République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master 2 en Informatique

Thème

SÉLECTION DES SERVICES WEB A BASE DE L'ALGORITHME D'ESPÉRANCE MAXIMISATION

Réalisé par :

-**Mr Chater Mohamed Amine**

Présenté le *22 Juin 2015 devant la commission d'examination composée de*

- *Mr. Hadjila Feth Allah* *(Encadreur)*
- *Md. Halfaoui Amel* *(présidente de jury)*
- *Mr.Merzoug Mohamed* *(Examinateur)*
- *Mr.Mouffak B* *(Examinateur)*

Année universitaire: 2015-2016

Dédicace

Je dédie ce mémoire à :

Ma feue grand-mère qui a fait de ma réussite une préoccupation et qui, depuis ma tendre enfance, n'a cessé d'investir financièrement, spirituellement pour ma formation et m'a encouragé à évoluer sur cette trajectoire. Grand-mère, ces mots ne sauraient suffire pour vous dire merci. Que votre âme repose en paix. Que Dieu ait votre âme dans sa sainte miséricorde et vous accorde son paradis éternel (amen).

Mon grand-père. Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous. Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Ce travail est le fruit des sacrifices que vous avez consentis pour mon éducation et ma formation.

Ma mère, qui a œuvré pour ma réussite, de par son amour, son soutien, tous les sacrifices consentis et ses précieux conseils, pour toute son assistance et sa présence dans ma vie.

Votre bonne éducation, vos conseils et vos bénédictions n'ont jamais fait défaut,

Vous m'avez soutenu, protégé et encouragé durant toutes ces années ; vous étiez toujours présente quand j'avais besoin de vous ; je n'aurais pu achever ce travail sans votre générosité et votre affection.

Recevez à travers ce travail aussi modeste soit-il, l'expression de mes sentiments et de mon éternelle gratitude.

Que Dieu le tout puissant soit à vos côtés, vous protège et vous accorde son paradis éternel (amen).

Mon père, en signe de reconnaissance de l'immense bien qu'il a fait pour moi concernant mon éducation qui aboutit aujourd'hui à la réalisation de cette étude. Soyez fier et trouvez ici le résultat de longues années de sacrifices et de privations qui m'ont aidé à avancer dans la vie. Puisse Dieu faire en sorte que ce travail porte son fruit ; Merci pour les valeurs nobles, l'éducation et le soutien permanent venu de

vous. Je vous dois ce que je suis aujourd'hui et ce que je serai demain et je ferai toujours de mon mieux pour rester votre fierté et ne jamais vous décevoir.

Dieu le tout puissant soit à vos côtés et vous accorde une meilleure santé.

Ma grande sœur

Tu n'as pas cessé d'être pour moi un exemple de persévérance, de courage et de générosité.

Même si on a grandi ensemble, on a su forger notre propre personnalité, avant de suivre notre destin. Nous ne sommes pas d'accord sur tout, mais tellement de points en commun nous unissent. La vie nous trace des chemins, parfois opposés qui espacent nos moments ensemble. Mais les joies et les peines partagées, chaque merveilleux souvenir, passé ou à venir, font que tu auras toujours une place de choix dans mon cœur. La vie m'a fait un très beau cadeau en faisant de toi ma Sœur.

Ma petite sœur, je te souhaite une bonne continuation. Malgré nos 14 ans d'écarts on arrive tout de même à s'entendre.

Les olives sont amères sur la langue mais changent de goût une fois dans l'arrière. J'apprendrai à assumer mon rôle de frère, mes responsabilités en avant, et des fois, mes droits après.

Je sais que je t'ai fait parfois souffrir, que tu m'en veux mais j'aimerais vraiment me racheter. Je te souhaite un avenir plein de joie, de bonheur, et de réussite.

Tous mes oncles et tantes qui m'ont toujours soutenu et épaulé dans la vie ainsi que tout le long de mon cursus scolaire.

Mes professeurs qui doivent voir dans ce travail la fierté d'un savoir bien acquis

Remerciements

Nulle œuvre n'est exaltante que celle réalisée avec le soutien moral et financier des personnes qui nous sont proches.

Mes remerciements s'adressent d'abord à ALLAH le tout puissant et à son prophète MOHAMED (paix et salut sur lui) pour les chances qui m'ont été offertes pour réaliser ce travail.

Je tiens à remercier mon professeur encadreur Mr HADJILA Fethallah, qui m'a accordé sa confiance en acceptant de diriger ce mémoire, malgré les multiples occupations qui lui sont infligées.
Votre ouverture d'esprit et surtout l'intérêt que vous portez à la science font de vous une source intarissable à laquelle tout étudiant devrait s'abreuver. Votre gentillesse, votre modestie, votre riche expérience et l'accueil cordial que vous m'avez toujours réservé m'ont inspiré une grande admiration à votre égard. Trouver ici le témoignage de ma profonde gratitude et de mes sincères remerciements.

Je tiens à exprimer ma plus profonde reconnaissance :
A la Présidente du jury Mme HALFAOUI Amel et les honorables membres, Mr Merzoug Mohamed, Mr Mouffak B, à Mr le doyen de la faculté et le chef de département informatique. Je suis sensible à l'honneur que vous me faites en acceptant de siéger dans le jury pour juger la qualité de ce travail. Vos observations contribueront à son amélioration et à l'ouverture de nouvelles pistes de recherche.
Soyez rassurés de ma profonde gratitude et de mes sentiments respectueux.
Mes remerciements s'adressent également à mon cher ami et conseiller, Mr Mosbah Réda du ministère des télécommunications, pour ses précieuses collaborations.

A tous mes professeurs du Département d'Informatique pour leurs disponibilité et conseils.

A mon père et à ma mère qui m'ont toujours entouré et motivé sans cesse à devenir meilleur.

A mes sœurs qui m'ont assisté dans ces moments difficiles et m'ont servi d'exemple.

A mes tantes, oncles, cousins et cousines, paternels et maternels.

A mes amis et collègues qui n'ont cessé de m'encourager.

Mes remerciements s'adressent également à tous ceux qui, de près ou de loin, ont apporté leur contribution à la réalisation de ce travail, je vous prie de trouver l'expression de ma profonde reconnaissance.

Résumé

La technologie des services web joue un rôle très important dans l'interopérabilité et l'intégration des applications.

Il arrive très fréquemment que de nombreux services répondent à u n même ensemble de besoins fonctionnels : pour réaliser la réservation d'un vol, un client peut mettre en concurrence plusieurs compagnies aériennes. Ces services se distinguent les uns des autres par leurs propriétés non fonctionnelles. Nous nous intéressons ici à les comparer selon des critères de QOS (réputation, fiabilité, etc.).

Pour alléger la tâche de sélection faite par les utilisateurs, nous devons proposer une approche qui instancie des compositions de services abstraites avec des composants concrets, tout en assurant les besoins suivants : les critères de QOS associés à la combinaison de services recherchée doivent être optimisés, en plus les contraintes globales de l'utilisateur doivent être satisfaites.

Mots clés : Sélection de web services, QOS, optimisation combinatoire, clustering, EM.

Abstract

The web services technology plays a very important role in the interoperability and the integration of application. Many services meet the same set of functional requirements, but differ in QOS: for instance, to book a flight, a customer may consult several airline companies. These services are distinguished from each other by their non-functional properties. We are interested in comparing

them according to quality of service criteria (reputation, reliability, etc.).

to ensure the selection aim, we propose an approach that instantiates the abstract composition with concrete components, while ensuring the following needs: the QOS criteria of the service composition should be optimized, in addition to that , global constraints of the user must be met.

Keywords: *selection de web services, QOS, combinatorial optimization, clustering, EM.*

ملخص

تكنولوجيا خدمات الشبكة تلعب دورا هاما جدا في العمل المشترك والتكامل بين المنظمات التطبيقية. العديد من الخدمات تلبي نفس المجموعة من المتطلبات الوظيفية، وتتميز هذه الخدمات عن بعضها البعض بممتلكاتهم الغير وظيفية. ونحن مهتمون في مقارنتها وفقا لمعايير الجودة (السمعة، والموثوقية، وما إلى ذلك).

لحل هذه المشكلة من طرف المستعملين لابد من استعمال طريقة للتخيير بين هذه الخدمات و ذللك حسب متطلبات المستعمل دون التفريط في معايير الجودة

الكلمات المفتاحية: تخيير خدمات الويب, المركب المثالي

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

1- Contexte

De nos jours, le Web n'est plus simplement un énorme entrepôt de texte et d'images, son évolution a fait qu'il est aussi un fournisseur de services. La notion de "service Web" désigne essentiellement une application mise à disposition sur Internet par un fournisseur de services, et accessible par les clients à travers des protocoles Internet standard. Par essence, les services Web sont des composants logiciels autonomes et auto-descriptifs et constituent par ce fait un nouveau paradigme pour l'intégration d'applications.

Actuellement, les services Web sont mis en œuvre au travers de trois technologies standards : WSDL, UDDI et SOAP. Ces technologies facilitent la description, la découverte et la communication entre services. Cependant, cette infrastructure de base ne permet pas encore aux services Web de tenir leur promesse d'une gestion largement automatisée. Cette automatisation est pourtant essentielle pour faire face aux exigences de passage à l'échelle et de la volonté de réduire les coûts de développement et de maintenance des services. Fondamentalement, elle doit s'accommoder d'un moyen pour décrire les services Web d'une manière compréhensible par une machine.

2- Problématique

Avec la croissance explosive du nombre de services web publiés sur Internet, il est difficile de choisir ceux susceptibles de satisfaire les besoins d'un quelconque client, en matière de qualité de service. Par ailleurs, la plupart des clients obligent à recourir à une combinaison de plusieurs services web, ce qui rend alors la tâche plus difficile encore.

On est obligatoirement amené à procéder à une sélection.

L'objectif de ce travail consiste donc à optimiser au maximum cette sélection afin de répondre aux préférences du client tout en respectant une contrainte spécifique qu'il impose.

3- Contribution

Pour solutionner le problème cité plus haut on a pensé à un apprentissage automatique appelé **espérance maximisation** (EM)

L'algorithme d'espérance maximisation est intéressant et très puissant (application multiple), qui a été implémenté dans plusieurs domaines :

- Pour compléter des données manquantes
- Modelé de mélange

Pour ce qui est des paramètres du modèle, il est possible d'obtenir de bons résultats avec une bonne précision.

Les caractéristiques globales sont donc bien reproduites par l'algorithme.

Nous proposons dans ce mémoire d'étendre l'algorithme d'espérance maximisation pour l'optimisation de la sélection des services web.

Cette approche se divise en deux étapes :

Étape E : Évaluation de l'espérance selon les données observées et les paramètres à notre disposition.

Étape M : Maximisation de cette espérance

4- Plan de travail

Le présent mémoire se compose de deux chapitres :

Premier chapitre

Tout d'abord on a commencé par définir la notion des services web ainsi que leur historique.
Ensuite on les a détaillés du point de vue intérêt, caractéristiques, et les standards qui leur sont associés
On a pensé également aux protocoles d'envoi de messages et au fonctionnement des services web
Enfin on a fini notre recherche en citant les différentes couches technologiques, les avantages et inconvénient de ces services sans omettre la notion de qualité de service (**QOS**)

Deuxième chapitre 2

Ce chapitre a été complétement réservé à l'algorithme d'espérance maximisation **EM.**
Partant de la définition, historique et principes de fonctionnement, on a fini le chapitre par une expérimentation globale en passant par les variantes usuelles d'EM, son organigramme, quelques exemples d'application de l'algorithme, sans oublier ses avantages et inconvénient et sa conception.

Chapitre I

I. Introduction

Les organisations sont de plus en plus nombreuses à se tourner vers des architectures à base de services Web pour le développement et l'intégration d'applications ou de systèmes d'information. L'importance des standards dans ce contexte a sans doute accentué le phénomène.
Il arrive très fréquemment que de nombreux services répondent à u n même ensemble de besoins fonctionnels : pour réaliser la réservation d'un vol, un client peut mettre en concurrence plusieurs compagnies aériennes. Ces services se distinguent les uns des autres par leurs propriétés non fonctionnelles. Nous nous intéressons ici à les comparer selon des critères de qualité (réputation, fiabilité, etc.).
Nous supposons que les services web sont groupés dans des clusters (ou *communautés*) qui représentent un ensemble de services de même fonctionnalité. Une communauté offre à ses utilisateurs. A partir de ces clusters l'utilisateur final cherche à créer une combinaison de services, tels que chaque élément de cette combinaison appartient à une communauté donnée, en plus l'utilisateur veut optimiser les critères de QOS associés à cette combinaison et préserver un ensemble de contraintes globales.

II. Les services web

1. Définition service web

Il s'agit d'une technologie permettant à des applications de dialoguer à distance via Internet, et ceci indépendamment des plates-formes et des langages sur lesquels elles reposent. Pour ce faire, les services Web s'appuient sur un ensemble de protocoles Internet très répandus (XML, HTTP), afin de communiquer. Cette communication est basée sur le principe de demandes et réponses, effectuées avec des messages XML.

Les services web sont décrits par des documents WSDL (Web Service Description Language), qui précisent les méthodes pouvant être invoquées, leurs signatures et les points d'accès du service (URL, port.). Les services Web sont accessibles via SOAP, la requête et les réponses sont des messages XML transportés sur HTTP.
Il existe probablement autant de définitions des Web Services que d'entreprises qui les créent, mais presque toutes ces définitions ont ceci en commun :

- Les Web Services proposent aux utilisateurs du Web des fonctionnalités pratiques grâce à un protocole Web standard (dans la plupart des cas, le protocole utilisé est SOAP).

- Les Web Services offrent un moyen de décrire leurs interfaces suffisamment en détail pour permettre à un utilisateur de créer une application cliente capable de converser avec eux cette description et généralement fournie dans un document XML nommé WSDL (Web Services Description Language).
- Les Web Services sont inscrits afin que les utilisateurs potentiels puissent les trouver facilement. Ceci est possible grâce à UDDI (Universal Discovery Description and Integration).

