Ministère de l'Enseignement Supérieur et de le Recherche Scientifique

Université Saad Dahlab de Blida

Faculté des sciences

Département d'informatique

Mémoire Présenté par

Nabi Khaled
Medjeber Ahmed

En vue d'obtenir le diplôme de Master

Domaine : Mathématique et Informatique. MI
Filière : Informatique.
Option : Ingénierie des logiciels.

Titre

La conception et la réalisation d'une architecture orientée service pour la méthode ELECTRE I, cas d'étude : le processus réponse aux appels d'offre.

Proposé et encadré par :
Mme BOUMAHDI Fatima

Soutenue le : 17 Septembre 2012, devant le jury composé de :

- Mme Rezoug	Présidente
- Mme Fareh	Examinatrice
- Mlle Guesmia	Examinatrice

- promotion 2011/2012-

ملخص

البنية الخدمية هي أسلوب هندسة البرمجيات التي تكون من خلالها العمليات المهنية للشركة مشكلة من مكونات برمجية، و التي تنظم بدورها مهام الشركة بين موظفيها ، وبين طلب مكونات الخدمة بهدف التشغيل.

العمل المقدم في هذه المذكرة يقترح بنية خدمية التي تدعم التشغيل الآلي للعملية المهنية باستخدام طريقة دعم اتخاذ القرار المناسب في عملية المناقصة التي تقوم بها الشركة.

الحل المنفذ يسمح للمورد بتقديم عروض توصل عن طريق الانترنت، ومن ثم تتم معالجة تلك العروض بصفة تلقائية. مع إضفاء الطابع الرسمي على هذه العملية والبنية التي يقوم عليها، سوف يقدم هذا البرنامج لمستخدميه الاستقلالية في المنظمة، رؤية أفضل للعملية، بالإضافة إلى الميزة الرئيسية و هي تطور و مرونة النظام الخاص بهم.

الكلمات الرئيسية :البنية الخدمية ، خدمة الويب، استخدام القرار متعدد المعايير، عملية المناقصة، وسير العمل.

Résume

L'Architecture Orientée Service est un style d'architecture logicielle pour lequel les processus métiers de l'entreprise sont des composants logiciels paramétrables, orchestrant des tâches avec les acteurs de l'entreprise et des appels à des composants de services pour s'exécuter.

Le travail présenté dans ce mémoire propose une architecture orientée service, prenant en charge l'automatisation de processus métier RAO en utilisant la méthode d'aide à la décision Electre (le processus **R**éponse aux **A**ppels d'**O**ffre « RAO »).

La solution mis en œuvre permettra à un fournisseur de faire des soumissions en ligne et de traiter ensuite ces propositions de façon automatisée. Grâce à la formalisation de ce processus et à l'architecture sur laquelle il est basé, cet outil offrira à ses utilisateurs une autonomie dans l'organisation, une meilleure vue sur le processus ainsi qu'un atout majeur qu'est l'évolutivité et la flexibilité de leur système.

Mots clés: Architecture Orientée Services, Services Web, l'aide à la décision multicritère, processus appel d'offre, et flux de travail (workflow).

Abstract

Service Oriented Architecture is a style of software architecture for which business processes of the enterprise are configurable software components, orchestrating tasks with actors in the company and calls for service components to run.

The work presented in this thesis proposes a service oriented architecture that supports the automation of business processes call for tender using the method of decision support Electre (Reply to process call for tender).

The solution implemented to allow a supplier to bid online and then process these proposals in an automated fashion. With the formalization of this process and the architecture on which it is based, this tool will provide its users with autonomy in the organization, a better view of the process as well as a major asset and what scalability the flexibility of their system.

Keywords: Service Oriented Architecture (SOA), Web services, the multicriteria decision, for tenders process, and workflow.

Remerciements

C'est avec l'aide de Dieu qu'a vu le jour ce présent travail.

Ensuite, il n'aurait pas pu être achevé sans le soutien, les conseils et les encouragements de certaines personnes auxquelles nous tenons ici à exprimer nos sincères remerciements.

Tout d'abord, notre sincère reconnaissance et notre sincère gratitude à Madame BOUMAHDI Fatima, notre promotrice maître assistante à l'Université SAAD DAHLAB de Blida, pour nous avoir proposé le sujet, pour sa disponibilité, pour ses précieux conseils, sa confiance qu'elle nous a toujours témoigné et sa sollicitude dont elle nous a entouré, et ce tout au long de l'élaboration du présent travail.

Nous tenons à remercier tout particulièrement Mlle GUENDOUZ Amina et son amie GUESMIA Khalida pour tous leurs aides.

Egalement nous tenons à remercier chaleureusement nos enseignants de l'Université SAAD DAHLAB de Blida, pour le savoir qu'ils nous ont transmis toute au long de nos cursus universitaire.

Un grand merci à tous ceux qui n'ont épargné le moindre effort, de près ou de loin, pour nous permettre d'accomplir notre projet.

Pour conclure, Nous adressons nos remerciements les plus respectueux au jury qui ont accepté d'évaluer notre travail.

Table des matières

Liste des tableaux

Liste des figures

Introduction générale

1. Introduction

Le but des appels d'offres est de mettre des sociétés en concurrence pour fournir un produit ou un service. Les appels d'offres sont vitaux pour les sociétés car ils permettent de réaliser de nombreuses économies. Les systèmes d'aide à la décision sont également un avantage supplémentaire dans le cadre des appels d'offres vu que la plate-forme proposée permet aux entreprises de passer ces derniers sans aucun effort ou contribution. Les fournisseurs doivent avoir répondu à l'appel d'offre.

Les processus métiers se répartissent souvent sur plusieurs applications. L'intégration des informations et des processus est donc nécessaire pour obtenir une vue globale d'un processus métier complexe. Jusqu'à présent, cela nécessitait des interventions manuelles laborieuses ou le développement de solutions spécifiques, rigides et difficiles à gérer.

De ce fait a émergé la notion de l'architecture orientée services (SOA).

L'objectif de ce travail est la mise en œuvre d'une architecture orientée service, prenant en charge l'automatisation de processus métier RAO en utilisant la méthode d'aide à la décision Electre (le processus **R**éponse aux **A**ppels d'**O**ffre « RAO »).

2. Présentation du sujet

2.1. Problématique

Notre problème principal de ce mémoire est d'analyser et de comprendre les méthodes d'aide à la décision multicritère ainsi que le fonctionnement et les enjeux des SOA, pour notre étude de cas RAO afin de concevoir un prototype d'application web.

Nous souhaitons l'automatisation de choix de meilleure soumission. En effet, nous avons proposé de mettre en place un portail destiné aux fournisseurs, ce portail permet à un fournisseur d'émettre manuellement ces réponses aux appels d'offres via des formulaires proposés par le portail.

2.2. Objectifs

L'objectif de notre projet est donc le développement d'une application orientée service pour le processus réponse aux appels d'offre, on prend en considération l'aspect d'aide à la décision au niveau de la conception et la réalisation. Cette application est composée de plusieurs services web assurant la gestion et la prise en charge des tâches administratives des différents intervenants lors de traitement des appels d'offres, ainsi que d'un service web général d'orchestration de ces différents services.

Pour répondre à ces objectifs, il est proposé de s'appuyer sur une architecture SOA. Pour bénéficier des avantages de cette approche, notamment :

- Mise en place de processus automatisés flexibles.
- Réutilisation des processus et des services quel que soit le canal d'envoi d'une demande de service.
- Intégration avec les systèmes existants via des services spécifiques.

Notre solution se base sur Electre, pour intégrer un aspect d'aide à la décision à notre processus RAO. Le choix de Electre est justifie par :

- Elle améliore la transparence du processus de décision.
- Elle définit, précise et met en évidence la responsabilité du décideur.

2.3. Organisation du mémoire

Ce mémoire est composé d'une introduction générale suivie de deux parties, une partie théorique et une autre pratique.

La première partie sera consacrée à l'étude théorique des grands concepts rencontrés lors de la réalisation de notre projet, structurée en trois chapitres. Le premier chapitre va traiter l'architecture orientée service et à la technologie des services web cœur de notre travail, le second s'intéressera à les méthodes d'aides à la décision et plus particulièrement les méthodes. Enfin, le troisième chapitre présentera l'approche de développement basée sur les modèles MDA, étant le domaine d'application de notre étude.

Après avoir donné une idée sur les concepts utiles pour la réalisation de notre projet, nous aborderons dans la deuxième partie, le développement de notre solution suivant la méthode MDA, adoptée pour la mise en œuvre d'une architecture orientée service, en utilisant les langages de modélisation UML, BPMN et SOAML. En dernier, nous passerons à l'implémentation de la solution retenue au niveau de la conception après une description de l'environnement du travail. Nous achèverons cette partie avec la présentation de l'application réalisée.

Puis une conclusion générale et des perspectives viendront clôturer ce mémoire.

Chapitre I

Architecture Orientée Service

1. Introduction

Les systèmes informatiques se sont développés de manière exponentielle, sous forme d'architectures logicielles de plus en plus complexes et difficiles à gérer pour les entreprises. Les architectures traditionnelles ont atteint leurs limites en termes de capacité, alors que les besoins traditionnels des organisations informatiques demeurent.

En effet, les organisations sont confrontées à un environnement qui les oblige à innover et à s'adapter sans cesse et à chaque fois de plus en plus vite pour réussir et même pour survivre. Dans cette environnement, il devient fondamental de faire communiquer et travailler ensemble les différents éléments informatiques, souvent hétérogènes car composés des matériels, des logiciels et des applications d'origines très diverses; d'autre part, les entreprises doivent réaliser ces objectifs au moindre coût.

La solution est de mettre en place une architecture bien définie permettant d'interconnecter les différentes applications de l'entreprise afin de remédier au grand problème d'hétérogénéité de son système d'information. L'architecture orientée service met en œuvre cette solution.

L'objectif de ce chapitre est de proposer une définition synthétique et claire des notions cachées derrière les trois lettres « SOA ».

2. Qu'est ce que SOA ?

L'architecture orientée service en anglais « Service Oriented Architecture SOA » est une philosophie d'organisation et de gestion des systèmes d'information de l'entreprise regroupant les bonnes pratiques issues des évolutions de l'ingénierie logicielle.

3. Pourquoi SOA ?

L'architecture orientée services est une approche architecturale de dernière génération qui permet de passer d'une vision « application» à une vision « services ».

Dans la SOA, les applications d'une entreprise fournissent à d'autres applications des services et non pas des données.

L'idée sous-jacente est de cesser de construire la vie de l'entreprise autour d'applications pour faire en sorte de construire une architecture logicielle globale décomposées en services correspondant aux processus métiers de l'entreprise.

Cette approche permet de définir les processus indépendamment des applications, elle autorise la définition et la modification des processus métiers indépendamment des applications informatiques.

L'architecture orientée service se base sur les principes suivants **[Gilbert Raymond, 07]**:

- **Diviser pour régner :** Substituer la découpe strictement applicative par une structuration en composants plus réduits et potentiellement plus simples à faire évoluer.
- **Alignement métier :** Construire et organiser le système à partir des réalités métiers, qui doivent se retrouver dans ses constituants.
- **Neutralité technologique :** Assurer une indépendance totale entre les interfaces et les implémentations. L'élément qui utilise un service ne doit pas être contraint ni par la technologie d'implémentation, ni par sa localisation (potentiellement distribué).
- **Mutualisation :** Favoriser la réutilisation de services métiers par plusieurs lignes métiers ou applications. Permettre la construction de services de haut niveau par combinaison de services existants.
- **Automatisation des processus métier :** Isoler la logique des processus métiers sur des composants dédiés qui prennent en charge les enchaînements et les échanges de flux d'information.
- **Echanges orientés document :** Les informations échangées par les services possèdent une structure propre, guidée par les besoins métiers.

