la génération automatique des emplois du temps. Le deuxième chapitre de cette partie est consacré d'une pat pour la réalisation et l'implémentation de nos services web tout en mettent en évidence les fonctionnalités de chaque service, et d'autres part à l'implémentation de notre méthode d'adaptation des algorithmes génétiques afin de résoudre le problème d'optimisation dans les emplois du temps.

Au terme de ce travail, les services web élaborés permettent :

- La création des emplois du temps d'une manière automatique.
- L'affichage des emplois du temps de chaque formation.
- La gestion des souhaits des enseignants concernant la mise à jour de leurs emplois du temps.

Après le déploiement de nos services web, on vise à ajouter quelques perspectives suivantes :

- Amélioration de la méthode d'adaptation en ajoutant des contraintes relâchées à côté des contraintes dures.
- Permettre la synchronisation et la communication entre ces trois services web pour produire un seul service web.

Dans notre système, l'interaction entre les services a bien existée mais pas la composition. C'est au cours de notre travail, qu'on a essayé d'améliorer notre système avec l'utilisation du principe de collaboration entre les services et la composition des services web. Mais, des problèmes

de logiciels et de version nous ont rendu la tâche assez difficile pour pourvoir y implémenter le fichier BPEL qui nous permettra la composition synchrone entre les trois services vu que la version 7.2 de NetBeans sur laquelle on a développé notre application ne reconnait pas le plugin « Soa BPEL ». Il est utile aussi de préciser que le cadre de notre travail est assez large, il couvre beaucoup de domaine, ce qui nous a empêchés de finir cette tâche. La composition n'est pas prise en compte dans nos objectifs mais plutôt un point qu'on penserait utile pour l'amélioration de notre système composé de services Web. On espérait bien qu'il y'aurai une suite de travaux dans ce cadre.

Références bibliographiques

[Adamidis et al, 1999] P.Adamidis, P.Arapakis. Evolutionary Algorithms in Lecture Timetabling. In Proceedings of the 1999 Congress on Evolutionary Computation. Washington, 1151 Vol. 2, 1999

[Adil, 2010] Adil Kenzi. Ingénierie des Systèmes Orientés Services Adaptables : Une Approche Dirigée par les Modèles. Thèse de doctorat, Université Mohamed V, Maroc. 2010.

[Albertella 2009] D. Albertella, Les avantages compétitifs de SOA : aspects techniques, organisationnels et financiers est-ce que SOA remplit ses promesses ? , Université de Fribourg, Suisse, 2009.

[Alonso et al., 2004] Alonso, G. Casati, F. Kuno, H. Machiraju, V. Web Services: Concepts, Architectures and Applications. Springer, 2004, 354p.

[Antonio, 2006] Antonio Bucchiarone, Stefania Gnesi. A Survey On Services Composition Languages and Modeles. Institut Discienza e Technologie dell'Informazione, Piza, Italie, 2006.

[Arenaza 2006] Nerea Arenaza. Composition semi-automatique des Web services. Projet de Master, Ecole Polytechnique Fédérale de Lausane, Février 2006.

[Barros et al., 2005b] A. Barros, M. Dumas, P. Oaks. Standards for Web Service Choreography and Orchestration: Status and Perspectives. In: Procs of the 3 rd International Conference the Business Process Management (BPM 2005), 1 st International Workshop on Web Service Choreography and Orchestration for Business Process Management, Nancy, France, Septembre 2005, pp.1-15.

[Bejaoui 2008] R. Bejaoui, SOA : les impacts sur le cycle de vie du projet de développement logiciel. Maîtrise en informatique de gestion, Université de Québec à Montréal, 2008.

[Ben 08] Ben-delhoum soheib, Hadji brahim. Gestion des emplois du temps par les Systèmes Multi-Agents. Mémoire d'ingénieur, ESI (Ex INI) Alger, Promotion 2007/2008.

[Ben Halima 2009] Riadh Ben Halima, Conception, implantation et expérimentation d'une architecture en bus pour l'auto réparation des applications distribuées à base de services web, Thèse de Doctorat, Université de TOULOUSE, 14 Mai 2009.