2. L'intérêt d'un Service Web

Les services Web fournissent un lien entre applications. Ainsi, des applications utilisant des technologies différentes peuvent envoyer et recevoir des données au travers de protocoles compréhensibles par tout le monde.

Les services Web sont **normalisés** car ils utilisent les standards XML et HTTP pour transférer des données et ils sont compatibles avec de nombreux autres environnements de développement. Ils sont donc indépendants des plates-formes.

C'est dans ce contexte qu'un intérêt très particulier a été attribué à la conception des services Web puisqu'ils permettent aux entreprises d'offrir des applications accessibles à distance par d'autres entreprises. Cela s'explique par le fait que les services Web n'imposent pas de modèles de programmation spécifiques. En d'autres termes, les services Web ne sont pas concernés par la façon dont les messages sont produits ou consommés par des programmes. Cela permet aux vendeurs d'outils de développement d'offrir différentes méthodes et interfaces de programmation au-dessus de n'importe quel langage de programmation, sans être contraints par des standards comme c'est le cas de la plate-forme *CORBA* qui définit des ponts spécifiques entre le langage de définition IDl et différents langages de programmation. Ainsi, les fournisseurs d'outils de développement peuvent facilement différencier leurs produits avec ceux de leurs concurrents en offrant différents niveaux de sophistication.

Les services Web représentent donc la façon la plus efficace de partager des méthodes et des fonctionnalités. De plus, ils réduisent le temps de réalisation en permettant de tirer directement parti de services existants.

3. Caractéristiques d'un service Web

La technologie des services Web repose essentiellement sur une représentation standard des données (interfaces, messageries) au moyen du langage XML. Cette technologie est devenue la base de l'informatique distribuée sur Internet et offre beaucoup d'opportunités au développeur Web.

Un service Web possède les caractéristiques suivantes :

- Il est accessible via le réseau.
- Il dispose d'une interface publique (ensemble d'opérations) décrite en XML.
- Ses descriptions (fonctionnalités, comment l'invoquer et où le trouver ?) sont stockées dans un annuaire.
- Il communique en utilisant des messages XML, ces messages sont transportés par des protocoles Internet (généralement HTTP, mais rien n'empêche d'utiliser d'autres protocoles de transfert tels : SMTP, FTP, BEEP...).
- L'intégration d'application en implémentant des services Web produit des systèmes faiblement couplés, le demandeur du service ne connaît pas forcément le fournisseur. Ce dernier peut disparaître sans perturber l'application cliente qui trouvera un autre fournisseur en cherchant dans l'annuaire.

4. Architecture des services web

a. Architecture de référence

Pour promouvoir l'interopérabilité et l'extensibilité du paradigme des Web services, une architecture de référence est nécessaire afin de préserver les objectifs initiaux visés par les Web services.
L'Architecture SOA [4] est un modèle abstrait qui consiste à diviser un logiciel répondant à un problème, en un ensemble d'entités proposant des services. Chacune de ces entités peut utiliser les services proposés par d'autres entités. On obtient ainsi un réseau de services interagissant entre eux [9].
L'architecture SOA vise trois objectifs importants [10] :
– Identification des composants fonctionnels.
– Définition des relations entre ces composants.
–Établissement 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 [11, 12] :
– Le fournisseur du service : Correspond à la personne ou à l'organisation propriétaire du service. D'un point de vue technique, il est constitué par la plateforme d'accueil du service;

– Le client : Correspond au demandeur du service. D'un point de vue technique, il est constitué par l'application d'invocation du service. L'application client peut être elle- même un Web service;

– L'annuaire des services : Il désigne l'entité logicielle qui joue le rôle de l'intermédiaire entre les clients et les fournisseurs de services. Il correspond à un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

Les interactions de base entre ces trois rôles incluent les opérations de publication, de découverte et d'invocation [12, 13]. Ce scénario est illustré par la **Figure 1.1**

– Publication : Le fournisseur de service définit la description de son service et la publie dans un annuaire de service en vue d'être localisé par des clients;

– Découverte : Le client utilise les facilités de recherche disponibles au niveau de l'annuaire pour retrouver et sélectionner un service;

– Invocation : Le client examine ensuite la description du service sélectionné pour récupérer les informations lui permettant de se connecter au fournisseur du service et d'interagir avec l'implémentation du service considéré.

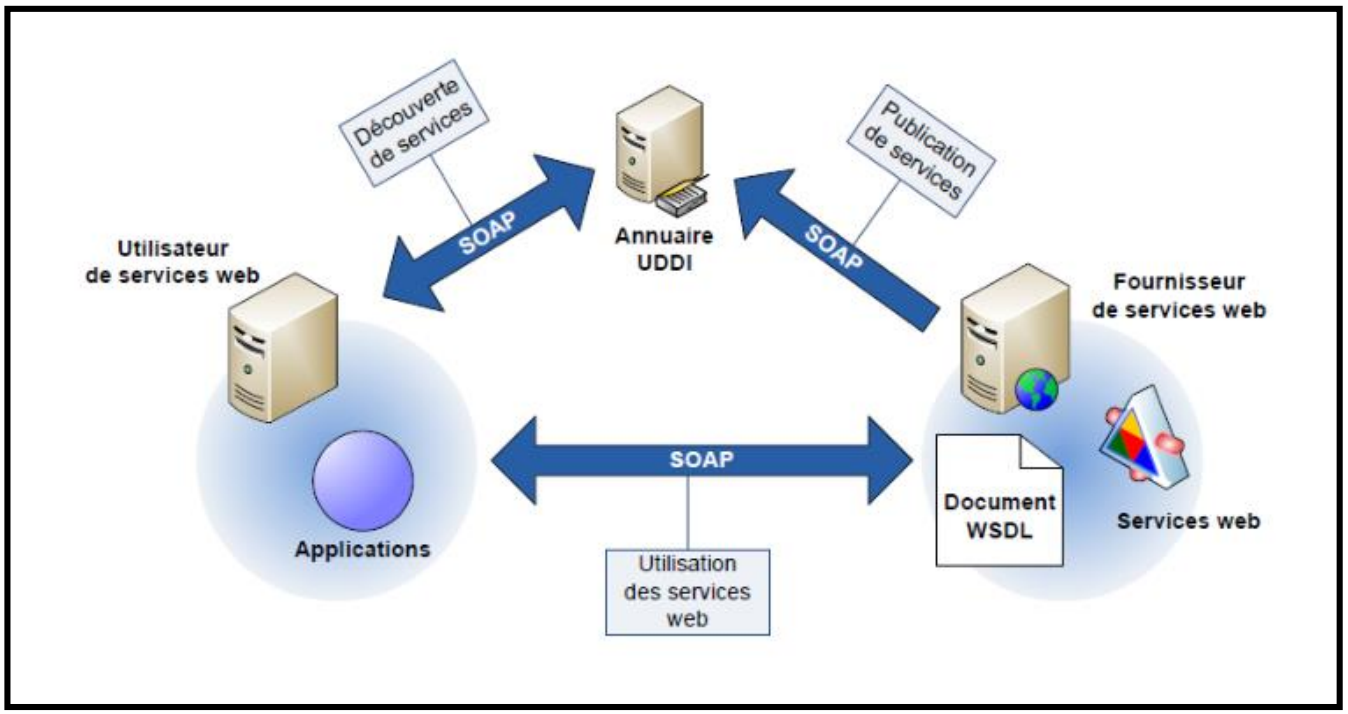


Figure 1.1 ***Architecture de référence***

Pour garantir l'interopérabilité des trois opérations précédentes, des propositions de standards ont été élaborées pour chaque type d'interaction. Nous citons, notamment les standards suivants :

– **SOAP** : Un protocole permettant d'invoquer à distance les opérations offertes par un Web service en utilisant des messages XML.

– **WSDL** : Un standard permettant de décrire l'interface d'un Web service sous la forme d'un fichier de description en XML.

– **UDDI** : Un protocole d'annuaire permettant à la fois de publier et de retrouver un Web service.

Cependant, cette architecture n'est pas suffisante pour permettre une utilisation effective des Web services. Il existe de nombreux nouveaux standards émergents (tels que les standards de sécurité, d'administration et du Web sémantique) dans le domaine des Web services que cette architecture ne peut pas aisément prendre en compte. Il s'avère donc nécessaire d'étendre l'architecture de base de Web services.

b. Architecture étendue

Cette architecture étendue est aussi appelée pile des Web services, du fait qu'elle est constituée de plusieurs couches se superposant les unes aux autres [12]. Elle utilise les couches standards de la première architecture en ajoutant au-dessus d'autres couches spécifiques.

La Figure 1.2 est une proposition de vue globale simplifiée de la pile des Web services [12]. Chaque couche de la pile répond à des préoccupations fonctionnelles différentes telles que la sécurité, les transactions, etc.

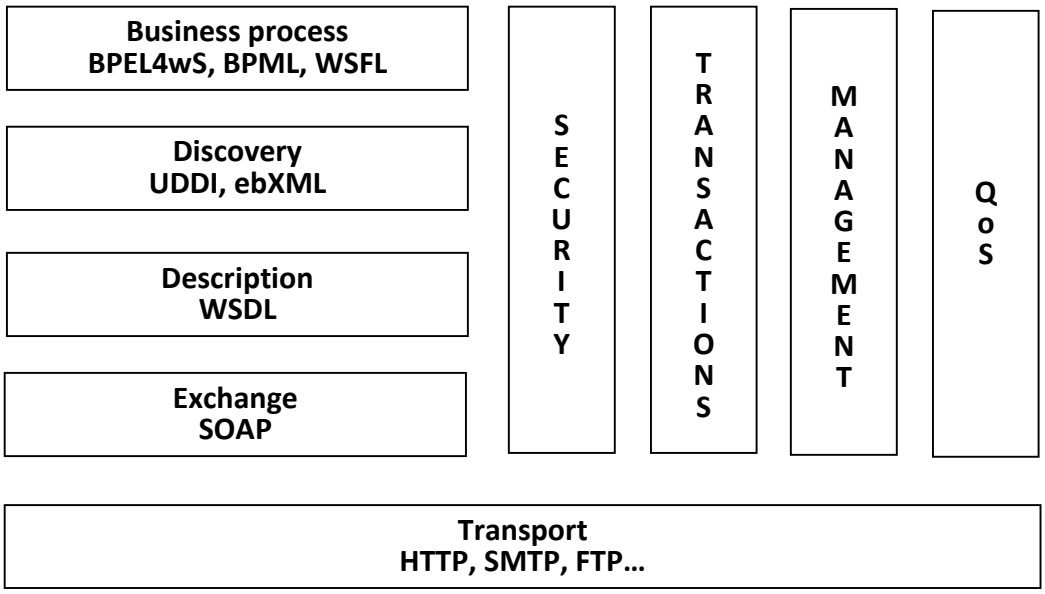


Figure 1.2 Architecture étendue

La pile est constituée de plusieurs couches, chaque couche s'appuie sur un standard particulier. On retrouve, au-dessus de la couche de transport (HTTP, SMTP), les trois couches formant l'infrastructure de base décrite précédemment.

D'après [15] cette architecture peut être décomposée en trois types de couches :

– **L'infrastructure de base :** Elle est constituée de trois couches, ces couches s'appuient sur les standards des Web services (SOAP, WSDL, UDDI). Elle définit le fondement technique de l'architecture de référence.

– **Les couches transversales :** [16] (sécurité, administration, ...) Ce sont les couches qui rendent viable l'utilisation effective des Web services dans le monde industriel et dont on trouve toute les notions associées aux problématiques de sécurité et celles en rapport avec la QoS. La partie administration est un peu particulière, car il s'agit de mettre en place des enchainements de services et par conséquent de créer des Web services composites. Des travaux tentent d'intégrer le Web sémantique dans ces couches transversales en ajoutant une couche verticale représentant le Web sémantique et étant utilisable par les quatre couches horizontales représentant les standards.

– **Business Process :** Cette couche permet l'intégration de Web services, elle établit la représentation d'un Business Process comme un ensemble de Web services. De plus, la description de l'utilisation de différents services composants ce service est disponible par l'intermédiaire de cette couche.

5. Standards associés aux services web

Les services Web reprennent la plupart des idées et des 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, REST, XML-RPC, SOAP et UDDI.

5.1 REST

REST (*Representational State Transfer*) est une architecture de services Web. Élaborée en l'an 2000 par *Roy Fielding*, l'un des créateurs du protocole HTTP, du serveur Apache HTTPd et d'autres travaux fondamentaux, REST est une manière de construire une application pour les systèmes distribués comme le World Wide Web.

• N'est pas un protocole ou un format, mais une architecture (celle de HTTP).

• Chaque « méthode » ou « service » est attaché à une URL.

• Consommer un Web Service REST revient à appeler une simple URL en http (Post ou Get), le serveur renvoie sa réponse, la plupart du temps en XML

• Très utilisé pour la communication entre machines.