4. Orientation service

L'électricité est un exemple typique d'approche orientée service. Les consommateurs ignorent comment l'électricité a été produite (source hydraulique, thermique, nucléaire) et acheminée jusqu'à eux. De même, les producteurs ne prêtent pas attention au type d'appareil branché (ordinateur, télévision, radiateur, lampe, etc.). Par contre les caractéristiques du service sont définies : le voltage (220 V) et la fréquence (50 Hertz). Une interface standardisée permet la connexion entre le producteur et le consommateur (la prise et la fiche électrique).

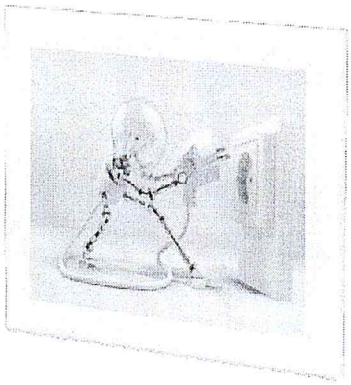


Figure.1 : Exemple d'approche orientée service : l'électricité. [Bernard Dubs,07]

Cette orientation service ne s'intéresse pas à la production ou à la consommation du service mais seulement à l'interface : comment le producteur propose son service aux consommateurs et comment le consommateur découvre les services disponibles et s'y connecte. L'interface entre les deux partenaires est définie de manière stricte, chacun peut organiser son environnement interne à sa guise. **[Bernard Dubs,07]**

L'implémentation du fournisseur de service est donc libre de changer sans qu'il y ait un impact sur son utilisation. En général, le client s'intéresse uniquement au résultat produit du service sans avoir le besoin ni le souci de savoir comment ce dernier est obtenu.

Alors on peut voir ce service comme une boîte noire : on sait qu'elle va rendre le service voulu sans savoir comment est faite la boite noire. On peut choisir de la remplacer par un autre service implémenté différemment mais répondant aussi à la même fonctionnalité.

Avec SOA, le concept d'application, tel qu'on le connaît depuis longtemps, change radicalement pour laisser la place au concept de composition de services ou aux applications composées. En effet, il ne s'agit pas de construire une application spécifique pour chaque exigence métier qui vient d'être identifiée mais plutôt de composer des nouvelles configurations de services permettant de satisfaire les besoins utilisateur.

Donc, on peut dire qu'un service est un module logiciel contenant une interface publique affichant ses fonctionnalités, et une implémentation dont les détails ne sont pas connus des clients.

Une architecture orientée service consiste donc en une collection de services, indépendants les uns des autres, qui interagissent et communiquent entre eux.

5. Les caractéristiques générales d'un service

Un service est un traitement qui respecte les quatre propriétés que nous détaillons ci-après:

- **Couplage faible**

L'échange entre le fournisseur de service et le consommateur doit se faire à travers des messages (couplage lâche vis-à-vis de son environnement).

L'utilisation d'une orchestration évite que les services aient besoin de connaître les autres services. **[Mickaël Baron,10]**

- **Activable à distance et interopérable**

Un service expose une interface d'utilisation qui est la même indépendamment de sa localisation sur les réseaux. L'appel au service fonctionne quel que soit le langage et le système d'exploitation du consommateur. **[Pierre Bonnet,05]**

- **Respecte le pattern d'architecture SOA**

Le pattern d'architecture SOA consiste à créer une architecture applicative qui décompose les traitements sous la forme de services rattachés à des paquets de classes. Ces paquets forment des catégories (objet métier, sujet métier), chacune dotée d'une façade d'accès qui contient l'ensemble des services qu'elle expose .Un service a le droit d'interagir uniquement avec les classes de sa catégorie. **[Pierre Bonnet,05]**

- **Expose un contrat d'utilisation**

Le contrat de service joue un rôle majeur, il détaille les conditions d'utilisation du service sous forme de pré et post conditions, protocoles, et contraintes non fonctionnelles.

En effet, la simple spécification fonctionnelle des opérations de services ne suffit pas à garantir le fonctionnement correct du système déployé. Les contraintes non fonctionnelles, comme le temps de réponse ou le nombre d'invocations par seconde permettent de fixer les termes du contrat opérationnel entre consommateur et fournisseur de service. **[Bernard Dubs,07]**

6. Cycle de vie des services

↳ Identifier les services à mettre en place

L'identification des services est une étape clef pour le succès de SOA. Un service est identifiable et n'a pas de réelle justification que s'il satisfait au moins un des trois critères suivants :

- ❑ Le service est mutualisé au sein d'une forte population d'application consommatrices, informatiquement, il permet la réutilisation
- ❑ Le service facilite l'intermédiation avec un fonctionnement existant interne à un SI, informatiquement, il permet l'interopérabilité offerte via internet à d'autre SI.
- ❑ Le service est nécessaire à un enchainement plus global d'activité via un processus métier, informatiquement, il autorise son emploi dans une composition.

↳ Mettre en place les services

Puisqu'il doit relier concrètement des processus métier avec des application et progiciels évolutifs et repartis, un service n'est pas uniquement un concept : il existe bien, en tant que composant logiciel exécutant une tâche spécifique plus ou moins ambitieuse pour le consommateur.

La mise en place d'un service distingue donc d'une part la modélisation du contrat qui décrit le service rendu, d'autre part l'implémentation du service, qui doit respecter le contrat défini, et enfin les modalités de déploiement.

Maintenir les services

La gestion des évolutions d'un service est un point clef, qui passe par la gestion de versions de composants logiciels.

Cette gestion n'est pas en soi une nouveauté apportée par SOA, cependant, les services étant la pierre angulaire de l'approche, SOA impose une mise en place rigoureuse de cette gestion de version.

7. La typologie des services

Service métier

Le rôle d'un service métier est d'offrir un ensemble cohérent de traitement métier. Il peut être soit un service d'accès à des informations, soit un service de calcul et vérification de règles métier, soit une composition des deux. **[Xavier Fournier-Morel,06]**. Donc, un service métier est la représentation d'une activité métier élémentaire ou complexe.

Le contrat d'utilisation du service métier, c'est-à-dire son interface, peut donc être plurielle puisque plusieurs opérations peuvent être exposées. Ceci à la différence du service issu du pattern d'architecture applicative qui n'expose qu'un seul et unique traitement.

Service technique

En plus des services métier, nous pouvons également trouver des services dits « technique », qui permettent de façadier et de mutualiser des fonctionnalités non-métier.

Le rôle d'un service technique est de donner accès à une ressource technique donnée. Il est très souvent générique : il n'est pas lié particulièrement à une ressource mais à une catégorie de ressources. **[Xavier Fournier-Morel,06]**

Un service métier peut s'appuyer sur un ou plusieurs services techniques pour exécuter ses traitements.

8. Application composite

La notion d'application composite fait référence aux applications construites par combinaison de services multiples. Cela se présente dans les offres que propose certaine entreprise à leurs collaborateurs et partenaires des environnements de travail de plus en plus personnalisés pour faciliter les échanges et le partage d'information. Cela va pleinement bénéficier de l'approche SOA, qu'il s'agisse de la mise à disposition rapide de nouvelle fonctions, de l'intégration.

L'approche SOA favorise la construction de nouveaux services par composition de services existants et cette composition devient à son tour un service. De plus la composition de service ne s'arrête pas non plus aux frontières du SI.

Dans une approche SOA, un service peut être consommé soit **[Xavier Fournier-Morel,06]** par :

- ⇨ Un utilisateur, via une application composite interactive ;
- ⇨ Un système traditionnel comme un batch ;
- ⇨ Un processus métier ;
- ⇨ Ou autre service SOA.

9. Les niveaux d'une SOA

L'architecture SOA est représentée par un modèle en couches, voir la figure suivante :

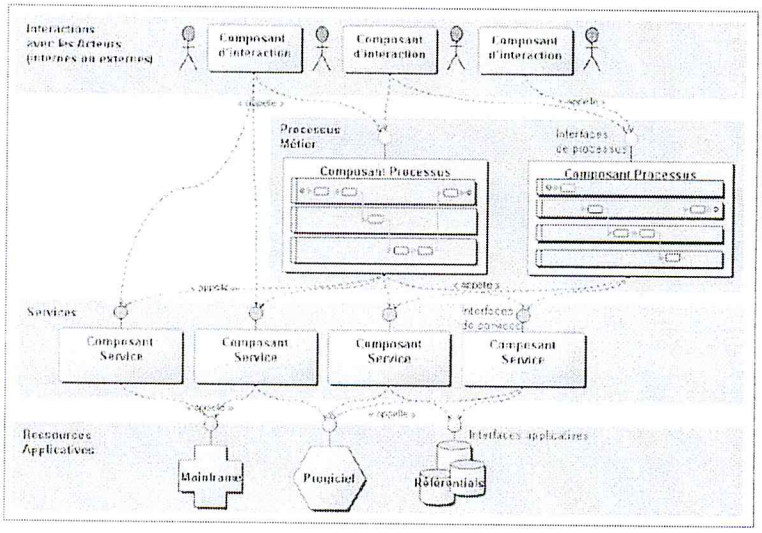


Figure.1 : Les niveaux d'une SOA. [Yann Letanou,07]

➲ **Interaction avec les acteurs**

Ensembles des composants permettant aux acteurs d'interagir avec le système d'information. Les acteurs sont des utilisateurs, pour lesquels on mettra typiquement en place des solutions de type portail permettant de traiter la diffusion multi-canal de l'information, ou des systèmes externes dans le cadre d'échange B2B.

➲ **Processus métier**

L'ensemble des composants logiciels permettant de réaliser les processus métier de l'entreprise par une orchestration de tâches automatiques ou utilisateurs.

➲ **Services**

L'ensemble des composants permettant de réaliser les services de l'entreprise et peut combiner plusieurs services de granularité plus fine.

➲ **Ressources applicatives**

Peuvent être des progiciels propres à l'entreprise, les basses de données, des applications existants, etc.

10. Technologies de mise en œuvre d'une SOA

Les services représentent, bien évidement, le cœur d'une SOA. Il est donc logique de trouver des standards concernant l'exposition et l'invocation de services. Le plus important de ces standards, est celui des Web Services (WS).

➢ Les Services Web

Les Services Web sont une nouvelle voie dans le développement des logiciels, ils s'imposent graduellement comme les connecteurs standards applicatifs. Il existe de multiples définitions des Services Web. Celle qui nous paraît la plus appropriée dans ce contexte est :

Un Service Web est un composant métier ou technique accessible par des protocoles standards. **[Tanguy Crusson,08]**

Cette technologie est la première technologie d'intégration faisant abstraction des détails d'implémentation. Les Services Web donc peuvent être écrits dans n'importe quel langage et exécutés sur n'importe quelle plate-forme. Un client d'un Service Web peut également être écrit dans n'importe quel langage et exécuté sur n'importe quelle plate-forme.

Un service est composé de deux parties : une interface et une implémentation.

L'interface est standard ; l'implémentation quant à elle est propriétaire, et dépend de la plateforme utilisée : cela a peu d'importance à partir du moment où l'interface est respectée, et que cette implémentation dialogue avec ses clients par échange de messages XML.

L'implémentation du Service Web accède aux applications du système d'information, et permet donc d'encapsuler leurs fonctionnalités sous formes de services accessibles depuis toute plateforme.

Le fonctionnement des Services Web repose sur un modèle en couches, dont les trois couches fondamentales sont les suivantes :

- **Description**

Un Service Web n'est utilisable que si d'autres personnes peuvent découvrir ce qu'il fait et comment l'appeler. De plus, les développeurs doivent posséder suffisamment d'informations sur un Service Web pour pouvoir écrire un programme client qui l'appelle.