[Benatallah et al., 2002] B. Benatallah, M. Dumas, M.-C. Fauvet, F.A. Rabhi, Q.Z. Sheng. Overview of Some Patterns for Architecting and Managing Composite Web Services. ACM SIGecom Exchanges, 2002, vol.3, n°3, pp.9-16.

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

[Bertino 2004] A. Bertino, M. Keidl, et E. Kemper, A Framework for Contextaware Adaptable Webservices, EDBT, LNCS 2992, Springer-Verlag Berlin Heidelberg, 2004.

[Bieberstein 2008] Norbert Bieberstein, Keith Jones robert, et Tilak Mitra, Executing SOA: a Practical Guide for the Service-Oriented Architect, édition IBM Press, Mai 2008.

[Boukhadra 2011] Adel Boukhadra, La composition dynamique des services Web sémantiques à base d'alignement des ontologies owl-s, 2011.

[Brown 2008] Paul C. Brown, Implementing SOA: Total Architecture in Practice, edition Addison Wesley Professional, April 2008.

[Claro 2006] Daniela Barrairo Claro. SPOC - Un canevas pour la composition automatique de services web dédiés à la réalisation de devis, Thèse de Doctorat de l'université d'Angers, Octobre 2006.

[Cerami 2002] Ethan Cerami, Web Services Essentials, édition O'Reilly, Février 2002.

[Chab 2007] Chabane Refes, les services Web, édition SoftDeath, 2007.

[Chappell 2002] David Chappell, et Tyler JEWELL, Java Web Service, édition O'Reilly, Mars 2002.

[Chinnici 2004] Roberto Chinnici et Martin Gudgin, Web Services Description Language, édition W3C, le 22 Aout 2004.

[Chouchani 2010] Imad Chouchani, Utilisation d'un algorithme génétique pour la composition de service web, Université du Quebec, Montréal, mai 2010.

[Collette 2002] Y.Collette and P.Siarry. Optimisation Multiobjective. Eyrolles, 2002.

[Devaux 2008] C. Devaux, Urbanisation et SOA : Quelques bonnes pratiques pour leur mise en œuvre, Livre Blanc, Aubay, France, 2008.

[Dovey et al., 2005] Dovey, M., Kostadinov, I., Giddy, J., Green, P., Berry, D., Chonan, D., Wang, X. UK Engineering Tasks Force Evaluation of UDDI for UK e-Science. UK Technical Report, 27 May 2009.

[Dustdar et Schreiner 2005] S. Dustdar, W. Schreiner. A Survey on Web Services Composition. InWeb and Grid Services, Vol.1, 2005.

[Erben 1995] W. Erben and J. Keppler. A Genetic Algorithm Solving a Weekly Course-Timetabling Problem, Selected papers from the First International Conference on Practice and Theory of Automated Timetabling, Edinburgh, U.K. pp. 283-295, 1995.

[Erl 2008] Thomas Erl, SOA Principles of Service Design, édition prentice hall, juillet 2008.

[Fakhfakh 2006] Kaouthar Fakhfakh, Composition de protocoles métier de services web pour le passage à l'échelle en utilisant les automates, édition Dunod, 22 juin 2006.

[Fielding et al, 1999] R.T. Fielding, Gettys. J. Mogul, J. Berners-lee. Hyper Text Transfert Protocol (HTTP). RFC 2616, 1999.

[**Gardien 2002**] Georges Gardien, XML des bases de données aux Services Web, Édition Dunod, 2002.

[**Goldberg 1989**] D.E.Goldberg. Algorithmes Génétiques : Exploration, optimisation et apprentissage automatique, Addison-Wesley France, 1989.

[**Gregory 2009**] Gregory Le Bonniec. <PBEL> Orchestration de Web Services. Zenika, 26 novembre 2009.

[**Hao et al, 1999**] J. K. Hao, P. Galinier et M. Habib. Méthaheuristiques pour l'Optimisation Combinatoire et l'Affectation Sous Contraintes, Revue d'Intelligence, vol. 13, pp. 283-324, 1999.

[**Helga 2007**] Helga Duart-Amaya. Conevas Pour la composition de services web avec propriétés transitionnelles. Thèse PHD, Université de Joseph Fourrier, Grenoble I, Novembre 2007.