Requête

```
http://ws.ct-goat.com/getCityInfos.asp ?uID=xxxxxxxxxxxxx&comID=562
```

Réponse (*Figure 1.3) (Figure 1.4)*

```
<RETURN>
  <ERROR>
    <NUMBER> [numéro d'erreur] </NUMBER>
    <DESCRIPTION> [description de l'erreur] </DESCRIPTION>
  </ERROR>
  <RESULT>
    <ROWS>
      <ROWCOUNT>1</ROWCOUNT>
      <ROW>
        <COM_ID> [ID de la commune] </COM_ID>
        <COM_NAME> [nom de la commune] </COM_NAME>
        <COM_COMINSEE> [code INSEE de la commune] </COM_COMINSEE>
        <COM_PAE> [Id des points d'arrêts principaux de la commune]</COM_PAE>
      <ROW>
    <ROWS>
  <RESULT>
<RETURN>
```

Figure 1.3 Réponse REST

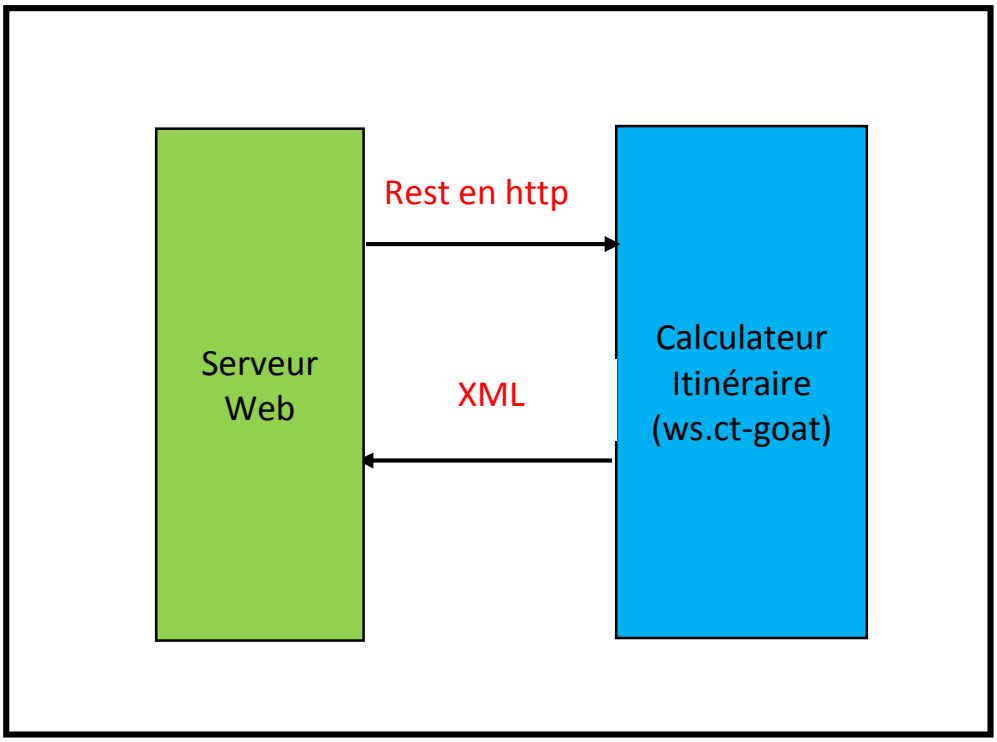


Figure 1.4

5.2 XML-RPC *Figure 1.5*

XML-RPC est un protocole simple utilisant XML pour effectuer des messages RPC. Les requêtes sont écrites en XML et envoyées via HTTP POST. Les requêtes sont intégrées dans le corps de la réponse HTTP. XML-RPC est indépendant de la plate-forme, ce qui lui permet de communiquer avec diverses applications. Par exemple, un client Java peut parler de XML-RPC à un Perl Server.

• XML-RPC permet d'appeler une fonction sur un serveur distant à partir de n'importe quel système (Windows, MacOSX, Linux) et avec n'importe quel langage de programmation. Le serveur est lui-même sur n'importe quel système et est programmé dans n'importe quel langage.

• Cela permet de fournir un Service Web utilisable par tout le monde sans restriction de système ou de langage.

• Les processus d'invocation à distance utilisent le protocole HTTP pour le transport des données et la norme XML pour le codage des données.

• XML-RPC est conçu pour permettre à des structures de données complexes d'être transmises, exécutées et renvoyées très facilement.

• XML-RPC est l'ancêtre de SOAP.

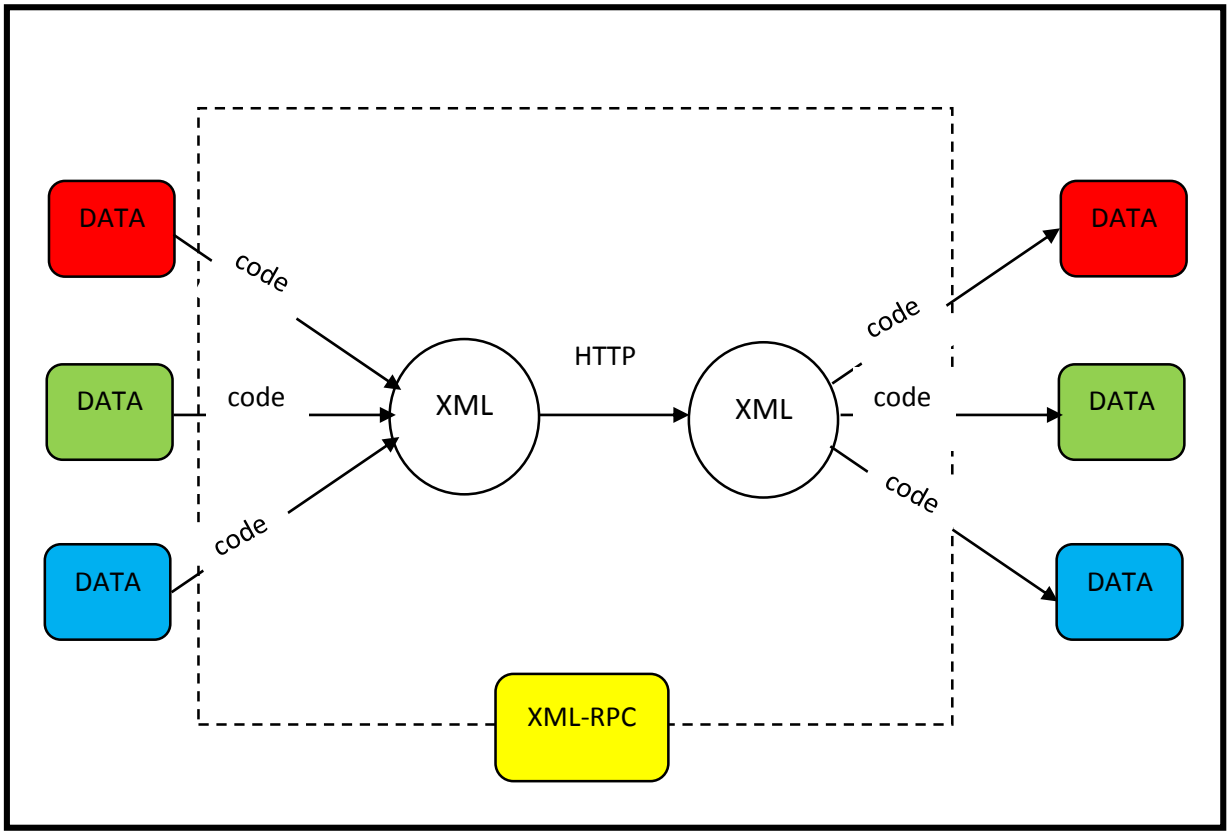


Figure 1.5 XML-RPC

Exemple *figure 1.6, figure 1.7*

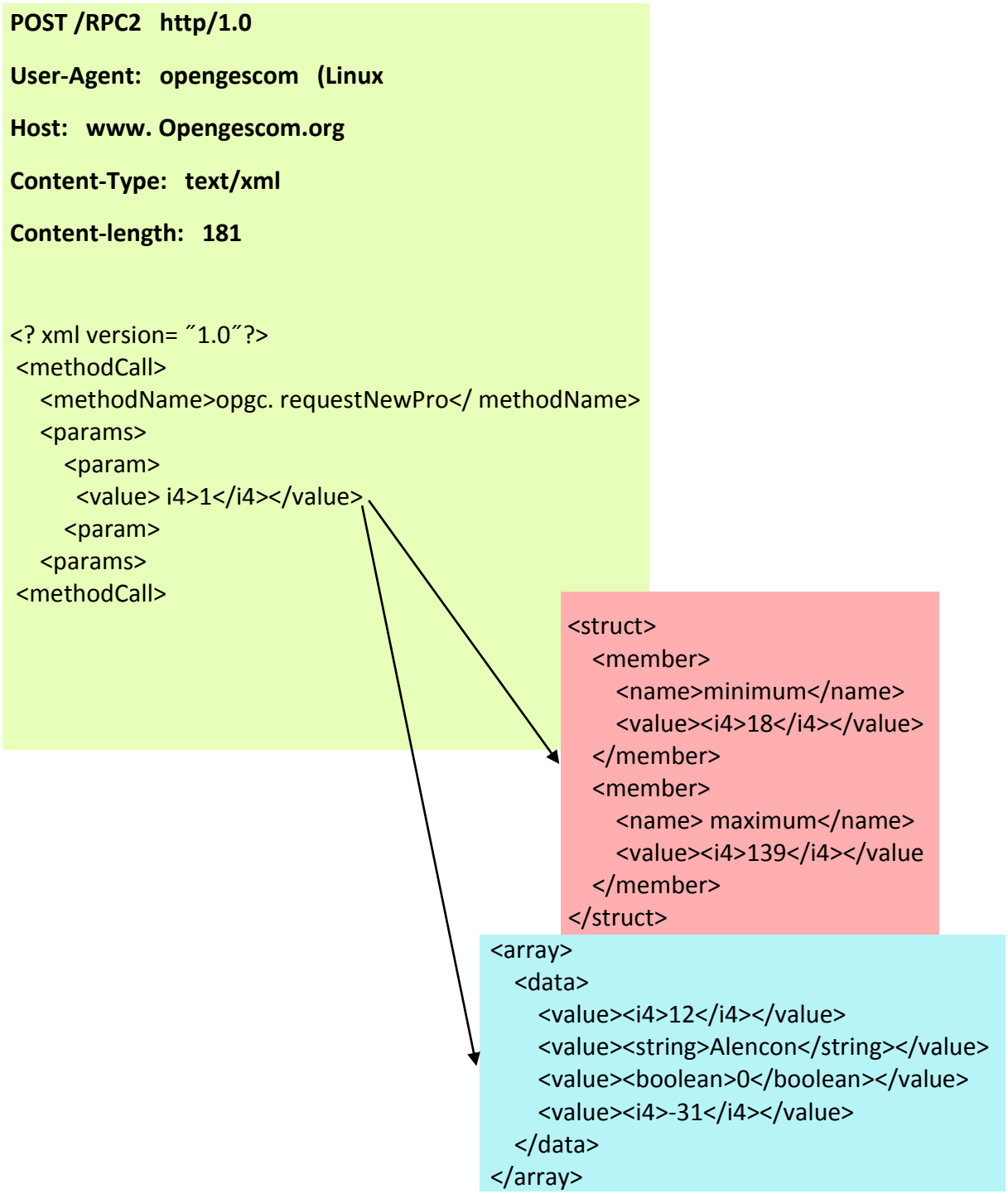


Figure 1.6 exemple xml-rpc

Réponse correcte

```
HTTP/1.1 200  OK
Connection:   close
Content-Length: 158
Content-Type: text/xml
Date: Tue, 22 Feb 2005 18:30:08 GMT
Server: OpenGesCom/1.0.1- Linux Debian

<? xml version=˝1.0˝?>
<methodResponse>
 <params>
  <param>
   <struct>
    <member>
     <name> code produit </name>
     <value><string>AB000010
</string></value>
    </member>
    <member>
     <name> code barre </name>
     <value><string>3270190113508
</string></value>
    </member>
   <struct>
  </param>
 </params>
</methodResponse>
```

Réponse en erreur

```
HTTP/1.1 200  OK
Connection:   close
Content-Length: 426
Content-Type: text/xml
Date: Tue, 22 Feb 2005 18:30:08 GMT
Server: OpenGesCom/1.0.1- Linux Debian

<? xml version=˝1.0˝?>
<methodResponse>
 <fault>
  <value>
   <struct>
    <member>
     <name> fault code </name>
     <value><int>4 </int></value>
    </member>
    <member>
     <name> faultString </name>
     <value><string>Too many
parameters</string></value>
    </member>
   </struct>
  </value>
 </fault>
</methodResponse>
```

Figure 1.7

5.3 SOAP

SOAP (*Simple Object Access Protocol*) **figure 1.8** est un protocole standard de communication. C'est l'épine dorsale du système d'interopérabilité. SOAP est un protocole décrit en XML et standardisé par le W3C. Il se présente comme une enveloppe pouvant être signée et pouvant contenir des données ou des pièces jointes.

Il circule sur le protocole HTTP et permet d'effectuer des appels de méthodes à distance.