Pour permettre à un client de consommer un service web, ce dernier a besoin d'une description détaillée du service avant de pouvoir interagir avec lui. Un WSDL (Web Service Description Langage) fournit cette description dans un document XML.

WSDL est un document XML qui décrit de manière indépendante de tout langage un Service Web pour permettre l'appel de ses opérations et l'exploitation des réponses (les paramètres, le format des messages, ...).

- **Publication et découverte**

Une organisation a besoin de publier les Services Web qu'elle possède, pour les autres organisations pour qu'elles puissent les découvrir.

Universal Description, Discovery and Integration (UDDI) est utilisé pour publier et rechercher des services web.

Il s'agit d'une spécification de registre distribué d'informations sur des Web Services. Lorsqu'un Service Web a été développé et qu'un document le décrivant a été créé, il doit y avoir un moyen de mettre les informations à la disposition des utilisateurs désirant utiliser le Service Web décrit.

La publication d'un Service Web dans un registre UDDI permet aux utilisateurs de rechercher et d'apprendre l'existence du Service Web.

- **Invocation**

Une fois l'organisation a découvert le Service Web via l'interface UDDI, et elle a prit la décision d'utiliser ce service dans son application, elle a besoin d'invoquer le Service Web.

Pour échanger des messages entre un Service Web et une application appelante, on utilise les messages SOAP (*Simple Object Access Protocol*). Ces messages sont des documents XML transportés via le protocole HTTP. En réalité, il s'agit d'un important composant de base pour développer des applications distribuées.

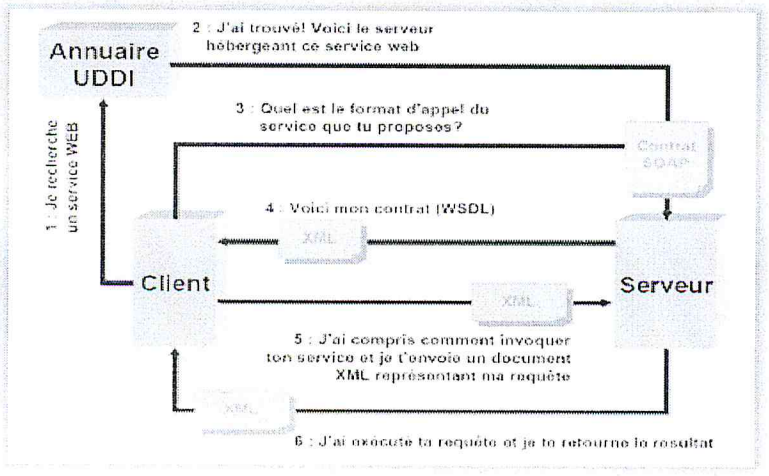


Figure.2 : Architecture Service Web. [Jeremy Fierstone,02]

- Attention à ne pas confondre les deux !

  - Une SOA peut se mettre en œuvre sans les Services Web.
  - On peut utiliser les Services Web sans faire de SOA.

Mais Les Services Web constituent la meilleure solution standardisée disponible pour mettre en œuvre une SOA.

11. Conclusion

L'Architecture Orientée Service a apporté une nouvelle manière de penser, de concevoir les architectures des systèmes d'information.

SOA est une réponse très efficace pour les problématiques que rencontrent les entreprises en termes de réutilisabilité, d'interopérabilité et de réduction de couplage entre les différentes applications qui implémentent leurs systèmes d'informations. Elle est devenue un mot clef et un concept incontournable pour les développements de projets d'envergure.

Dans ce chapitre, nous avons présenté les concepts de bases de l'Architecture Orientée Service et les technologies principales permettant de la mettre en œuvre, en découvrant leurs caractéristiques, leurs intérêts et leurs modes d'emploi qui seront utiles par la suite pour la conception et le développement de notre projet.

Chapitre II

La méthode ELECTRE

1 Introduction

L'aide à la décision multicritère se présente comme une alternative aux méthodes d'optimisation classiques basées sur la définition d'une fonction unique, souvent exprimée en terme économique (monétaire) et qui reflète la prise en compte de plusieurs critères, souvent incommensurables. L'intérêt des méthodes multicritères est de considérer un ensemble de critères de différentes nature (exprimés en unité différentes), sans nécessairement les transformer en critères économiques, ni en une fonction unique. Il ne s'agit pas de rechercher un optimum, mais une solution compromis qui peut prendre diverses formes : choix, affectation ou classement. Plusieurs méthodes existent dans la littérature, dans le cadre de ce cours nous allons définir le cadre théorique et les aspects méthodologies des méthodes multicritères, ensuite nous allons illustrer leurs approches en étudiant les types de méthodes multicritère : Electre I, Electre II, Electre III, Electre IV, Electre-tri et Electre IS. **[Amir NAFI , Caty WEREY.10]**

2 L'aide à la décision et les méthodes multicritère

D'après **[ROY, 1985]** « L'aide à la décision est l'activité de celui qui, prenant appui sur des modèles clairement explicités mais non nécessairement complètement formalisés, aide à obtenir des éléments de réponses aux questions que se pose un intervenant dans le processus de décision, éléments concourant à éclairer la décision et normalement à prescrire, ou simplement à favoriser un comportement de nature à accroître la cohérence entre l'évolution du processus d'une part, les objectifs et le système de valeurs au service desquels cet intervenant se trouve placé d'autre part. ». L'aide à la décision est donc un processus qui utilise un ensemble d'informations disponibles à un instant donné, afin de formuler un problème et aboutir à une décision sur un objet précis. Dans le cadre de la décision multicritère, l'objet de la décision est formé par un ensemble d'actions ou alternatives. Pour **[Roy, 1996]** les problèmes réels peuvent être formulés à l'aide des méthodes d'analyse multicritère, selon trois formulations de bases : problématique de choix, notée $P\alpha$, la problématique de tri ou d'affectation notée $P\beta$ et la problématique de rangement noté $P\gamma$. Voir Tableau 1.

Problématique	Objectif	Résultat
$P\alpha$	Eclairer la décision par le choix d'un sous- ensemble aussi restreint que possible en vue d'un choix final d'une seule action. (optimums et satisfecums)	Un **choix** ou une procédure de sélection.
$P\beta$	Eclairer la décision par un tri résultant d'une affectation de chaque action à une catégorie, les catégories étant définies a priori en fonction des normes ayant trait à la suite à donner aux actions qu'elles sont destinées à recevoir	Un **tri** ou une procédure d'affectation
$P\gamma$	Eclairer la décision par un rangement obtenu en regroupant tout ou partie (les « plus satisfaisantes ») des actions en classes d'équivalence, ces classes étant ordonnées, de façon complète ou partielle, conformément aux préférences	Un **rangement** ou procédure de classement
$P\delta$	Eclairer la décision par une description, dans un langage approprié, des actions et de leurs	Une **description** ou une procédure cognitive

Tableau 1. Identification des types de problématique [Amir NAFI , Caty WEREY.10]

Le processus de décision multicritère peut être décrit par la Figure 4. Il est caractérisé par 4 étapes essentielles :

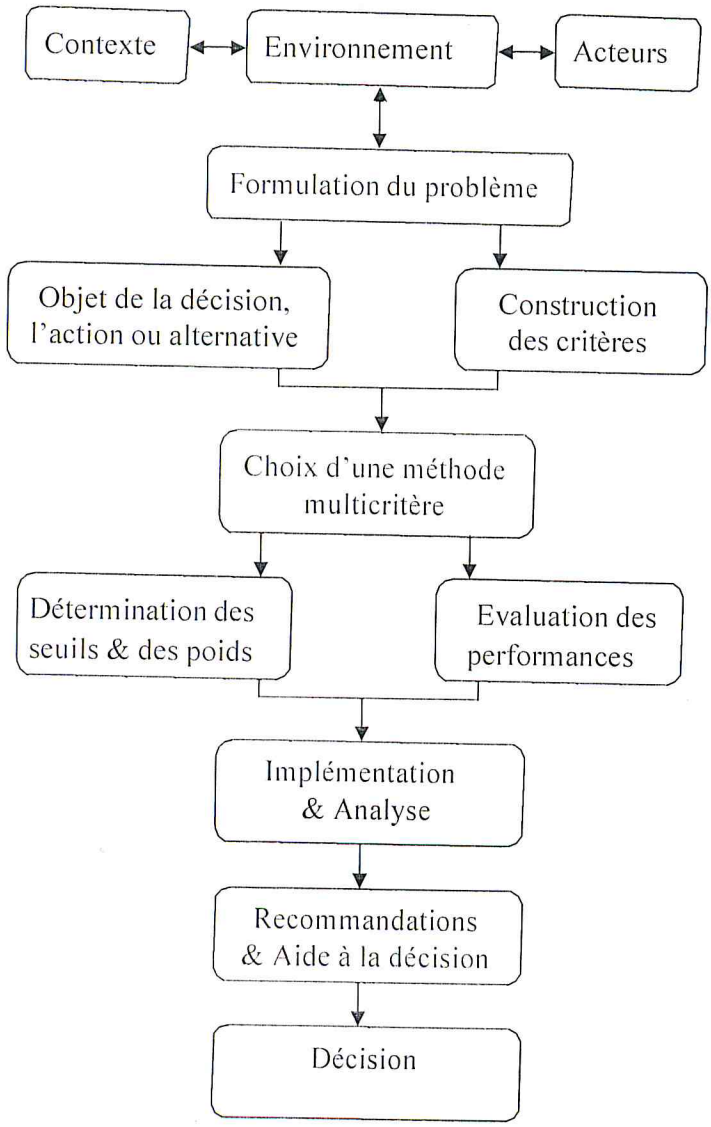


Figure.4 : Le processus de décision multicritère [Amir NAFI , Caty WEREY.10]

2.1 Définition du problème et l'objet de la décision, l'action

La définition du problème requiert une compréhension de la situation étudiée, du contexte et des acteurs impliqués dans la prise de décision. L'interaction avec les différents acteurs permet de comprendre le processus de décision, les enjeux, l'objet de la décision et la nature de la décision à prendre. Il s'agit donc de définir la nature du problème posé en le formulant soit en une problématique de choix, de tri ou de rangement. La détermination de l'objet de la décision consiste à identifier l'ensemble des actions ou alternatives sur lesquelles va porter la décision.

« Une action a est la représentation d'une éventuelle contribution à la décision globale susceptible, eu égard à l'état d'avancement du processus de décision, d'être envisagée de façon autonome et de servir de point d'application à l'aide à la décision». C'est l'objet de la décision.

Une action est dite globale, si, dans sa mise en exécution, elle est exclusive de toute action introduite dans le modèle ; dans le cas contraire, elle est dite fragmentaire. Une action potentielle est une action réelle ou fictive provisoirement jugée réaliste par un acteur au moins. On note par A est l'ensemble des actions potentielles.

2.2 L'analyse des conséquences et détermination des critères

Il s'agit en effet d'identifier et mesurer les conséquences des actions sur lesquelles va porter la décision. Les critères découlent des conséquences des actions. Souvent, une action a plusieurs conséquences, ainsi la conséquence d'une action selon un critère donné est évaluée par une fonction g (à valeurs réelles) définies sur l'ensemble A des actions potentielles de telle sorte qu'il soit possible de raisonner ou de décrire le résultat de la comparaison de deux actions a et b relativement à partir des nombres g(a) et g(b). L'évaluation de l'action sera donc effectuer sur un ensemble de critères. On distingue le vrai-critère et le pseudo-critère.

Pour le vrai critère, en considérant deux actions a et b à comparer, deux situations sont possibles :

$$g(b) = g(a) \Leftrightarrow b\ I_g\ a \text{ (indifférence)}$$
et
$$g(b) > g(a) \Leftrightarrow b\ P_g\ a \text{ (préférence stricte)}$$

C'est une vision peu réaliste car une simple différence g(b) - g(a) n'est pas significative d'une préférence stricte.