[**Hubert 2003**] Hubert Kadima, Valérie Monfort. LES WEB SERVICES (technique, démarche et outils XML, WSDL, SOAP, UDDI, Rosetta, UML).Mars 2003.

[**Jennings 2007**] Frank Jennings, Ramesh Loganathan et Poornachandra Sarang, Approach to Integration; XML, Web services, ESB, and BPEL in real-world SOA projects, edition Packt Publishing Ltd, November 2007.

[**Josuttis 2007**] Nicolaï Josuttis, SOA in Practice, édition O'Reilly, Septembre2007.

[**Kaabi 2008**] R. S. Kaabi, Une approche méthodologique pour la modélisation intentionnelle des services et leur opérationnalisation, Thèse de Doctorat, Université Paris 1, Sorbonne, 2008.

[**Karova 2004**] M.Karova. Solving Timetabling Problems Using Genetic Algorithms. 27th international Spring seminar on Electronics Technology, 2004.

[**Kenzi et al., 2009**] Adil Kenzi, B. El Asri, M. Nassar, A. Kriouile. Engineering Adaptable Service Oriented Systems: A Model Driven approach, IEEE International Conference on Service-Oriented Computing and Applications, Taipei, IEEE Computer Society Taiwan. 2009.

[**Kreger 2001**] Heather Kreger, Web Service Conceptual Architecture, edition IBM Software Group, Mai 2001.

[**Kulchenko 2001**] Pavel Kulchenko, James Snell et Doug Tidwell, Programming Web Services with SOAP, édition O'Reilly, Décembre 2001.

[**Lopez-Velasco 2008**] Céline Lopez-Velasco, Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation, Thèse de Doctorat, Université JOSEPH FOURIER, 18 novembre 2008.

[**Margolisand 2007**] Ben Margolisand, et Joseph Sharpe. SOA for the Business Developer: Concepts, BPEL, and SCA, édition MC Press, October 2007.

[**Melliti 2004**] Tarek Melliti. Interopérabilité des Services Web complexes, Thèse de Doctorat, Université Paris IX Dauphine, le 8 Décembre 2004.

[Mili 2010] Rédha MILI. Conception et implémentation d'un algorithme d'optimisation pour la Génération automatique d'emploi du temps Universitaire. Mémoire de Magister, Université 20 Août 1955 Skikda, Promotion 2010.

[Newcomere 2004] Eric Newcomere, Understanding Web Services Xml WSDL SOAP and UDDI, edition O'Reilly, 2004.

[Osman et al., 2005] T. Osman, D. Thakker, and D. Al-Dabass. Bridging the Gap between Work flow and Semantic-based Web services Composition. In Proc of the Web Service Composition Workshop WSCOMPS05, 2005.

[Papazoglou 2003] M.P. Papazoglou, Web Services and Business Transactions, World Wide Web Internet and Web Information Systems, edition Kluwer Academic, 2003.

[Peltz 2003] C. Peltz. Web Services Orchestration and Choreography. IEEE Computer, juillet 2003, vol.36, n°10, pp.46-52.

[Ponge 2008] Julien Ponge. Model Based Analysis of Time-aware Web Services Interactions. Thèse de Doctorat, Université Blaise Pascal - Clermont-Ferrand II, France, 2008.

[Saka 1984] M.Sakarovitch. Programmation Discrète, Hermann, 1984.

[Schaerf 1999] A. Schaerf. A Survey of Automatic Timetabling. Artificial Intelligence Review, Springer Netherlands, vol. 13, no. 2, pp. 87-127, 1999.

[Schreiner 2005] Wolfgang Schreiner et Schahram tdar, A survey on Web services composition, Int. J. Web and Grid Services, 2005.

[Scott 2002] Scott Short, Construire des Services Web XML, édition Dunod, 2002.

[Sun et al., 2003] H. Sun, X. Wang, B. Zhou and P. Zou. Research and Implementation of Dynamic Web Services Composition. Springer-Verlag Berlin Hbidelberg, 2003, pp.457--466.

[Velasco 2008] Céline Lopez-Velasco. Sélection et composition de services web pour la génération d'application adaptées au contexte d'utilisation. Thèse de Doctorat, Université Joseph Fourrier, 2008.