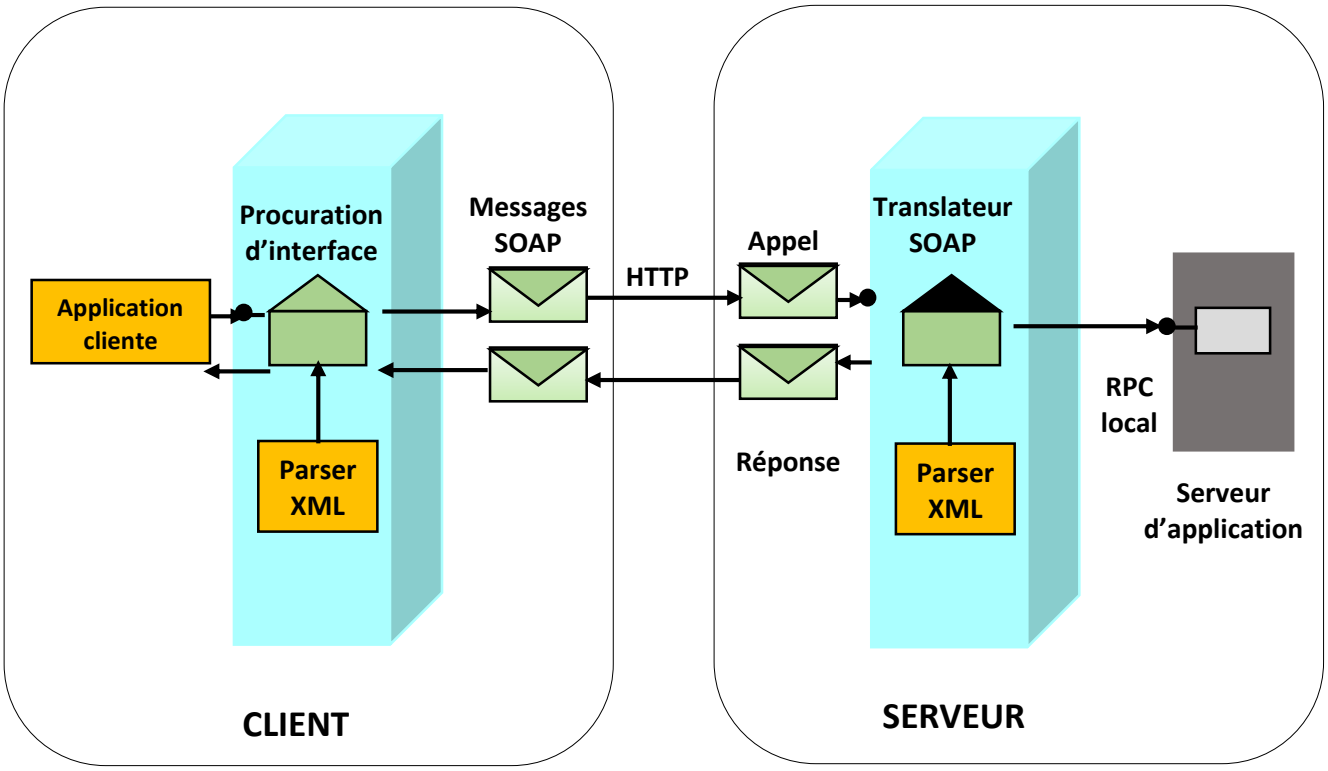


Figure 1.8

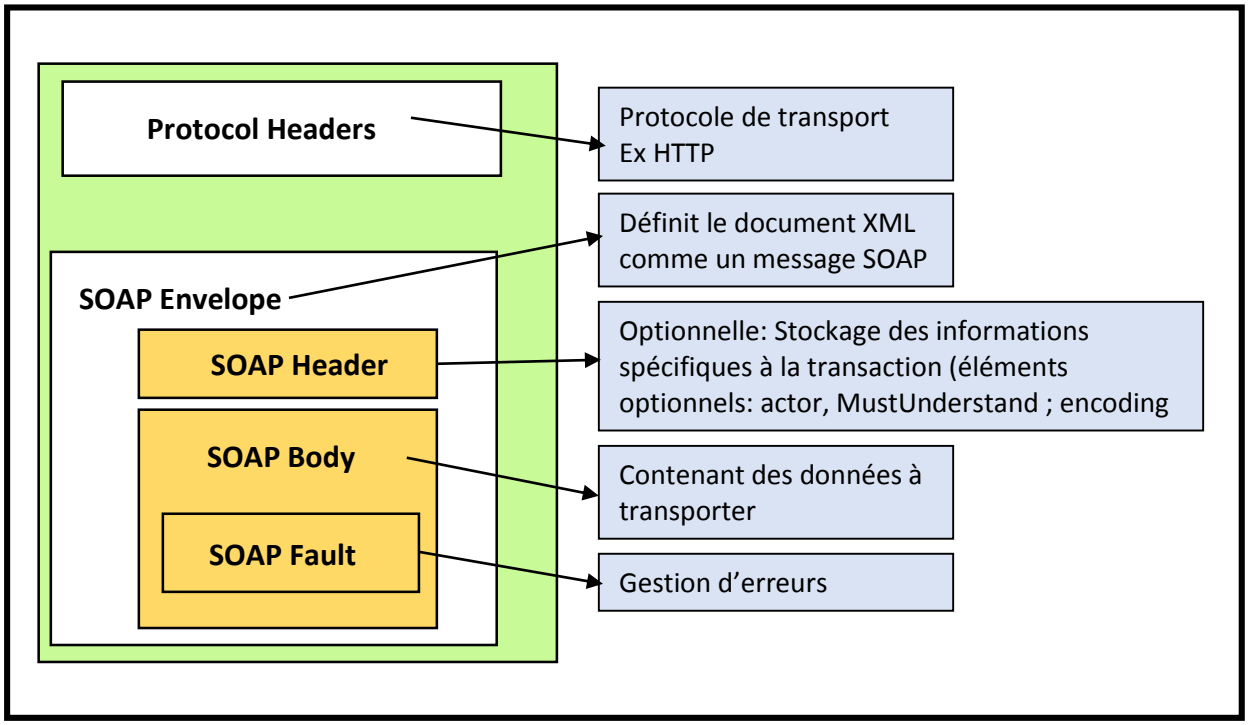


Figure 1.9 message SOAP

a. **Envelope**: C'est ce bloc qui contient le message et ses différents sous-blocs. Il s'agit du bloc racine XML. Il peut contenir un attribut encoding Style dont la valeur est une URL vers un fichier de typage XML qui décrira les types applicables au message SOAP**.**

b. **Header:** C'est un bloc optionnel qui contient des informations d'en-têtes sur le message. S'il est présent, ce bloc doit toujours se trouver avant le bloc Body à l'intérieur du bloc Envelope.

c. **Body:** C'est le bloc qui contient le corps du message. Il doit absolument être présent de manière unique dans chaque message et être contenu dans le bloc Envelope. SOAP ne définit pas comment est structuré le contenu de ce bloc. Cependant, il définit le bloc Fault qui peut s'y trouver.

d. **Fault:** Ce bloc est la seule structure définie par SOAP dans le bloc Body. Il sert à reporter des erreurs lors du traitement du message, ou lors de son transport. Il ne peut apparaître qu'une seule fois par message. Sa présence n'est pas obligatoire.

Exemple

Appel du Web Service stock quote en PHP

Création d'un objet soap client
//L'ouverture du fichier WSDL va permettre d'automatiser l'utilisation du web service.
//Les méthodes définies dans le WSDL seront vues comme des méthodes internes

```
<? php
$params [˝symbol˝] =˝ford˝;
$client = new soapClient (˝http://www.webservicex.net/stockquote.asmx?wsdl˝);
$result = $client->GetQuote ($params);
$ResultQuote = $result->GetQuoteResult;
 echo $ResultQuote;
?>
```

Appel de la méthode GETQUOTE du WS STOCKQUOTE
//Vu comme une méthode locale

Figure 1.10 exemple message SOAP

5.4.WSDL

WSDL (*Web Services Description Language*) est un langage de description standard. C'est l'interface présentée aux utilisateurs. Il indique comment utiliser le service Web et comment interagir avec lui. WSDL est basé sur XML et permet de décrire de façon précise les détails concernant le service Web tels que les protocoles, les ports utilisés, les opérations pouvant être effectuées, les formats des messages d'entrée et de sortie et les exceptions pouvant être envoyées.

5.5 **UDDI (annuaire des services web)**

UDDI (*Universal Description, Discovery and Integration*) est un annuaire de services. Il fournit l'infrastructure de base pour la publication et la découverte des services Web. UDDI permet aux fournisseurs de présenter leurs services Web aux clients.

Les informations qu'il contient peuvent être séparées en trois types :

- Les pages blanches qui incluent l'adresse, le contact et les identifiants relatifs au service Web.
- Les pages jaunes qui identifient les secteurs d'affaires relatifs au service Web ;
- Les pages vertes qui donnent les informations techniques.

6. Fonctionnement des services Web

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

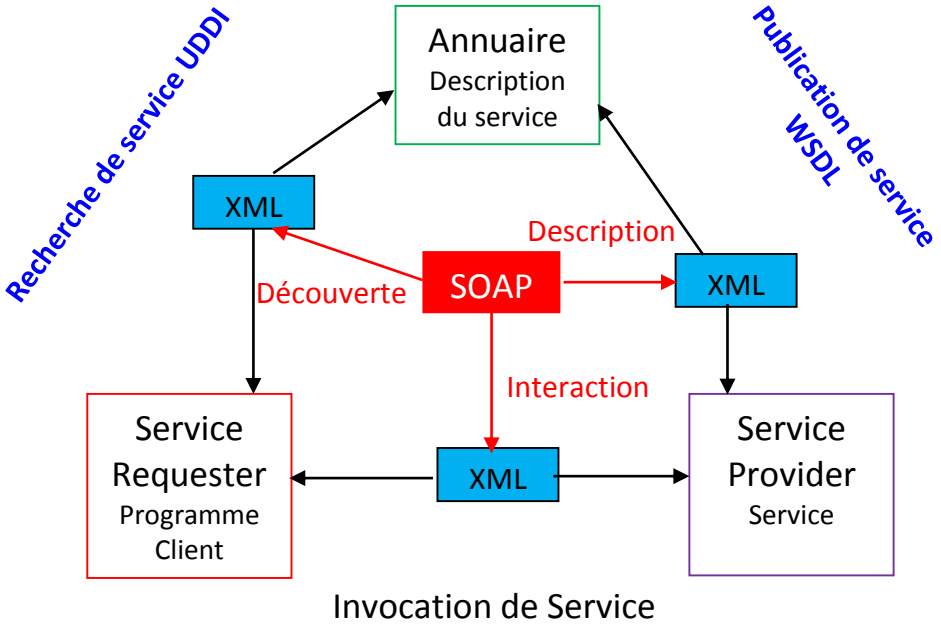


Figure 1.11 **Fonctionnement des services Web**

Décortiquons ce schéma.

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 (REST, XML-RPC, SOAP).

Annuaire service registry

Le registre de service est un annuaire de services. Il fournit un endroit central où les programmeurs peuvent publier de nouveaux services ou en trouver. Les interactions entre ces trois acteurs suivent plusieurs étapes :

a. **La publication du service** : le fournisseur diffuse les descriptions de ses services Web dans l'annuaire.

b. **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.

c. **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.

Le transport de messages XML-RPC ou SOAP est assuré par le standard HTTP.

- SOAP ou XML-RPC prévoit la couche de communication basée sur XML pour accéder à des services Web.

- La description d'un service Web se fait en utilisant le langage WSDL qui expose l'interface du service.

- La publication et la découverte des services Web sont assurées par le biais du référentiel UDDI. Un référentiel UDDI est un catalogue de services Web.

8. Description en couches des services Web *figure1.12*

Les services Web emploient un ensemble de technologies qui ont été conçues afin de respecter une structure en couches sans être dépendante de façon excessive de la pile des protocoles. Cette structure est formée de quatre couches majeures :

Découverte de services	UDDI
Description de services	WSDL
Communication	SOAP
Transport	HTTP

Figure 1.12 **couches des services Web**

Couches technologiques des services Web

a. Couche transport :

Cette couche est responsable du transport des messages XML échangés entre les applications. Actuellement, cette couche inclut HTTP, SMTP, FTP, et de nouveaux protocoles tels que BEEP.

b. Couche communication :

Cette couche est responsable du formatage des données échangées de sorte que les messages peuvent être compris à chaque extrémité. Actuellement, deux styles architecturaux totalement différents sont utilisés pour ces échanges de données. Nous avons d'un côté l'architecture orientée opérations distribuées (protocoles RPC) basée sur XML et qui comprend XML-RPC et SOAP et de l'autre côté une architecture orientée ressources Web, REST (Representational State Transfer) qui se base uniquement sur le bon usage des principes du Web (en particulier, le protocole HTTP).

c. Couche description de service :

Cette couche est responsable de la description de l'interface publique du service Web. Le langage utilisé pour décrire un service Web est WSDL qui est la notation standard basée sur XML pour construire la description de l'interface d'un service. Cette spécification définit une grammaire XML pour décrire les services Web comme des ensembles de points finaux de communication (ports) à travers lesquels on effectue l'échange de messages.

d. Couche publication/découverte de service :

Cette couche est chargée de centraliser les services dans un registre commun, et de simplifier les fonctionnalités de recherche et de publication des services Web. Actuellement, la découverte des services est assurée par un annuaire UDDI (Universal Description, Discrovery and Integration).

9. Critères de QoS (Quality of service)

La qualité de service désigne la capacité d'un service à répondre par ses caractéristiques aux différents besoins de ses utilisateurs ou consommateurs (définition Afnor).
Les caractéristiques prises en compte pour déterminer la qualité d'un service sont évidemment variables en fonction du service proposé.
Dans notre travail, les principales composantes de la qualité de service sont :

- La réputation d'un service
- la disponibilité d'un service
- les temps de réponses d'un service
- la fiabilité d'un service
- le cout d'un service

La notion de qualité de service s'applique au secteur des services mais également dans le domaine des produits en ce qui concerne les services associés et notamment le service après-vente.
La qualité de service est normalisée par les normes ISO 9000 et par de nombreuses normes spécifiques aux différents secteurs (service de l'eau, services informatiques,..). Elle fait donc également l'objet de certification qualité.

10. Avantages et inconvénients des services web

a. Avantage des services web

Quels avantages les services web offrent-ils sur les technologies concurrentes (CORBA, RMI, EJB)?

- Ils sont indépendants des plateformes et des langages (XML)
- La plupart des services web utilisent le protocole HTTP pour transmettre les messages entre clients et serveur
- *D'autres protocoles comme TCP/IP, SMTP, JMS sont possibles.*
- Ils autorisent un couplage faible entre le client et le serveur.
- *Le client n'a pas connaissance du service web jusqu'à ce qu'il l'utilise*
- Des mécanismes de découverte sont prévus.
- *Un client peut déterminer le fournisseur le plus apte à effectuer une action* voulue.
- La technologie Web Service met en œuvre des fonctionnalités d'auto-description des fournisseurs et des services qui leurs sont associés.
- *L'utilisation d'un fournisseur de services web peut être gérée de manière dynamique.*

b. inconvénients des services web

- Les services web sont un concept, leur implémentation étant laissée libre dès le départ.
- Manque de polyvalence
- La sécurité des services web n'est pas encore finalisée pour gérer les défis courants.
- Les technologies CORBA et EJB proposent des services de cycle de vie, de persistance, de transaction, de sécurité.
- Surcoût dans la transmission des informations.
- Le transfert de données en XML est moins efficace qu'en binaire

Conclusion

Les Web services sont la dernière technologie pour l'intégration et l'interopérabilité des systèmes distribués. Basés sur le standard XML, ils sont caractérisés par leurs indépendances par rapport aux plates formes et aux systèmes d'exploitation, ce qui a impliqué leur adoption par les différentes organisations commerciales et industrielles offrant leurs services à travers le Web, et par conséquent l'augmentation du nombre de services offerts.