Pour le pseudo-critère on associe à la fonction critère g deux fonctions seuils $q_g(g(a))$ exprimant un seuil d'indifférence et $p_g(g(a))$ exprimant un seuil de préférence.

$$g(b) \geq g(a) \Rightarrow b\ S_b\ a$$

Sb : « aussi bon que » ou, S est une relation de surclassement, c'est à dire que b est au moins aussi bon que a sur une majorité de critères sans être vraiment plus mauvais relativement sur les autres critères. On dira dans se cas que b surclasse a, on notera b Sb a. On introduit des seuils (constants ou fonction de g) tels que :

$$g(b) - g(a) \quad q_g(g(a)) \Leftrightarrow b\ I_g\ a$$
$$p_g(g(a)) < g(b) - g(a) \Leftrightarrow b\ P_g\ a$$

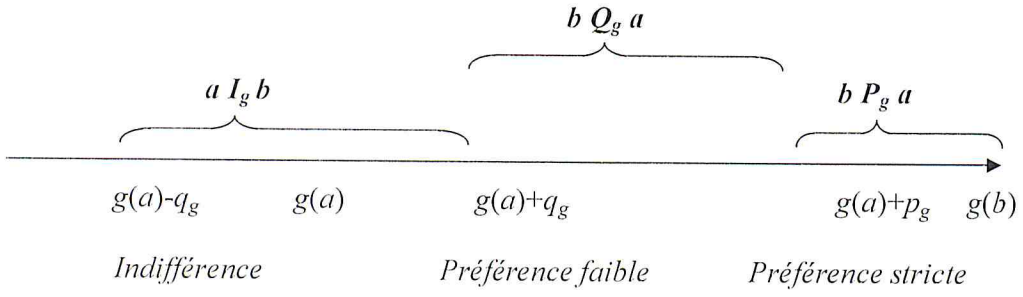
Où q_g est un seuil dit d'indifférence et p_g un seuil dit de préférence.

La situation non couverte par ces deux éventualités, à savoir :

$$q_g(g(a)) < g(b) - g(a) \leq q_g(g(a))$$

correspond à une situation d'hésitation (indétermination) entre l'indifférence et la préférence stricte appelée préférence faible et notée Q_g.

Ce qui peut se traduire ainsi :



Ces différentes situations sont reprises dans le Tableau 2.

Situation	Définition	Relation binaire (propriétés)
Indifférence	Elle correspond à l'existence de raisons claires et positives qui justifient une équivalence entre deux actions.	***I* : relation symétrique réflexive**
Préférence stricte	Elle correspond à l'existence de raisons claires et positives qui justifient une préférence significative en faveur de l'une (identifiée) des deux actions.	***P* : relation symétrique (irréflexive)**
Préférence faible	Elle correspond à l'existence de raisons claires et positives qui infirment une préférence stricte en faveur de l'une (identifiée) des deux actions. Mais ces raisons sont insuffisantes pour en déduire soit une préférence stricte en faveur de l'autre, soit une indifférence entre ces deux actions.	***Q* : relation asymétrique (irréflexive)**
Incomparabilité	Elle correspond à l'absence de raisons claires et positives justifiant l'une des trois situations précédentes	***R* : relation symétrique irréflexive**

Tableau 2. Situations possibles lors de la comparaison de deux actions [Amir NAFI , Caty WEREY.10]

La construction des critères est une étape délicate qui nécessite une compréhension du problème posé et une interaction avec les acteurs impliqués dans la prise de décision. Il s'agit d'identifier les enjeux et la nature des conséquences possibles sur l'objet de la décision, c'est à dire les actions considérées. La définition des critères nécessite par la suite une évaluation de la contribution et l'influence de chaque critère dans la décision finale. Ceci ce traduit par des pondérations qui sont définies par les acteurs impliqués ou bien obtenus par un processus itératif suite à l'interaction avec les acteurs concernés.

Les critères à retenir pour juger quelle est l'action préférée, doivent présenter les conditions suivantes :

- L'aide multicritère à la décision doit permettre de juger de l'intérêt économique des différentes actions entre elles. Il s'agit donc de construire une famille de critères qui puissent représenter, d'une façon aussi proche que possible, les coûts et les avantages des actions, bénéfices.

- Les critères doivent être d'une part, suffisamment nombreux et précis pour bien discriminé entre elles les différentes actions ; d'autre part, ne pas être redondant pour éviter de majorer l'importance attribuée à une dimension d'analyse.
- Les critères peuvent être de nature différente. On définit des familles de critères : économiques, sociaux, environnementaux, techniques. Chaque famille de critères peut contenir un ou plusieurs critères.

Les critères doivent également vérifier des axiomes :

- Axiome d'exhaustivité : si deux actions ont les mêmes vecteurs performances (mêmes conséquences pour tous les critères) alors il faut être sûr que les acteurs sont bien indifférents entre les deux actions.
- Axiome de cohésion : En partant de deux actions qui sont jugées équivalentes, si l'on accroît la performance de la première sur un critère quelconque, alors elle apparaît « *comme au moins aussi bonne* » que la seconde action inchangée.
- Axiome de non-redondance : un critère est redondant si son retrait de la famille laisse une nouvelle famille vérifiant les deux axiomes précédents.

2.3 Choix d'une méthode d'aide à la décision multicritère

Cette étape dépend de la nature du problème posé. Plusieurs méthodes ont été développées, le Tableau 3 identifie certaines méthodes en fonction de la nature du problème étudié.

	Nature du problème		
Critères	α (sélection)	β (affectation)	γ (classement)
Vrai critère	I	-	II
Pseudo-critère	IS	Tri	III, IV

Tableau 3.Choix de la méthode multicritère [Roger et al., 2000]

2.4 Performance des actions

Lorsque l'analyse des actions a conduit à la construction d'un seul critère, on peut réaliser une optimisation sur ce critère, ce qui peut être simple lorsque le nombre d'action est faible, sinon il faut avoir recours à des outils plus ou moins compliqués.

Dans le cas fréquent, où l'analyse des conséquences des actions potentielles a conduit à construire plusieurs critères, c'est l'analyse multicritère qui permet de donner des réponses au problème posé. Pour chaque action considérée, et pour chaque critère un seuil de préférence p, d'indifférence q et un seuil de veto v sont estimés. Chaque critère se voit attribué un poids k traduisant sa contribution dans la décision finale. Le résultat de l'analyse des conséquences est présenté dans un tableau de performances. Voir Tableau 4.

Critères Poids	g_1 k_1	g_2 k_2	...	g_j k_j	...	g_n k_n
	p_1	p_2	...	p_j	...	p_n
Seuils	q_1	q_2	...	q_j	...	q_n
	v_1	v_2	...	v_3	...	v_n
Actions						
a_1	$g_1(a_1)$	$g_2(a_i)$	...	$g_j(a_1)$	...	$g_n(a_i)$
a_2	.	.		.		.
.	.	.		.		.
.	.	.		.		.
.	.	.		.		.
a_i	$g_1(a_i)$	$g_2(a_i)$	...	$g_j(a_i)$	...	$g_n(a_i)$
.						
.						
.						
a_m	$g_1(a_m)$	$g_2(a_m)$	...	$g_j(a_m)$	...	$g_n(a_m)$

Tableau 4. Tableau de performances

3 L'agrégation des critères et l'analyse multicritère

Dans le cadre de ce cours, on distingue entre l'approche d'agrégation classique et les approches dites de surclassement proposé par les méthodes ELECTRE. L'approche classique se base sur l'agrégation des critères de décision en un critère unique. Elle consiste à bâtir un critère unique de synthèse en utilisant une fonction d'agrégation V en posant : g(a)=V(g1(a), g2(a), g3(a), ..., gn(a)). Deux actions quelconques deviennent ainsi comparables grâce à l'utilisation des différents critères présentés plus haut. Une fois obtenu le critère de synthèse, on peut simplement élaborer

une prescription dans une des trois problématiques : α,β, γ. La fonction d'agrégation V prend généralement une des deux formes :

1. agrégation par somme pondérée : $g(a) = \sum_{j=1}^{n} k_j \cdot g_j(a)$
2. agrégation additive : $g(a) = \sum_{j=1}^{n} k_j \cdot v_j[g_j(a)]$

Les poids k_j sont des coefficients strictement positifs et les V_j n des fonctions monotones strictement croissantes. Il n'est pas restrictif d'imposer $\sum_{j=1}^{n} k_j = 1$ et $0 \leq v_j \leq 1$.

Par opposition à l'approche classique, l'approche dite de surclassement vise à construire sur l'ensemble A une relation de surclassement globale S enrichissant la relation de dominance. Elle vise à modéliser la part des préférences globales que l'on est en mesure d'asseoir de façon suffisamment probante.
La relation S est généralement construite à l'aide d'un test de surclassement appliqué à toutes les paires d'actions de A. Les méthodes ELECTRE utilisent une relation de surclassement pour l'agrégation des critères de décision.

4 Illustration des méthodes multicritère

Nous présentons dans ce qui suit six méthodes multicritères, en fonction de la problématique considérée. La méthode Electre I ou Electre IS, est préconisée pour répondre à une problématique de choix. La méthode Electre-Tri est préconisée pour les problématiques de tri ou d'affectation. La méthode Electre II, Electre III ou Electre IV est utilisée pour répondre à des problématiques de rangement ou de classement.

4.1 La méthode Electre I

Cette méthode proposée par **[Roy,1996]** permet de résoudre les problèmes multicritère de choix. Cette méthode permet d'identifier le sous-ensemble d'actions offrant le meilleur compromis possible. Souvent utilisée dans le choix de projets concurrents, afin d'identifier le sous-ensemble de projets le plus performant sur la base des critères considérés. Dans le cas de la méthode Electre I, on définit de vrai-critères, on retrouve également une notion de concours dans cette méthode; retenir les meilleurs.

On considère un ensemble A de m actions, qui représentent l'objet de la décision, dont le but est d'identifier un sous-ensemble d'actions offrant un meilleur compromis parmi l'ensemble de départ. On définit pour chaque critère une fonction d'évaluation g_j (où j=1 à n, n est le nombre de critères), pour chaque critère, on évalue un poids k_j qui augmente avec l'importance du critère. L'indice de concordance pour deux actions a et b est noté par C(a,b), compris entre 1 et 0, il mesure la pertinence de l'assertion « a surclasse b », comme suit :

$$C(a,b) = \frac{\sum_{\forall j, g_j(a) \geq g_j(b)} k_j}{K} \text{ avec } K = \sum_{j=1}^{n} k_j$$

L'indice de discordance *D(a,b)* est défini par :

$$D(a,b) = 0 \quad \text{si } \forall j, g_j(a) \geq g_j(b)$$

Sinon

$$D(a,b) = \frac{1}{\delta} \max_j \left[g_j(b) - g_j(a) \right]$$

avec δ est la différence maximale entre le même critère pour deux actions donnée.

La relation de surclassement pour Electre I est construite par la comparaison des indices de concordance et de discordance à des seuils limites de concordance $\hat{c}$ et de discordance $\hat{d}$.

Ainsi, *a surclasse b*, si : $aSb \Leftrightarrow C(a,b) \geq \hat{c}$ et $D(a,b) \leq \hat{d}$.

L'algorithme ELECTRE I est représenté par la Figure 5 ci-dessous.

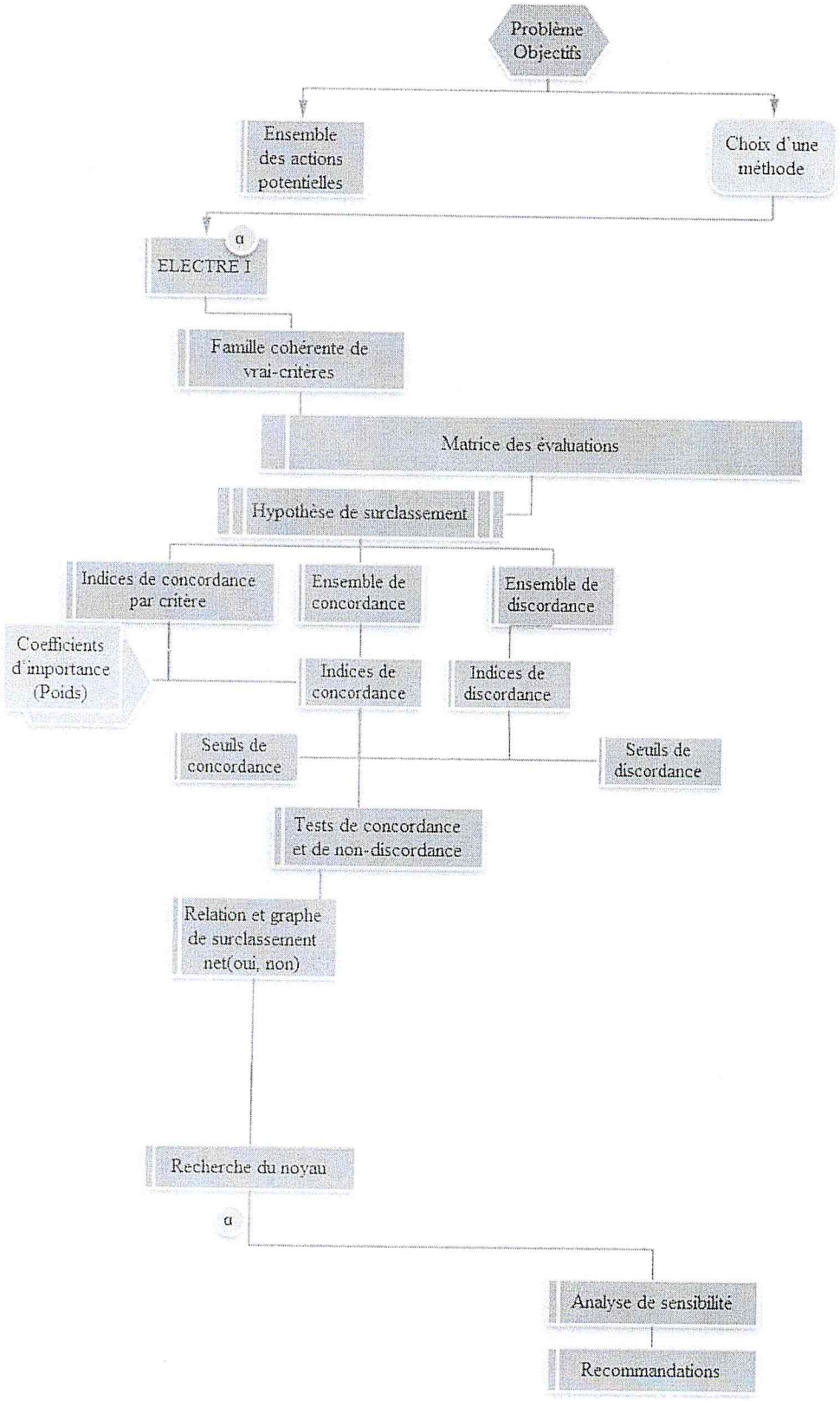


Figure.5 : Un organigramme qui représente la méthode Electre I [Maystre L. Yves,et al.,1996]

4.2 Electre-Tri

Electre-tri est une méthode qui permet de résoudre des problèmes d'affectation, le principe de la méthode est d'assigner un ensemble de m d'alternatives ou d'actions noté A={a1, a2,a3,…,am} sur lesquelles se base la décision à des catégories ou classes bien définies. On note l'ensemble F={1,2,…,n} l'ensemble des indices des critères. Chaque action de l'ensemble A sera évaluée par une fonction réelle, exprimant l'évaluation de l'action pour un critère donné, on note G={g1,g2,…,gn} l'évaluation de l'action pour les critères considérés.

L'importance des critères dans la prise de décision est évaluée par un ensemble de poids K={k1,k2,…,kn}. Par opposition aux autres approches, les alternatives qui constituent l'objet de la décision ne sont pas comparées entre elles, mais à des seuils traduisant la frontière entre les h classes prédéfinies, noté C={C1,C2,…,Ch}. Chaque alternative sera comparée aux frontières de chaque catégorie, formant un profil B={b1,b2,b3,…,bh}. La Figure 6 illustre la problématique de tri ou d'affectation.

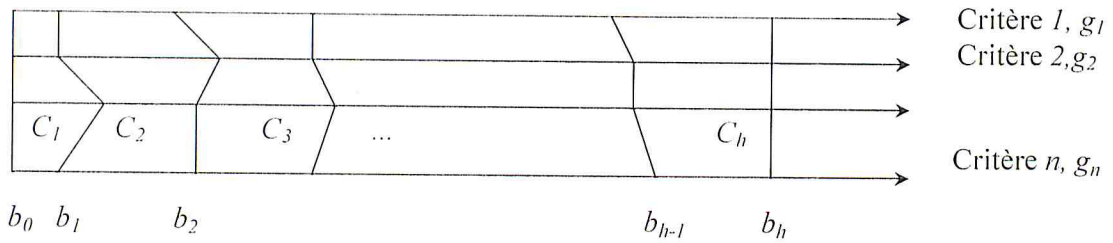


Figure.6 : Illustration de la problématique de tri

L'affectation des actions dans les catégories se base sur le concept de surclassement. Une action a de l'ensemble A surclasse b_h, noté a S b_h, si a est aussi bonne que b_h sur tous les critères et a n'est pas mauvaise que b_h sur la majorité des critères (a_h peut être mauvaise que b_h sur certains critères).

L'approche ELECTRE-TRI s'appuie sur les étapes suivantes :

1. Evaluation des indices de concordance

$$c_j(a,b_h) = \begin{cases} 0 \; si \; g_j(b_h) - g_j(a) \geq p_j(b_h) \\ 1 \; si \; g_j(b_h) - g_j(a) \leq q_j(b_h) \\ \dfrac{p_j(b_h) + g_j(a) - g_j(b_h)}{p_j(b_h) - q_j(b_h)} \end{cases}$$

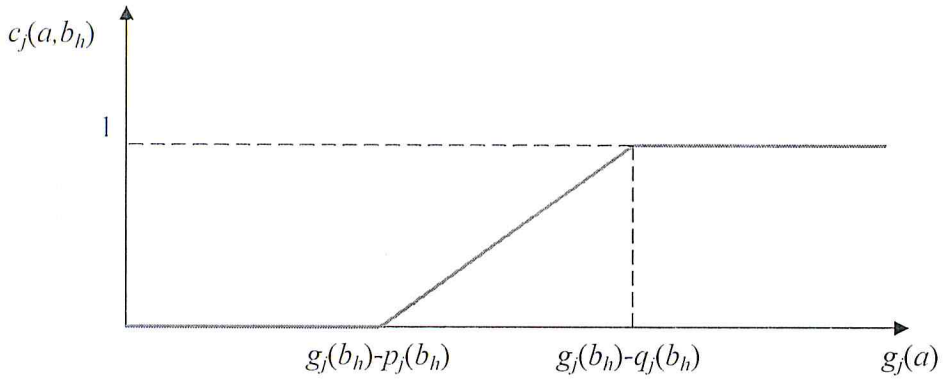


Figure .7 : Détermination de l'indice de concordance

2. Calcul de l'indice de concordance global :

$$C(a_i,b_i)=\frac{\sum_{j\in F} k_j c_j(a,b_k)}{\sum_{j\in F} k_j}$$

3. Calcul de l'indice de discordance :

$$d_j(a,b_i)=\begin{cases} 0\ si\ g_j(a_k)\le g_j(b_k)+p_j(b_k) \\ 1\ si\ g_j(a_k)>g_j(b_k)+v_j(b_k) \\ \in [0.1]\ \text{sinon} \end{cases}$$

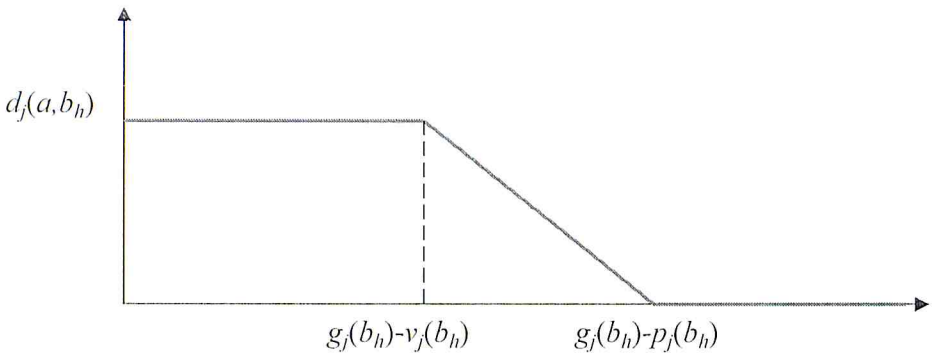


Figure.8 Détermination de l'indice de discordance

4. Calcul de l'indice de crédibilité et définition de la relation de surclassement floue :

$$\sigma(a,b_h) = C(a,b_h)\prod_{j\in\bar{F}}\frac{1-d_j(a,b_h)}{1-C(a,b_h)}$$

Avec $\bar{F} = \{j \in F : d_j(a,b_h) > C(a,b_h)\}$

On définit l'indice de coupe λ comme le paramètre qui détermine la situation de préférence entre a et b_h.

La relation de surclassement définie se base sur l'indice de crédibilité $\sigma(a,b_h)$ et l'indice de coupe λ tel que :

- $\sigma(a,b_h) \geq \lambda$ et $\sigma(b_h,a) \geq \lambda \Rightarrow a\ S\ b_h$ et $b_h\ S\ a \Rightarrow a\ I\ b_h$, a est indifférent à b_h.
- $\sigma(a,b_h) \geq \lambda$ et $\sigma(b_h,a) < \lambda \Rightarrow a\ S\ b_h$ et b_h ne surclasse pas a $\Rightarrow$ *a surclasse* b_h
- $\sigma(a,b_h) < \lambda$ et $\sigma(b_h,a) \geq \lambda \Rightarrow$ *a ne surclasse pas* b_h *et* $b_h\ S\ a \Rightarrow b_h$ *surclasse a.*
- $\sigma(a,b_h) < \lambda$ et $\sigma(b_h,a) < \lambda \Rightarrow$ *a ne surclasse pas* b_h *et* b_h *ne surclasse pas a, dans ce cas a et b sont incomparables.*

Deux procédures d'affectation sont possibles :

1. *Procédure pessimiste*

   a) Comparer successivement a à b_i , tel que $i=p,p-1,\ldots,0$

   b) Si $a\ S\ b_h$, a est assigné à la catégorie C_{h+1}.

2. *Procédure optimiste*

   a) Comparer successivement a à b_i , tel que $i=1,2,\ldots,p$

   b) Si $b_h\ S\ a$, a est assigné à la catégorie C_h.

L'algorithme ELECTRE TRI est représenté par la Figure 9 ci-dessous.