Chapitre II

I. Introduction

Avec l'augmentation du nombre de fournisseurs de Web services, il arrive très fréquemment que de nombreux Web services répondent à un même besoin fonctionnel, mais qui offrent des propriétés de qualité de services différentes. Une sélection doit être alors réalisée pour déterminer quels sont les Web services pertinents qui satisferaient les besoins d'un utilisateur. Le problème de sélection est considéré comme étant un problème **NP-difficile** vu le nombre croissant de combinaisons de compositions possibles, les propriétés de qualité de services ne sont pas supportées par le standard UDDI et les Web services sont dispersés dans ce dernier, c'est- à-dire qu'ils ne sont pas regroupés en services compatibles (qui offrent la même fonctionnalité).

Dans notre travail on se focalise sur la sélection et non le classement des Web services en services compatibles, néanmoins nous supposons que les Web services dans le registre UDDI sont regroupés en services compatibles afin de minimiser la complexité de la sélection. Le but de notre travail est de proposer une méthode de sélection de Web services qui prend en compte les préférences d'un utilisateur en termes de propriétés non fonctionnelles. Cette méthode cherche les Web services pertinents qui satisferaient les différentes contraintes définies par un utilisateur.

Dans la suite de ce chapitre, nous allons présenter les détails de notre proposition.

II. Algorithme espérance-maximisation

1. Définition et historique

L'Algorithme **espérance-maximisation** (en anglais "Expectation-maximisation algorithm", souvent abrégé "EM"), proposé par Dempster et al. (1977), est une classe d'algorithmes qui permettent de trouver le maximum de vraisemblance des paramètres de modèles probabilistes lorsque le modèle dépend de variables latentes non observables.

On utilise souvent Espérance-maximisation pour la classification de données, en apprentissage machine, ou en vision artificielle. Espérance-maximisation alterne des étapes d'évaluation de l'espérance (E), où l'on calcule l'espérance de la vraisemblance en tenant compte des dernières variables observées, et une étape de maximisation (M), où l'on estime le maximum de vraisemblance des paramètres en maximisant la vraisemblance trouvée à l'étape E.

On utilise ensuite les paramètres trouvés en M comme point de départ d'une nouvelle phase d'évaluation de l'espérance, et l'on itère ainsi.

[c1]

2. Principe et fonctionnement

En considérant un échantillon $\mathbf{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ d'individus suivant une loi $f(\boldsymbol{x}_i, \theta)$ paramétrée par $\boldsymbol{\theta}$, on cherche à déterminer le paramètre $\boldsymbol{\theta}$ maximisant la log-vraisemblance donnée par

$$L(\mathbf{x}; \boldsymbol{\theta}) = \sum_{i=1}^{n} \log f(\boldsymbol{x}_i, \boldsymbol{\theta}).$$

Cet algorithme est particulièrement utile lorsque la maximisation de *L* est très complexe mais que, sous réserve de connaître certaines données judicieusement choisies, on peut très simplement déterminer $\boldsymbol{\theta}$.

Dans ce cas, on s'appuie sur des données complétées par un vecteur

$\mathbf{z} = (z_1, \ldots, z_n)$ Inconnu.

En notant $f(z_i|\boldsymbol{x}_i; \theta)$ la probabilité de z_i sachant $\boldsymbol{x}_i$ et le paramètre $\boldsymbol{\theta}$, on peut définir la log-vraisemblance complétée comme la quantité

$$L\left((\mathbf{x}, \mathbf{z}); \boldsymbol{\theta}\right) = \sum_{i=1}^{n} \left(\log f(z_i|\boldsymbol{x}_i, \boldsymbol{\theta}) + \log f(\boldsymbol{x}_i; \boldsymbol{\theta})\right).$$

Et donc,

$$L(\mathbf{x}; \boldsymbol{\theta}) = L\left((\mathbf{x}, \mathbf{z}); \boldsymbol{\theta}\right) - \sum_{i=1}^{n} \log f(z_i|\boldsymbol{x}_i, \boldsymbol{\theta}).$$

L'algorithme EM est une procédure itérative basée sur l'espérance des données complétées conditionnellement au paramètre courant. En notant $\boldsymbol{\theta}^{(c)}$ ce paramètre, on peut écrire

$$E\left[L(\mathbf{x}; \boldsymbol{\theta})|\boldsymbol{\theta}^{(c)}\right] = E\left[L\left((\mathbf{x}, \mathbf{z}); \boldsymbol{\theta}\right))|\boldsymbol{\theta}^{(c)}\right] - E\left[\sum_{i=1}^{n} \log f(z_i|\boldsymbol{x}_i, \boldsymbol{\theta}))|\boldsymbol{\theta}^{(c)}\right],$$

ou encore

$$L(\mathbf{x}; \boldsymbol{\theta}) = Q\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(c)}\right) - H\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(c)}\right)$$

Avec

$$Q\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(c)}\right) = E\left[L\left((\mathbf{x}, \mathbf{z}); \boldsymbol{\theta}\right))|\boldsymbol{\theta}^{(c)}\right]$$

et

$$H\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(c)}\right) = E\left[\sum_{i=1}^{n} \log f(z_i|\boldsymbol{x}_i, \boldsymbol{\theta}))|\boldsymbol{\theta}^{(c)}\right].$$

On montre que la suite définie par

$$\boldsymbol{\theta}^{(c+1)} = \arg\max_{\boldsymbol{\theta}} \left(Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(c)}\right)\right)$$

Fait tendre $L\left(\mathbf{x}; \boldsymbol{\theta}^{(c+1)}\right)$ vers un maximum local.

On peut donc définir l'algorithme EM de la manière suivante:

- Initialisation au hasard de $\boldsymbol{\theta}^{(0)}$
- c=0
- Tant que l'algorithme n'a pas convergé, faire
  - Evaluation de l'espérance (étape E) :

    $$Q\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(c)}\right) = E\left[L\left((\mathbf{x}, \mathbf{z}); \boldsymbol{\theta}\right)) | \boldsymbol{\theta}^{(c)}\right]$$

  - Maximisation (étape M) : $\boldsymbol{\theta}^{(c+1)} = \arg\max_{\boldsymbol{\theta}} \left(Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(\boldsymbol{c})}\right)\right)$
- c=c+1
- Fin

En pratique, pour s'affranchir du caractère local du maximum atteint, on fait tourner l'algorithme EM un grand nombre de fois à partir de valeurs initiales différentes de manière à avoir de plus grandes chances d'atteindre le maximum global de vraisemblance.

3. Variantes usuelles d'EM

L'algorithme EM, bien que très performant et souvent simple à mettre en œuvre, pose quand même parfois quelques problèmes qui ont donné lieu à des développements complémentaires. Parmi ceux-ci, nous évoquerons un développement appelé *GEM* (Generalized EM) qui permet de simplifier le problème de l'étape maximisation, un autre, appelé *CEM* (Classification EM) permettant de prendre en compte l'aspect classification lors de l'estimation (c'est la version que nous avons adopté dans ce travail), et un dernier, *SEM* (Stocastic EM) dont l'objectif est de réduire le risque de tomber dans un optimum local de vraisemblance.

a. Algorithme GEM

GEM a été proposé en même temps qu'EM par Dempster et al. (1977) qui ont prouvé que pour assurer la convergence vers un maximum local de vraisemblance, il n'est pas nécessaire de maximiser *Q* à chaque étape mais qu'une simple **amélioration de *Q*** est suffisante.

GEM peut donc s'écrire de la manière suivante:

- Initialisation au hasard de $\theta^{(0)}$
- $c = 0$
- Tant que l'algorithme n'a pas convergé, faire
  - choisir $\theta^{(c+1)}$ tel que $Q\left(\theta, \theta^{(c+1)}\right) > Q\left(\theta, \theta^{(c)}\right)$
  - $c = c + 1$
- Fin

b. Algorithme CEM

L'algorithme EM se positionne dans une optique *estimation*, c'est-à-dire qu'on cherche à maximiser la vraisemblance du paramètreθ, sans considération de la classification faite *a posteriori* en utilisant la règle de Bayes.

L'approche *classification*, proposée par Celeux et Govaert (1991) consiste à optimiser, non pas la vraisemblance du paramètre, mais directement la vraisemblance complétée, donnée, dans le cas des modèles de mélange, par

$$L(x, z; \theta) = \sum_{i=1}^{n} \sum_{k=1}^{g} z_{ik} \log\left(\pi_k f(x, \theta_k)\right)$$

Pour cela, il suffit de procéder de la manière suivante:

- Initialisation au hasard de $\theta^{(0)}$
- $c = 0$
- Tant que l'algorithme n'a pas convergé, faire
  - $z^{(c+1)} = \arg\max_{z}\left(L\left(x, z; \theta^{(c)}\right)\right)$
  - $\theta^{(c+1)} = \arg\max_{\theta}\left(L\left(x, z^{(c+1)}; \theta\right)\right)$
  - $c = c + 1$
- Fin

c. Algorithme SEM

Afin de réduire le risque de tomber dans un maximum local de vraisemblance, Celeux et Diebolt (1985) proposent d'intercaler une étape stochastique de classification entre les étapes E et M. Après le calcul des probabilités $t_{ik}^{(c)}$, l'appartenance $z_{ik}^{(c)}$ des individus aux classes est tirée aléatoirement selon une loi multinomiale de paramètre $\mathcal{M}\left(1, t_{i1}^{(q)}, \ldots, t_{ig}^{(q)}\right)$

Contrairement à ce qui se produit dans l'algorithme CEM, on ne peut considérer que l'algorithme a convergé lorsque les individus ne changent plus de classes. En effet, celles-ci étant tirées aléatoirement, la suite $\left(z^{(q)}, \theta^{(q)}\right)$ ne converge pas au sens strict. En pratique, Celeux et Diebolt (1985) proposent de lancer l'algorithme SEM un nombre de fois donné puis d'utiliser l'algorithme CEM pour obtenir une partition et une estimation du paramètres θ.

4. Organigramme EM

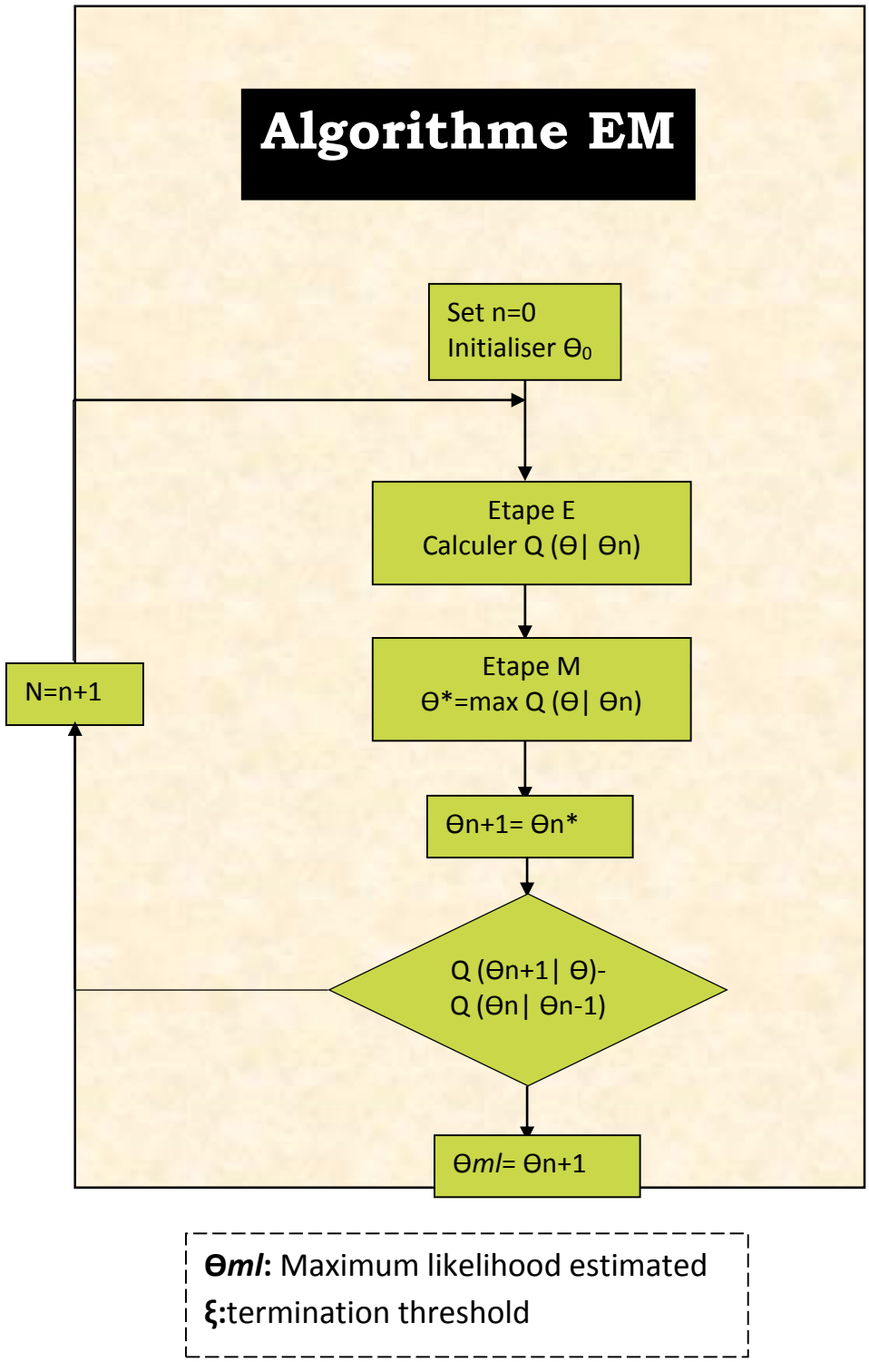


Figure 2.1 organigramme d'EM

5. Avantages et inconvénient d'EM

a. Avantages

Rapide : il est l'algorithme le plus rapide pour apprendre les modèles de mélange.

Agnostique : EM n'est pas juste une maximisation de la probabilité, il ne va pas converger la moyenné vers zéro et donnera une taille pour les clusters pour avoir une structure spécifique qui pourra être appliqué ou non aussi.

b. Inconvénient

Singularités : quand on a de nombreux points par mélange, ou de grandes base de travail, l'estimation de la matrice de covariance devient très difficile

Nombre de composants : cet algorithme utilisera toujours tous les composants qui l à accès, il a besoin de critères de donnée complexe pour décider le nombre de composant à utiliser dans l'absence d'indices externes.

6. Discussion

EM est intéressant et très puissant (application multiple), mais pas de bonne qualité pour des séries avec une grande variance-covariance.

Alternative: ex: EMBootstrapping.

Pour ce qui est des paramètres du modèle, il est possible d'obtenir de bons résultats avec une bonne précision.

Les caractéristiques globales sont donc bien reproduites par l'algorithme.

7. Fonction objectif

Les critères de services web doivent optimiser une fonction globale qui maximise les critères positifs (Réputation, Disponibilité et Fiabilité) et minimise les critères négatifs (cout et temps D'exécution), et nous devons également satisfaire des contraintes globales par exemple :

- Cout (sol) < 170 : Le cout d'une solution doit être moins de 170euros.
- Temps (sol) < 1050s : Le temps d'exécution doit être moins de 1050ms.
- Rep (sol) > 3 : La réputation doit être supérieure à 3.
- Disp (sol) > 0.6 : La disponibilité doit être supérieure à 60%.
- Fiab (sol) > 0.6 : La fiabilité doit être supérieure à 60%.

Nous définissons la fonction suivante :

$$\sum_{Qi \in Neg} Wi \frac{Qi\,max - Qi\,sol}{Qi\,max - Qi\,min} + \sum_{Qi \in Pos} Wi \frac{Qi\,sol - Qi\,min}{Qi\,max - Qi\,min}$$

On doit calculer la fonction objective de tous les services d'un cluster, le service ayant la fonction objective maximal sera le représentant du cluster (le meilleur service).

8. Les stratégies de sélection (sélection globale)

On choisit la combinaison de Web services qui garantit la meilleure qualité globale en tenant compte des contraintes de QoS et des préférences globales assignées pour l'ensemble des taches.

Afin de pouvoir appliquer une stratégie de sélection globale, on calcule la valeur des critères de QoS relative au service composite. Le calcul de ces critères s'appuie sur l'application de fonctions d'agrégation (qui peuvent être une somme, un produit, un minimum, ...) .Ensuite, on choisit la combinaison qui offre les meilleures valeurs de QoS parmi toutes les combinaisons possibles par rapport aux contraintes et aux préférences de l'utilisateur. La Figure suivante décrit la stratégie de sélection globale.

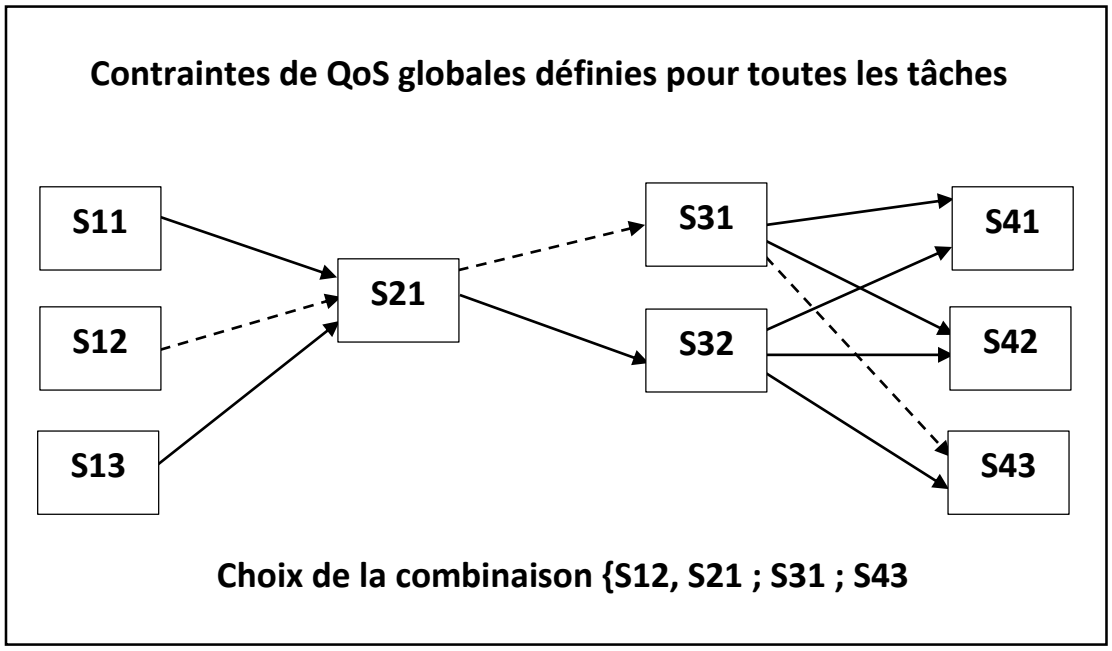


Figure 2.2 Sélection Globale

Algorithme de parcours des compositions

Entrées

- N classes (ou hiérarchies de clusters de services, nous notons que chaque cluster est caractérisé par un représentant calculé avec la fonction objective citée précédemment).
- Dernier_niveau ∈ {1,2}

Sorties

- L'optimum global : og (s'il existe), sinon « échec»

1. og=echec // nous initialisons l'optimum global à « échec »
2. initialiser le niveau courant : niveau_courant =1 (la racine de chaque arbre)
3. pour j=1 jusqu'à N

   Lj=representants_classe (j,niveau_courant)
4. **Tant que** ((niveau_courant≠ dernier_niveau+1) et (og=echec))

   Faire

   a- composition=recherche_exhaustive (L_1,…L_N)

   b- **Si** composition = échec **Alors //** les contraintes globales ne sont pas vérifiées.

      1- niveau_courant=niveau_courant+1

      2- pour j=1 jusqu'à N : Lj=representants_classe(j,niveau_courant)

   Sinon

   og= composition

fin_Faire

5. Retourner og

L'étape 4.a parcourt toutes les solutions possibles à ce niveau en vérifiant les contraintes globales, et en mettant à jours la solution finale dans le cas où la composition courante est meilleure que l'ancienne. Nous notons que le critère de comparaison des solutions, est tout simplement la moyenne des performances (fonction objective) des services constituant la composition.

Ce travail est répété pour chaque niveau de l'arborescence.

Exemple :

- Supposons qu'on a un client A qui veut effectuer une réservation d'hôtel et restauration pour fêter le nouvel an.
- Sachant que notre client peut dépenser un budget de 270 euros au maximum et qu'il n'aime pas trop attendre.

Déroulement de l'exemple

	Rep	Dis	Fia	Cout	Temps
Resto 1	4	5	0,70	100	220
Resto 2	5	2	0,36	230	250
Resto 3	3	4	0,75	125	170
Resto 4	1	2	0,50	135	195

Classification EM

Classification EM

	Rep	Dis	Fia	Cout	Temps
Resto 1	5	5	0,70	100	220
Resto 3	3	4	0,75	125	175

Sélection des représentants de cluster

	Rep	Dis	Fia	Cout	Temps
Resto 2	5	2	0,36	230	250
Resto 4	1	2	0,50	135	195

Sélection des représentants de cluster

Déroulement de l'exemple

	Rep	Dis	Fia	Cout	Temps
Hotel 1	5	4	0,75	120	350
Hotel 2	4	1	0,29	100	220
Hotel 3	2	3	0,35	70	420
Hotel 4	3	2	0,55	160	190

Classification EM

Classification EM

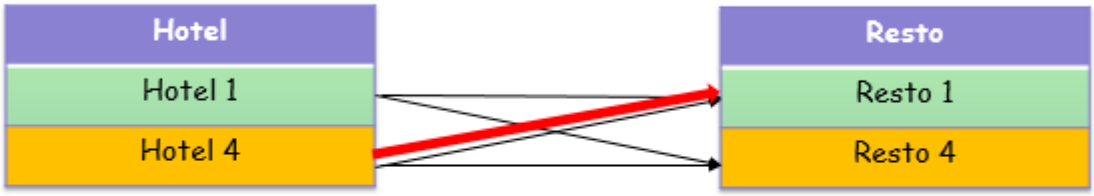
	Rep	Dis	Fia	Cout	Temps
Hotel 1	5	4	0,75	120	350
Hotel 3	2	3	0,35	70	420

Sélection des représentants de cluster

	Rep	Dis	Fia	Cout	Temps
Hotel 2	4	1	0,29	100	220
Hotel 4	3	2	0,55	160	190

Sélection des représentants de cluster

Déroulement de l'exemple



La meilleure combinaison pour le client A est (Hotel 4, Resto 1) pour un coût de 260 euro et de 310 ms

La distance utilisée pour la classification est la suivante :

$$distance(x_i,j) = -log\left(prop_j / \sqrt{2\pi^p \times det(Cj)} \times e^{-\frac{1}{2}(x_i-\mu_j)^t Cj^{-1}(x_i-\mu_j)}\right)$$

- ✓ K : le nombre de clusters.
- ✓ P : la dimension du vecteur xi
- ✓ $C\,j$: la matrice variances-covariances .
- ✓ μ_j : la moyenne des vecteurs .
- ✓ x_i : le nième vecteur .
- ✓ $prop_j$: la proportion des vecteurs(points) appartenant au cluster j

III. Description de la base

a. Description de la base

Nous avons créé un schéma de service qui contient dix classes de services web. Le nombre d'instances dans chaque classe et de quarante (50) fournisseurs de services web.

Chaque fournisseur se caractérise par des qualités de service. Nous utilisons cinq paramètres pour évaluer les opérations des services : latence, fiabilité, disponibilité, cout et réputation. Les valeurs de ces paramètres sont générées en se basant sur une distribution uniforme. Un échantillon de notre base est illustré dans la ***table II.1*** qui représente le fournisseur i des cinq premières classes.

Critère	Intervalles
Réputation	[0, 10]
Cout	[0, 70]
Disponibilité	[0.5, 1]
Temps d'exécution	[1, 350]
Fiabilité	[0.5, 1]

Tableau II.1 : description de base de données

b. Description des requêtes

Les requêtes concernent les utilisateurs selon leurs budgets et leurs besoins, le client doit donner 5 requêtes :

- La borne maximale du cout.
- La borne maximale du temps d'exécutions.
- La borne minimale de la disponibilité.
- La borne minimale de la réputation.
- La borne minimale de la fiabilité.

IV. Conception

1. Diagramme de cas d'utilisation

Un diagramme de cas d'utilisation permet de structurer les besoins des utilisateurs et les objectifs correspondants d'un système. Il permet aussi d'identifier les possibilités d'interactions entre le système et les acteurs.

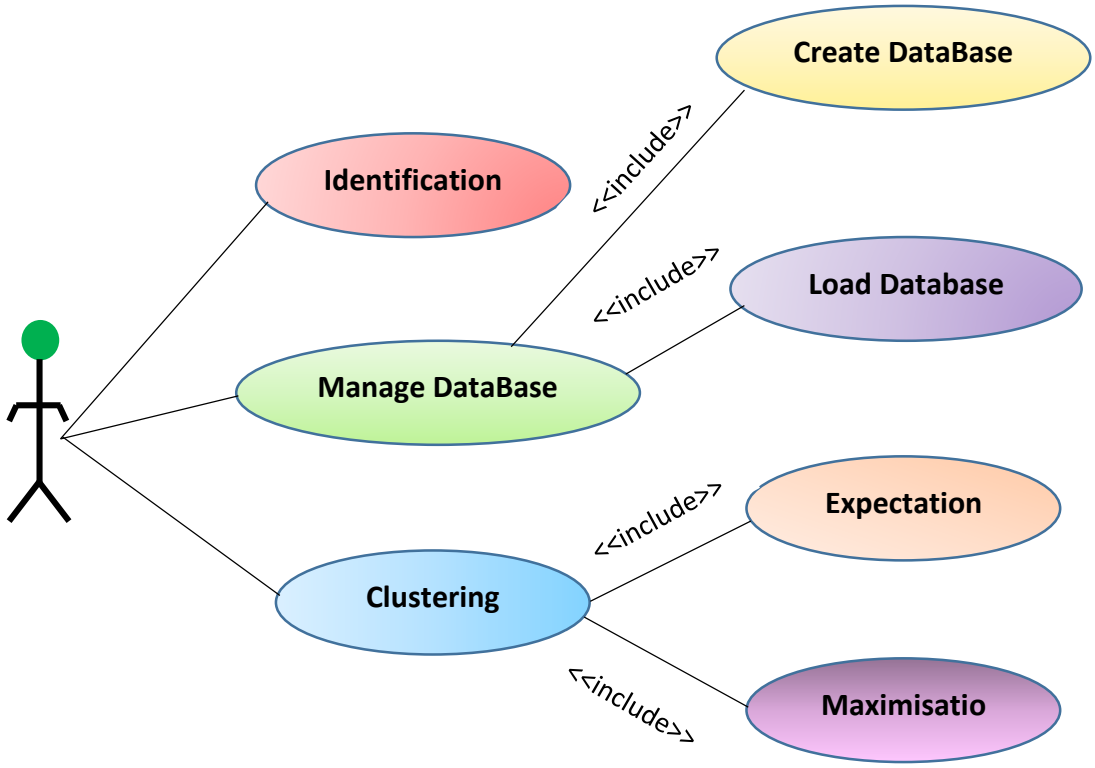


Figure 2.3 Diagramme de cas D'utilisation

2. Diagramme de séquence

Un diagramme de séquence permet de représenter des collaborations entre objets selon un point de vue temporel. Les objets communiquent en échangeant des messages représentés sous forme de flèches. Il peut servir à illustrer un cas d'utilisation.