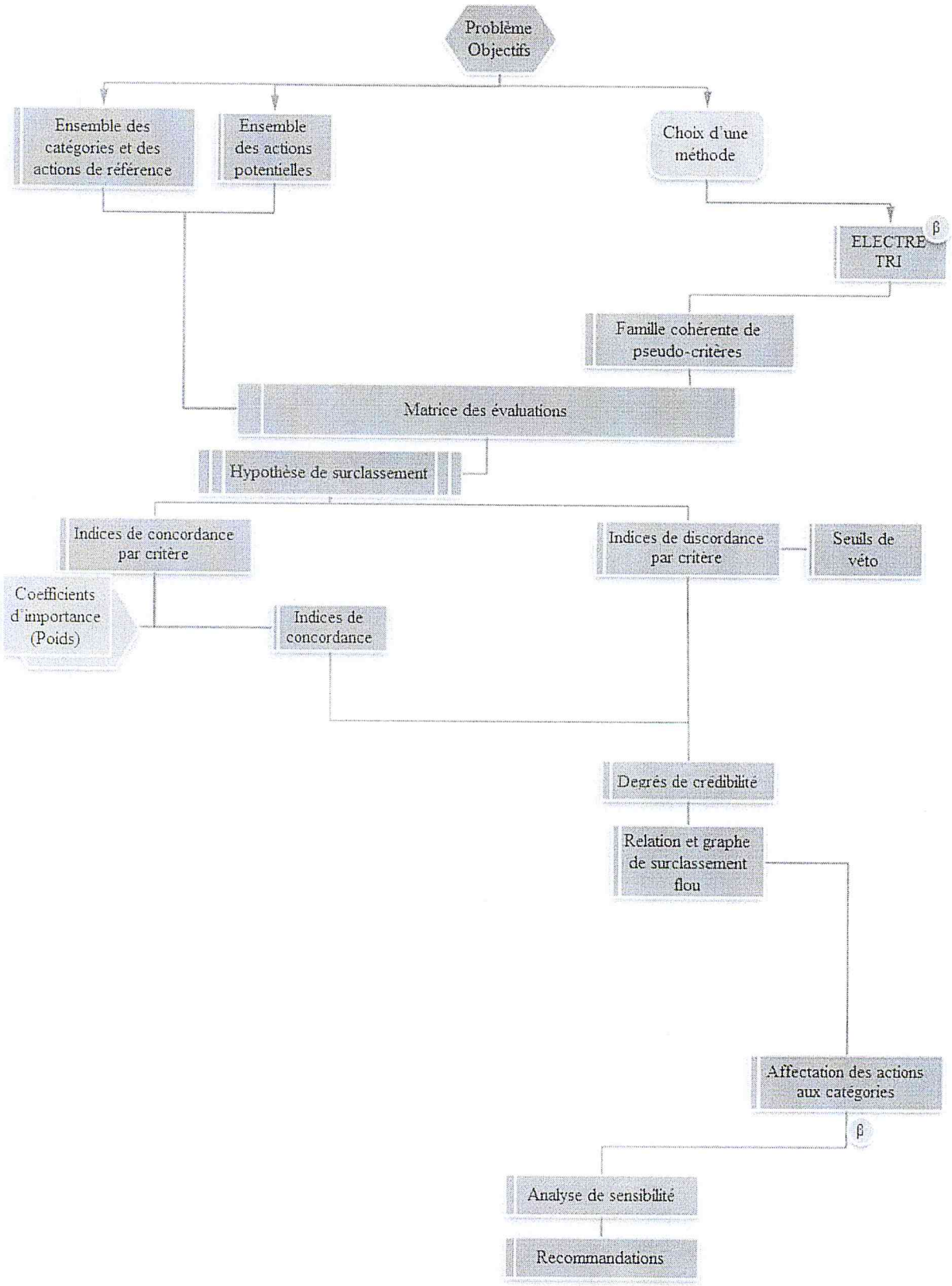


Figure.9 : Un organigramme qui représente la méthode Electre TRI [Maystre L. Yves,et al.,1996]

4.3 ELECTRE II

La méthode ELECTRE II relève de la problématique g (procédure de classement).

Elle vise, en utilisant les relations d'ordre sur chacun des critères, à munir l'ensemble A des actions potentielles d'une structure de préordre total afin de faciliter le choix. En résumé, cette méthode a pour but de classer les actions potentielles, depuis les "meilleures" jusqu'aux "moins bonnes", en tolérant les ex æquo.

Dans un **préordre** (relation réflexive et transitive), des ex æquo sont possibles. Dans un **ordre** (relation réflexive, transitive et antisymétrique), il n'y pas d'ex æquo.

Un **préordre total** est un préordre dans lequel les éléments sont toujours comparables (incomparabilité exclue).
Un **préordre partiel** est un préordre dans lequel l'incomparabilité est permise.

Il faut remarquer qu'en problématique g, il n'est pas tenu compte de la valeur intrinsèque de chaque action mais seulement de sa valeur relative par rapport aux autres actions.

Cette méthode utilise, tout comme la méthode ELECTRE I, la relation de surclassement S. Cependant, la distinction est faite entre deux sortes de surclassements :

- les **surclassements forts** qui reposent sur des bases solides et sont donc avancés avec une grande certitude,
- les **surclassements faibles** qui concernent ceux des surclassements qui sont sujets à caution.

L'exploitation de ces deux graphes (l'un fort, l'autre faible) s'opère selon un algorithme qui permet de classer les actions. Cet algorithme permet d'obtenir deux classements différents (ou deux préordres totaux différents) : un classement direct et un classement inverse.

Les deux classements s'opèrent à partir du graphe de surclassement fort, le graphe de surclassement faible n'étant utilisé que pour départager si possible les ex æquo.

A partir de ces deux préordres totaux, un préordre partiel est établi, ainsi l'intérêt de ces deux classements provient de leur effet sur des actions incomparables.

Le point de départ d'ELECTRE II est tout à fait différent de celui d'ELECTRE I, il ne s'agit plus d'essayer de trouver la "meilleure" action, mais de classer toutes les actions de la "meilleure" jusqu'à la "moins bonne".

Du même coup, les résultats sont plus tranchés que dans la précédente méthode. L'approche utilisée reste toujours la même, elle est fondée sur la concordance et la discordance. Cependant, les **moyens utilisés** pour exprimer ces notions sont enrichis par rapport à ceux d'ELECTRE I et **permettent de tenir compte de la volonté du décideur d'une manière plus fine**.

Une autre grande nouveauté d'ELECTRE II est l'introduction de deux types de surclassements fort et faible, la méthode essaie ainsi de mieux respecter les nuances du réel.

Enfin l'algorithme de classement est un important pas d'innovation. L'existence de deux préordres, établis d'une manière différente, offre la possibilité de se faire une idée de la solidité des résultats selon un angle de vue complémentaire à celui de l'analyse de robustesse : une action qui change énormément de rang entre les deux classements, direct et inverse, est une action qui peut difficilement se comparer aux autres.

L'algorithme ELECTRE II est représenté par la Figure 10 ci-dessous.

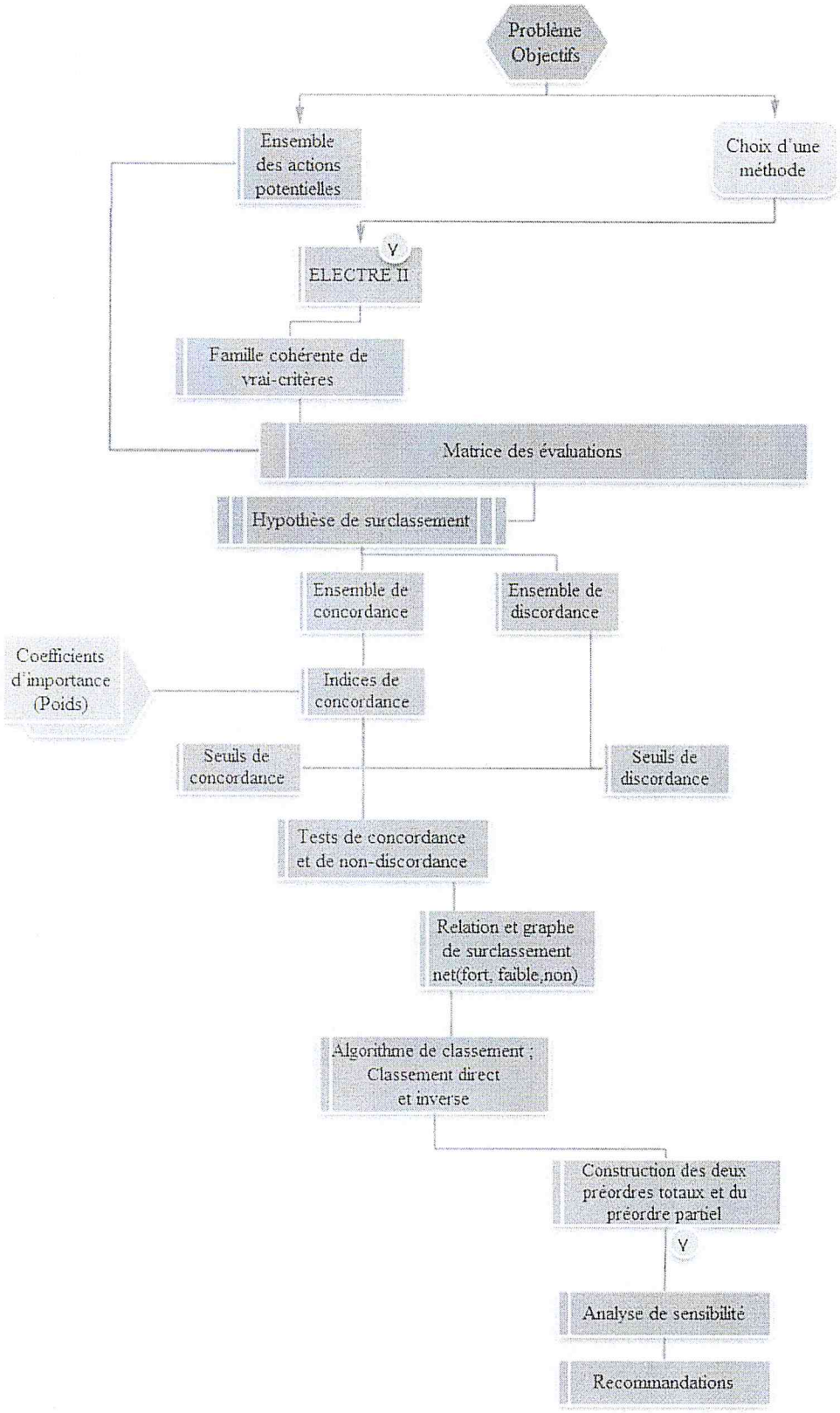


Figure.10 : Un organigramme qui représente la méthode Electre II [Maystre L. Yves,et al.,1996]

4.4 Electre-III

La méthode Electre-III est une méthode d'analyse multicritère qui permet de résoudre des problèmes de classement, la méthode s'appuie sur la définition d'une relation de surclassement ***S*** permettant de comparer deux actions a et b distinctes.

En considérant un ensemble d'actions $A=\{a1, a2, a3,\ldots,am\}$, il s'agit de classer les actions en les comparants par pairs. Chaque action est donc comparée aux autres sur la base des critères considérés. L'évaluation des actions est effectuée par une fonction réelle, pour chaque critère on définit l'ensemble $G=\{g1,g2,\ldots,gn\}$ contenant l'évaluation de l'action sur l'ensemble des critères.

L'importance des critères dans la prise de décision est évaluée par un ensemble de poids $K=\{k1,k2,\ldots,kn\}$. Pour cette méthode, les seuils d'indifférence, de préférence et de veto sont fonction de l'évaluation de l'action pour chaque critère. Pour une action a, évaluée par $gj(a)$ pour le critère j, dans ce cas le seuil d'indifférence est noté $qj(gj(a))$, le seuil de préférence par $pj(gj(a))$ et le seuil de veto par $vj(gj(a))$. La méthode Electre III s'appuie sur les étapes suivantes :

1. Evaluation des indices de concordance :

Dans ce cas on considère le sens de préférence des critères considérés, on distingue un sens de préférence croissant et décroissant.