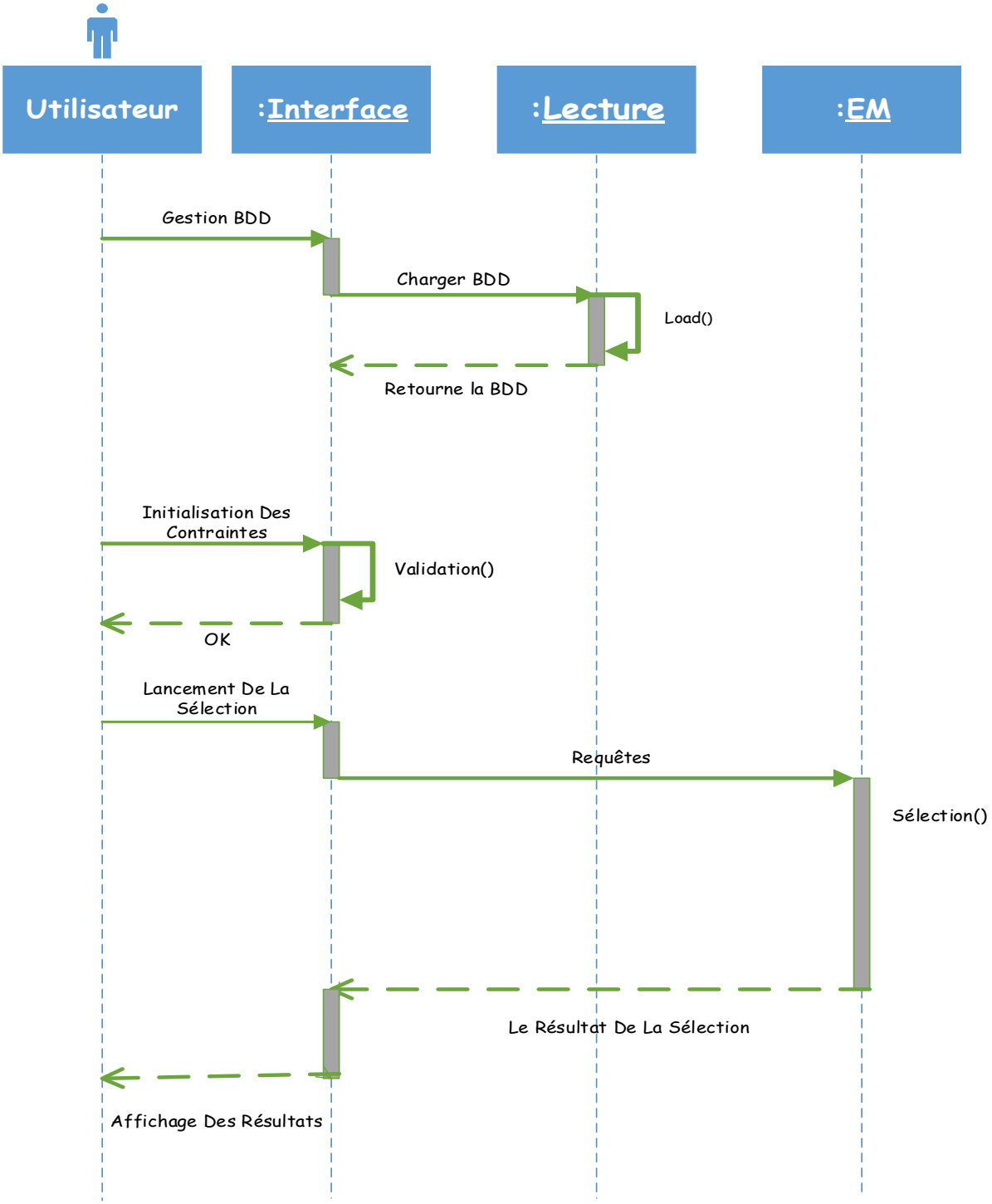


Figure 2.4 Diagramme De Séquence

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 quelles seront les composantes du système final. Il permet de structurer le travail de développement de manière très efficace.

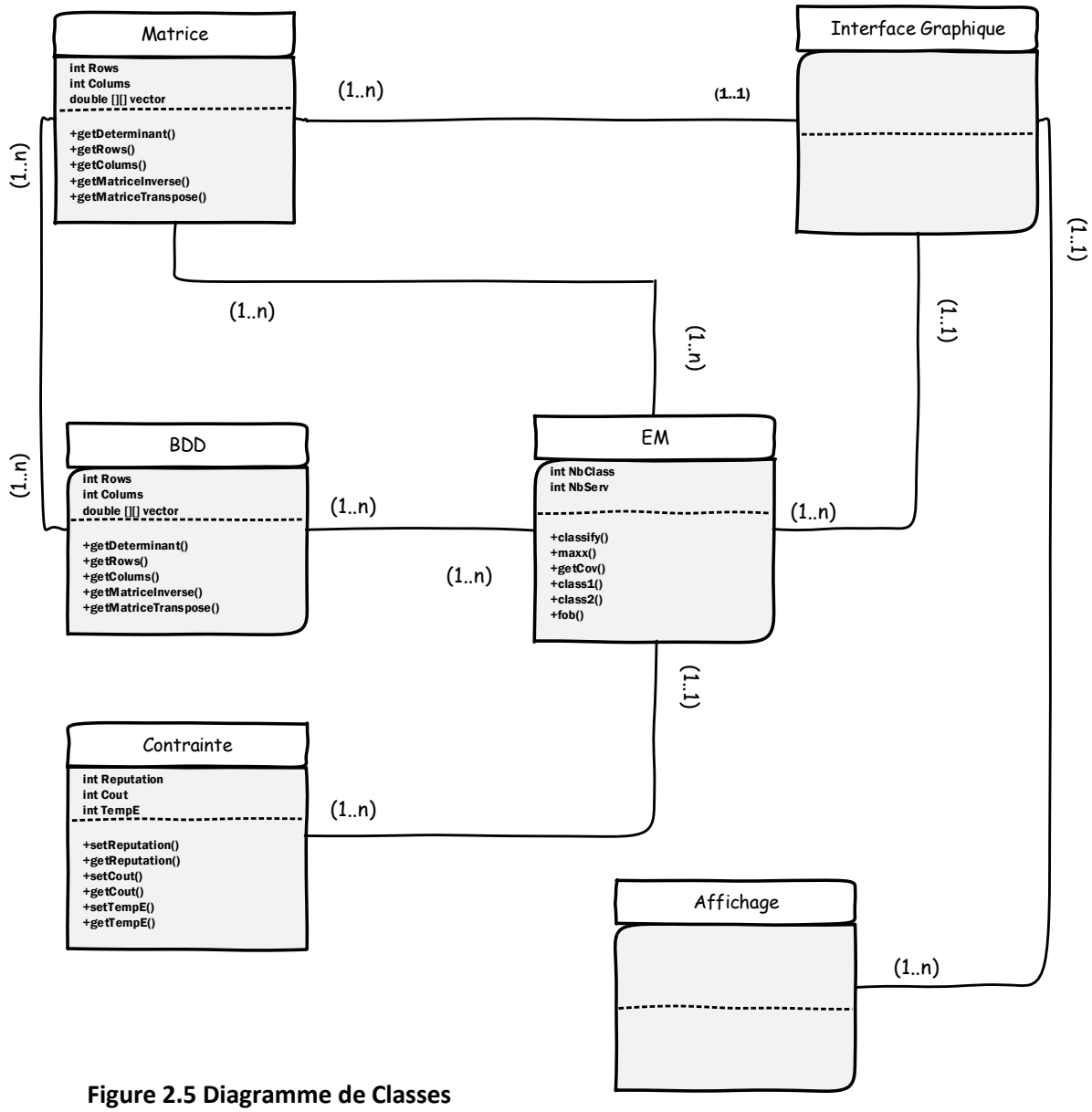


Figure 2.5 Diagramme de Classes

V. Interface Humain/Machine (IHM)

1. Fenêtres principales

Figure 2.6 Identification

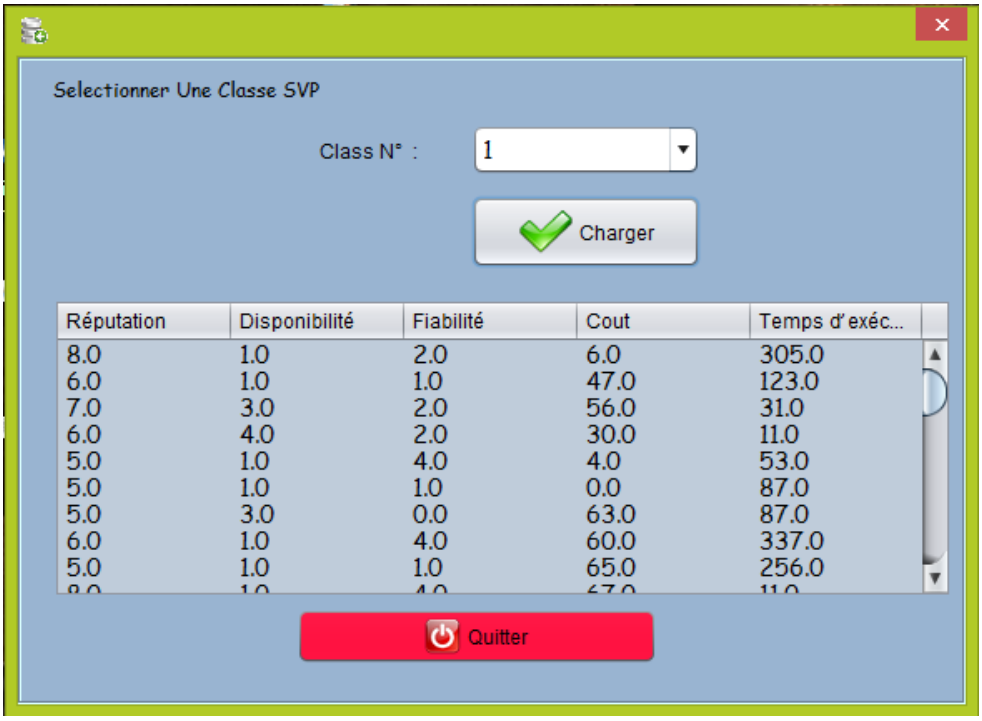


Figure 2.7 Chargement de la base de données

Figure 2.8 Génération des contraintes

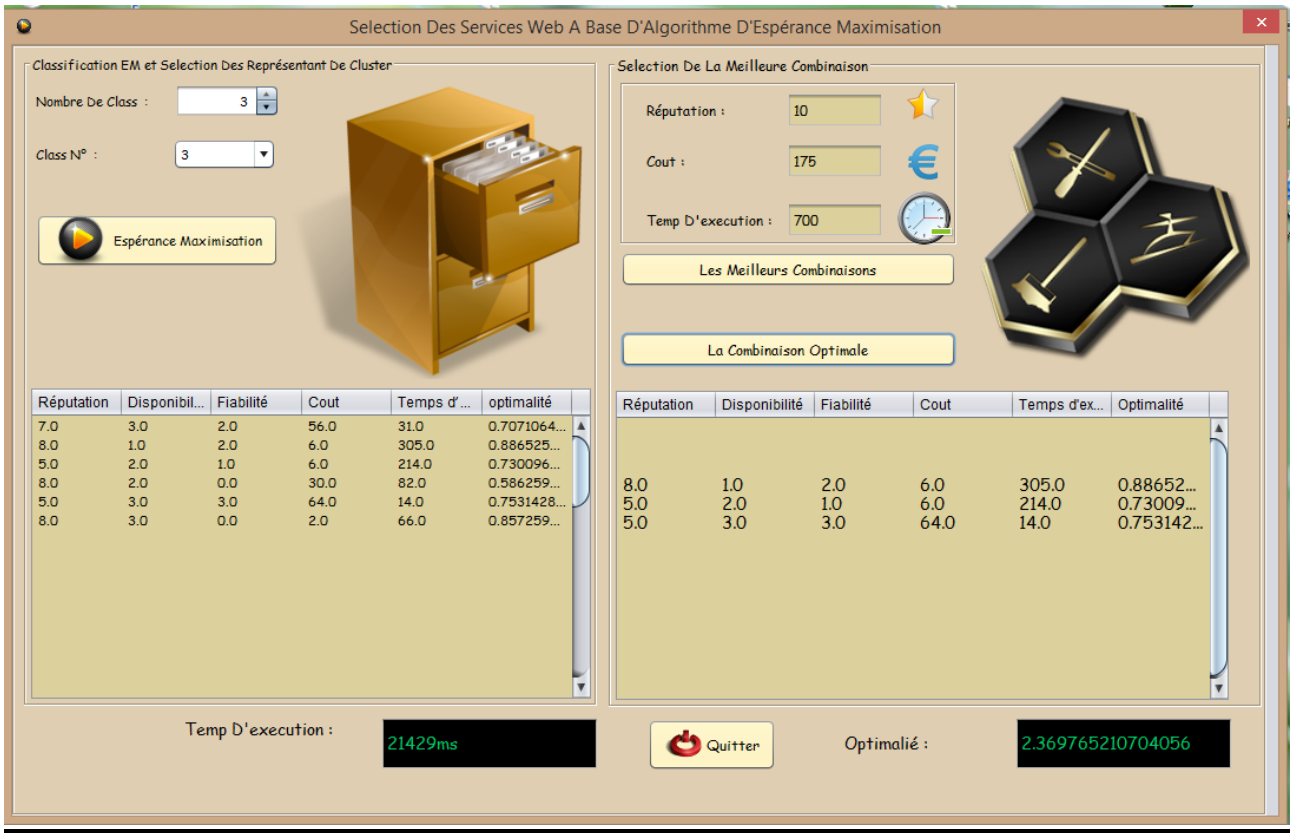


Figure 2.9 exécution d'EM et résultat final

2. Expérimentation

Pour évaluer l'efficacité de notre proposition, nous avons implémenté un prototype et réaliser des expériences sur ce dernier afin d'évaluer les performances du système et de comparer les résultats trouvés par notre proposition avec les résultats trouvés par d'autre approches.