A titre d'exemple l'indice de concordance dans le cas de préférence croissante est obtenu comme suit :

$$C_j(a,b_h)\begin{cases} 0 \; si \; g_j(b)-g_j(a) \geq p_j(g_j(a)) \\ 1 \; si \; g_j(b)-g_j(a) \leq q_j(g_j(a)) \\ \dfrac{p_j(g_j(a)) \quad g_j(b)-g_j(a)}{p_j(g_j(a))-q_j(g_j(a))} \end{cases}$$

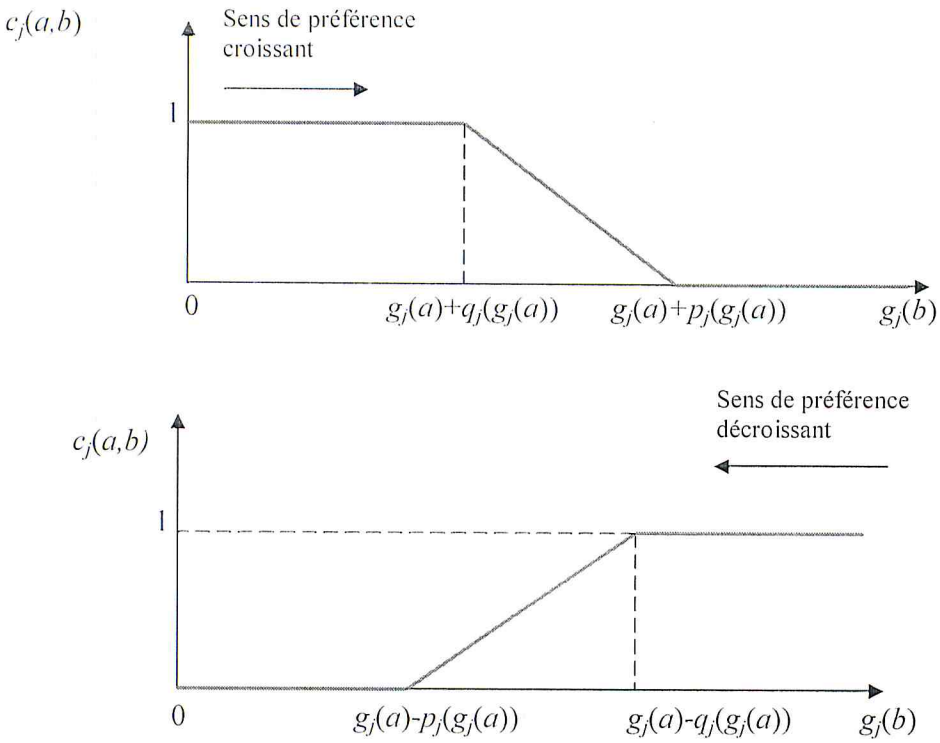


Figure.11 : Détermination de l'indice de concordance

2. Calcul de l'indice de concordance global :

$$C(a.b) = \frac{\sum_{j=F} k_j c_j(a.b)}{\sum_{j=F} k_j}$$

3. Evaluation des indices de discordance :

Dans ce cas on considère le sens de préférence des critères considérés, on distingue un sens de préférence croissant et décroissant.

A titre d'exemple l'indice de discordance dans le cas de préférence croissante est obtenu comme suit :

$$D_j(a,b_h) \begin{cases} 0 \; si \; g_j(b) - g_j(a) \leq p_j(g_j(a)) \\ 1 \; si \; g_j(b) - g_j(a) \leq v_j(g_j(a)) \\ \in [0,1] \; sinon \end{cases}$$

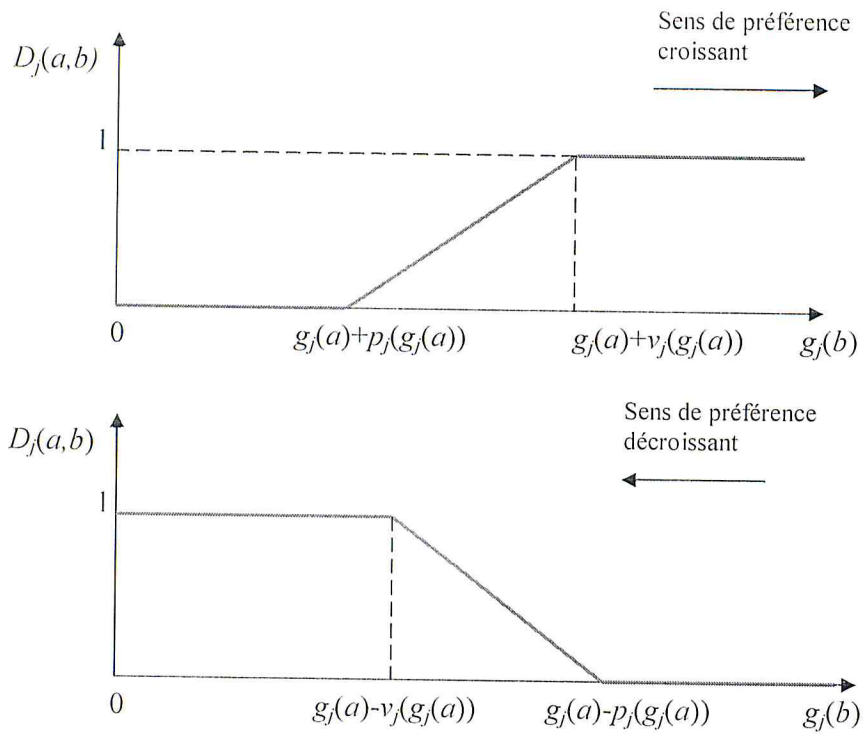


Figure.12 : Détermination de l'indice de discordance

5. Calcul de l'indice de crédibilité et définition de la relation de surclassement floue :

$$d(a,b) = C(a,b)\prod_{j\in\bar{F}} \frac{1 - D_j(a,b)}{1 - C(a,b)}$$

Avec $\bar{F} = \{j \in F : d_j(a,b) > C(a,b)\}$

Cette relation de surclassement est utilisée pour classer les actions à l'aide de préordre. La première relation est obtenue de manière descendante, en sélectionnant la meilleure action et en classant les autres actions de la meilleure à la moins bonne, on parle alors de ***distillation descendante***. La seconde se fait de manière ascendante, en choisissant d'abord la mauvaise action, et en classant de la plus mauvais à la meilleure action, on parle alors de ***distillation ascendante***. La construction des deux préordres se base dans un premier temps sur la définition d'un niveau de coupe $\lambda_1 \in [0,1[$, on préconise $\lambda_1 = \max_{a,b\in A}\{d(a,b)\}$ sélection les actions vérifiant la condition $d(a,b) > \lambda 1$. On obtient ainsi une relation de surclassement nette notée $S_1^{\lambda_1}$ définie par : A $S_1^{\lambda_1}$ b ssi $d(a,b) > \lambda 1$ et $d(a,b) > d(b,a)+s(d(a,b))$ Avec $s(d(a,b)) = -0.5.\ d(a,b)+0.30.$ valeurs préconisées.

L'algorithme ELECTRE III est représenté par la Figure 13 ci-dessous.

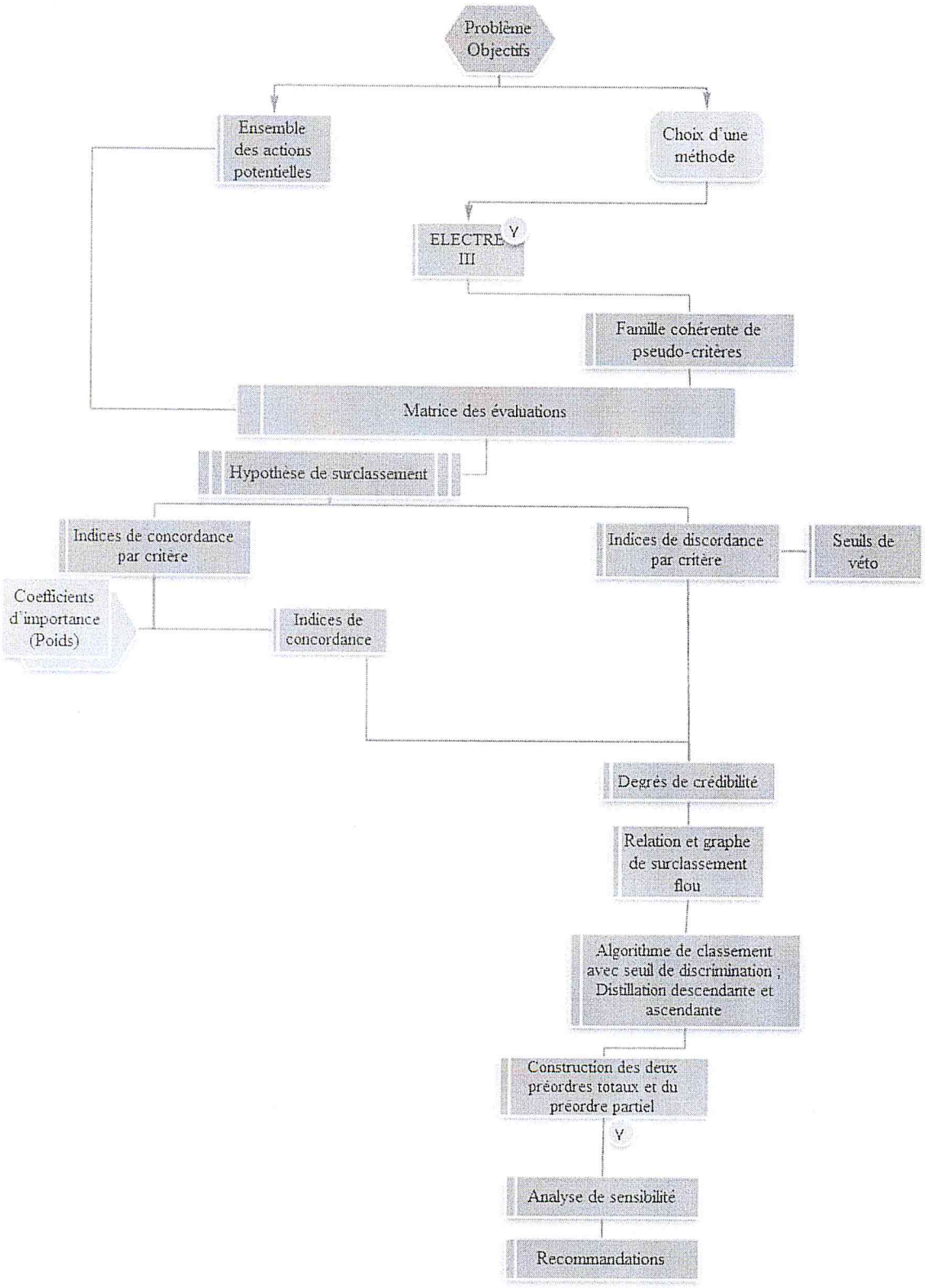


Figure.13 : Un organigramme qui représente la méthode Electre III [Maystre L. Yves,et al.,1996]

4.5 ELECTRE IV

La méthode ELECTRE IV qui relève aussi de la problématique g (procédure de classement) témoigne d'une sophistication de plus en plus poussée.

ELECTRE II et ELECTRE III ont certes inspiré cette méthode mais, néanmoins, la plus grande originalité est qu'il n'y a plus de poids attribué à chaque critère. Ce changement fondamental est accompagné d'une grande nouveauté : l'abandon de l'hypothèse de surclassement, qui rend inutiles les notions de concordance et de discordance.

ELECTRE IV utilise, comme ELECTRE III, des pseudo-critères, c'est-à-dire des critères associés à un seuil de préférence stricte et à un seuil d'indifférence. A partir de la matrice des évaluations, les actions sont comparées deux à deux. Cette comparaison situe, pour chaque critère, l'une des actions par rapport à l'autre selon un cas de figure déterminé. Le nombre de fois que chaque cas de figure particulier apparaît pour l'ensemble des critères est enregistré.