Outils de développement utilisés

a. Langage de programmation

Le langage Java a été utilisé pour la réalisation du prototype. Le langage Java est un langage orienté objet. L'avantage de celui-ci est qu'il fonctionne sur toute plateforme vu sa portabilité ce qui permet de créer des applications indépendantes de l'environnement du travail (indépendance vis-à-vis du système d'exploitation et du type de la machine).

b. Environnement utilisé

Pour le choix de l'environnement de développement, nous utilisons Netbeans car il est une plateforme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plugins et le support de plusieurs plateformes d'exécution comme : Windows, Linux, Mac OS.
On a utilisé netbeans sous Windows 8.1 64 bits processeur i7 12 giga de ram.

c. Bibliothèque utilisée

- commons-math3-3.0
- jakarta-poi-2.5
- jama
- matlabcontrol
- jmi

3. Résultât et discussion

a. Résultât

Réputation minimal=10, cout maximal=175, temps d'exécution maximal=700 ms

Nbr classes	Classes utilisé	Nbr service	Nbr résultat	optimalité	Temps moyen
3	{1, 2,3}	50	2	0.731 0.789	22348 ms
3	{3, 4, 5}	50	2	0.699 0.703	22984 ms
3	{1, 3, 5}	50	1	0.797	22633 ms
3	{2, 4, 6}	50	2	0.618 0.570	22901 ms
4	{1, 2, 3,4}	50	2	0.717	29167 ms
4	{1, 2, 5, 6}	50	1	0.703	29406 ms
4	{2, 3, 4, 5}	50	0	--	22816 ms
5	{1, 2, 3,4, 5}	50	0	--	36361 ms
5	{3, 4, 5, 6,7}	50	0	--	35156 ms

Tableau II.2 : 9 simulations d'EM

Discussion :

Dans les deux premiers cas nous avons trouvé deux solutions donc on doit faire une deuxième itération pour trouver la combinaison optimale.

Dans le troisième et quatrième cas nous avons trouvé une seule solution donc elle est optimale.

Dans les trois derniers cas nous n'avons trouvé aucune solution(les contraintes sont violées).

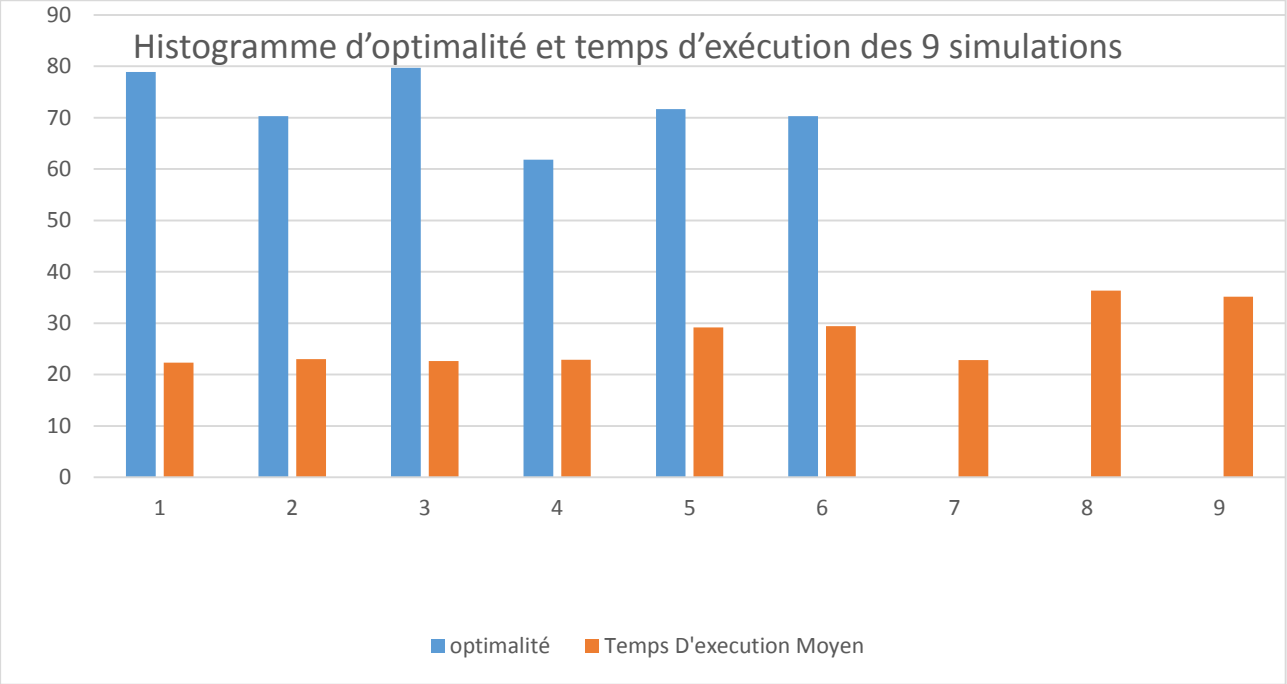


Figure 2.10 **Histogramme d'optimalité et temps d'exécution des 9 simulations**

Réputation minimal=27, cout maximal=210, temps d'exécution maximal=1050 ms

Nbr classes	Classes utilisé	Nbr service	Nbr résultat	optimalité	Temps moyen
5	{1, 2, 3,4, 5}	50	1	0.724	35522 ms
5	{2, 3,4, 5, 6}	50	1	0.671	35891 ms
5	{1, 3, 4, 5,6}	50	1	0.704	39658 ms
5	{2, 3, 6, 7,8}	50	2	0.706 0.677	48324 ms
6	{1, 2, 3, 4, 5, 6}	50	0	--	42298 ms
6	{2, 3, 6, 7, 8, 9}	50	2	0.680 0.656	62867 ms
6	{2, 4, 5, 6, 7, 8}	50	0	--	43955 ms
7	{2, 4, 5, 6, 7, 8, 9}	50	0	--	67944 ms
8	{3, 4, 5, 6,7}	50	0	--	35156 ms
10	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}	50	0	--	89245 ms

Tableau II.3 : 10 simulations d'EM

Discussion :

Pour les trois premiers cas nous avons trouvé une seule solution donc elle est optimale.

Pour le quatrième et le sixième cas nous avons trouvé deux solutions donc on doit procéder à une deuxième itération

Pour le cinquième et les quatre derniers cas nous n'avons trouvé aucune solution.

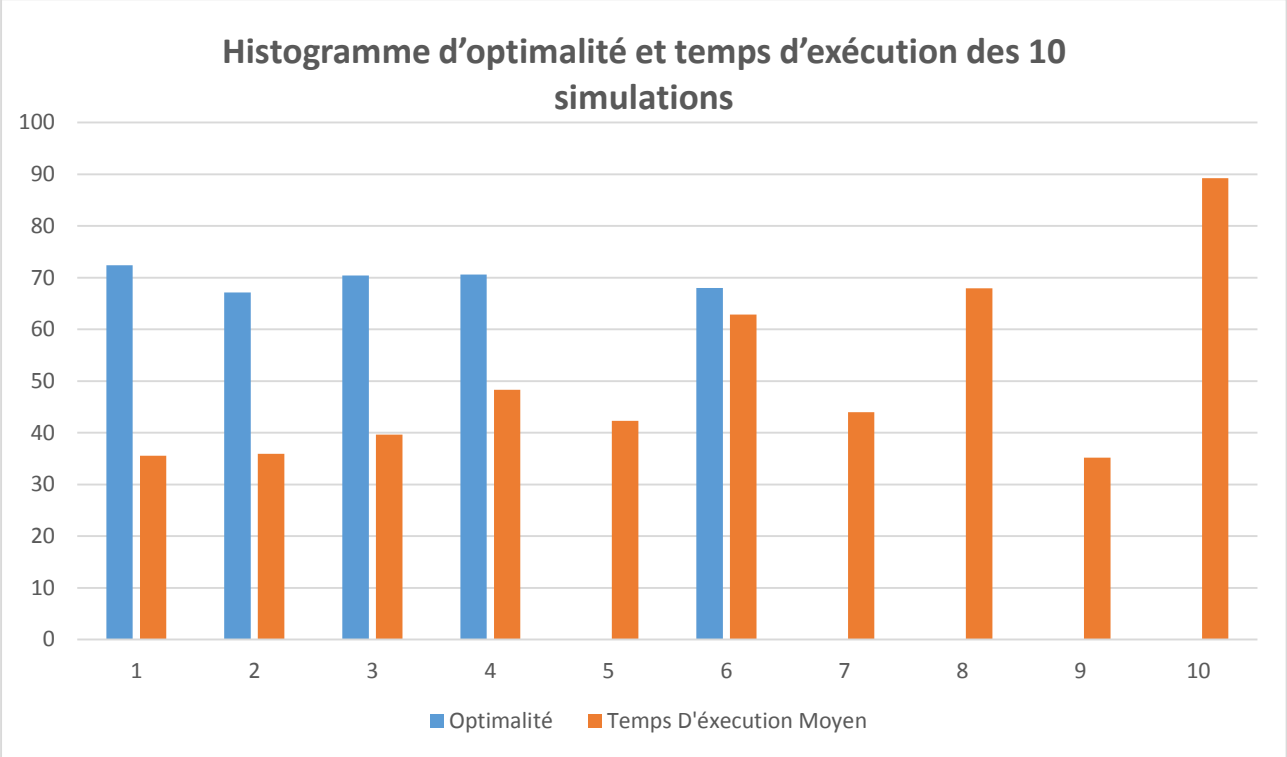


Figure 2.11 **Histogramme d'optimalité et temps d'exécution des 10 simulations**

VI. Conclusion

Dans ce chapitre, on a proposé l'algorithme d'Esperance Maximisation EM pour trouver une solution à la problématique citée précédemment.

Apres tests et expérimentation, on a relevé les résultats de notre étude, ce qui nous permettra d'estimer le procédé utilisé et de penser à améliorer l'optimalité.

Les résultats obtenus sont remarquables, peut-être pas meilleurs, mais nous invitent à se pencher vers d'autres approches pour pouvoir donner un jugement adéquat.

Conclusion générale

Dans ce travail, nous avons traité le problème de la sélection des Web services selon leurs critères de qualité de service. Notre objectif consiste à trouver une proposition qui répond aux préférences de l'utilisateur en termes de qualité de service et dans des temps de réponse raisonnables.
Pour répondre à notre objectif, nous avons débuté notre travail par des généralités sur les Web services et leurs architectures pour passer ensuite aux méthodes proposées dans la littérature concernant la sélection des Web services. Cette étude nous a permis de traiter la sélection et d'envisager une solution basée sur l'algorithme d'espérance maximisation **EM** auquel on a adapté certaines étapes (**étape E** et **étape M**) et opérateurs. En plus de l'adaptation de certains opérateurs et étapes de l'algorithme d'espérance maximisation, nous l'avons aussi hybridé avec une recherche exhaustive qui est exécutée sur les résultats trouvés par notre algorithme pour l'améliorer.

D'après les résultats obtenus notre proposition parvient à trouver des solutions de très bonne qualité et qui vérifient les contraintes de l'utilisateur.

Comme perspectives, nous envisageons un ensemble de travaux futurs qui concernent les aspects suivants :
– Améliorer le temps d'exécution.
– Amélioration de la gestion des contraintes.
– Introduction de la gestion des contraintes locales.
_ L'algorithme EM peut diverger en cas de singularités dans la fonction de vraisemblance. Ce problème peut être réglé par traitement bayésien.

Bibliographie

- ✓ [1] Celeux, G. et Diebolt, D. (1985). The SEM algorithm: a probabilistic teacher algorithm derived from the em algorithm for the mixture problem. Computational Statistics Quarterly, 2(1) :73–82.
- ✓ [2] Celeux, G. et Govaert, G. (1991). A classification EM algorithm for clustering and two stochastic versions Rapport de recherche RR-1364, Inria, Institut National de Recherche en Informatique et en Automatique.
- ✓ [3] Dempster, A. P., Laird, N. M. et Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm (with discussion). Journal of the Royal Statistical Society, B 39 :
- ✓ [4] D. Booth, H. Haas, F. McCabe, and Al. Web services architecture. W3C Working Group Note 11 February 2004, http://www.w3.org/TR/2004/NOTE-ws-arch20040211/,

Dernière consultation Février 2011.

- ✓ [5] H. Cervantes. Vers un modèle à composants orientés services pour supporter la disponibilité dynamique. PhD thesis, Université Joseph Fourier - Grenoble 1, Mars 2004.
- ✓ [6] K. Heather. Web Service Conceptual Architecture (WSCA 1.0). IBM Software Group, Mai 2001. http://www-3.ibm.com/software/solutions/webservices/pdf/WSCA.pdf, Dernière consultation Février 2011.
- ✓ [7] K. Hubert and V. Monfort. Les Web Services techniques, démarches et outils XML, WSDL, SOAP, UDDI, Rosetta, UML. edDunod, Mars 2003.
- ✓ [8] P.Kellert and T. Toumani. Les Web Services Sémantiques. Action spécifique 32 CNRS/STIC, Octobre 2003. http://www.irit.fr/journal-i3/hors_ série/annee2004/revue_i3_hs2004_01_07.pdf, Dernière consultation Février 2011.
- ✓ [9] S. Bhiri. Approche Transactionnelle pour assurer des compositions fiables de services web. PhD thesis, Université Henri Poincaré - Nancy 1, Octobre 2005.
- ✓ [10] J. De Roey. Web Services: Beyond the Peer-to-Peer architecture. PhD thesis, Université libre de Bruxelles, 2007.
- ✓ [11] P.Kellert and T. Toumani. Les Web Services Sémantiques. Action spécifique 32 CNRS/STIC, Octobre 2003. Http://www.irit.fr/journal-i3/hors_serie/annee2004/revue_i3_hs2004_01_07.pdf, Dernière consultation Février 2011.
- ✓ [12] K. Gottschalk, S. Graham, and H. Kreger. Introduction to Web services architecture. IBM Systems Journal, 41 (2) :170–177, 2002.