Des règles simples, utilisant ces chiffres, permettent d'établir des relations de surclassement entre deux actions. L'établissement de ces règles se fait de telle manière qu'aucun des critères ne soit par trop "prépondérant" ou par trop "négligeable". Cette notion, facile à comprendre mais un peu floue a depuis été précisée sous l'appellation d'hypothèse de disparité limitée.

La méthode admet plusieurs versions des types de surclassement :

- quatre niveaux dans la crédibilité du surclassement (4 types de relations)
- deux niveaux dans la crédibilité du surclassement (surclassements fort et faible)
- certaines des combinaisons intermédiaires

A chaque type de relation de surclassement correspond un degré de crédibilité attribué d'une manière plus ou moins volontariste. Ceci conduit à la construction d'une matrice des degrés de crédibilité contenant un nombre discret de valeurs possibles. Alors, à à partir de ce moment-là, la méthode ELECTRE IV reprend le même principe utilisé par la méthode ELECTRE III avec une distillation ascendante et une distillation descendante et enfin un classement final qui est aussi un préordre partiel.

L'algorithme ELECTRE IV est représenté par la Figure 14 ci-dessous.

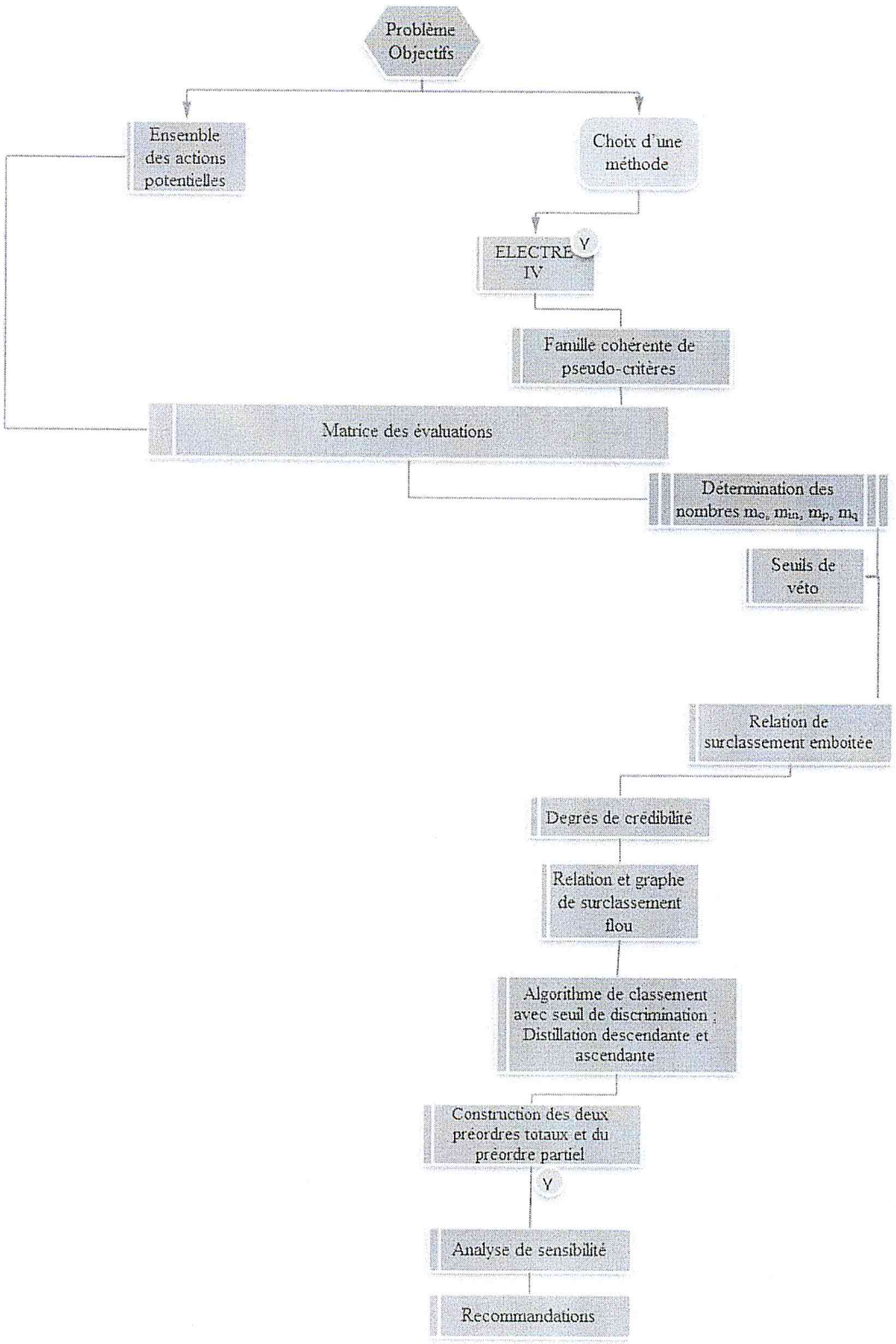


Figure.14 : Un organigramme qui représente la méthode Electre IV [Maystre L. Yves,et al.,1996]

4.6 ELECTRE IS

La méthode ELECTRE IS qui relève aussi de la problématique a est une adaptation de ELECTRE I à la logique floue, permettant d'utiliser des pseudo-critères (le S veut dire seuil).

Pour choisi la "meilleure" action potentielle, une partition des actions potentielles A en deux sous-ensembles doit être réalisée, comme ELECTRE I, c'est dans le noyau (sous-ensemble des actions non-surclassés) que se trouve la "meilleure" action.

Trouver une solution au problème que posent les circuits est difficile pour toutes les problématiques. Cette difficulté est encore renforcée dans cette problématique du fait du caractère "brutal" de l'attribution d'une action à l'un ou l'autre des deux sous-ensembles, alors que les autres problématiques permettent un traitement plus différencié, soit en termes de préordres (problématique g) soit en termes d'affectation à une catégorie (problématique b).

ELECTRE IS permet de mieux cerner ce problème et offre des outils d'analyse permettant de connaître, pour chaque circuit maximal, son taux de cohésion interne (relations entre les actions le composant) et son taux de liaison externe (relations avec les autres éléments du graphe, action ou circuit).

ELECTRE IS constitue une amélioration par rapport à ELECTRE I, dans la mesure où elle permet l'utilisation de pseudo-critères et donc de la logique floue.

Dans sa version originale, cet intérêt est partiellement perdu par le fait que l'on a toujours recours à deux tests disjoints, un pour la concordance, l'autre pour la non-discordance.

La seconde version, qui utilise des degrés de crédibilité, est à ce titre plus intéressant.

Cependant, la transformation de la relation de surclassement floue en une relation nette enlève de l'intérêt à cette version. Contrairement à l'exploitation minutieuse et précautionneuse des degrés de crédibilité dans ELECTRE III et IV, elle nivelle beaucoup de nuances. Les indicateurs de stabilité des circuits, qui constituent une aide appréciable pour mieux comprendre les relations entre les actions, tant à l'intérieur des circuits qu'entre les circuits et les autres éléments du graphe sont forts utiles. Ces indicateurs permettent notamment de connaître la stabilité des circuits et donc d'anticiper l'effet éventuel d'une modification des relations entre les actions.

L'algorithme ELECTRE IS est représenté par la Figure 15 ci-dessous.

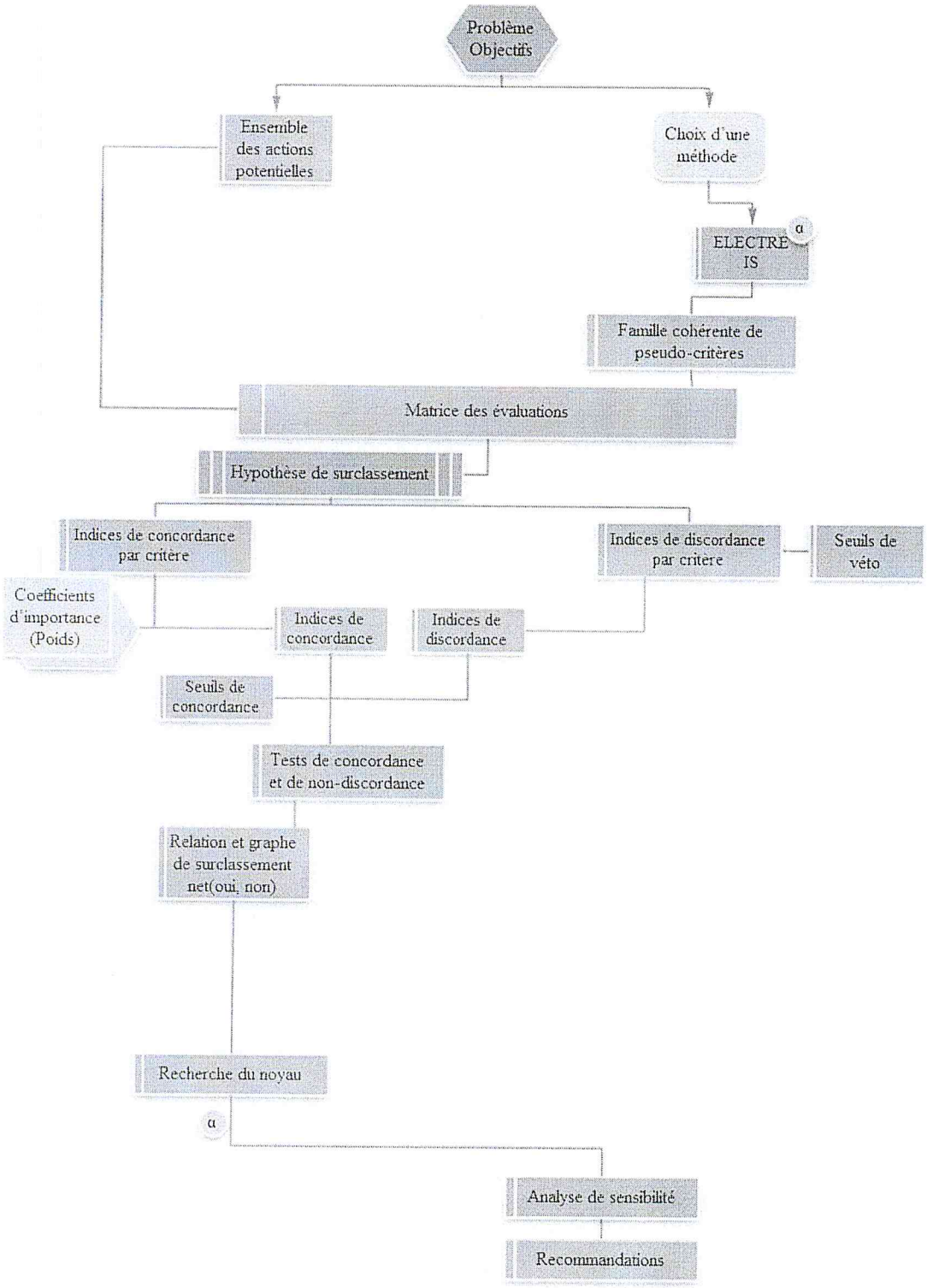


Figure.15 : Un organigramme qui représente la méthode Electre IS [Maystre L. Yves,et al.,1996]

5. Procédure de choix d'une méthode Electre

Au moment de choisir une méthode, il convient de se rappeler les deux points suivants :

- il faut hiérarchiser les choix, en commençant par la problématique et le type de critère, puis la méthode, enfin la version,
- il n'est pas nécessaire de choisir une méthode parmi les méthodes ELECTRE dès le début de l'étude. Ceci n'est nécessaire que lorsque la matrice des évaluations est remplie.

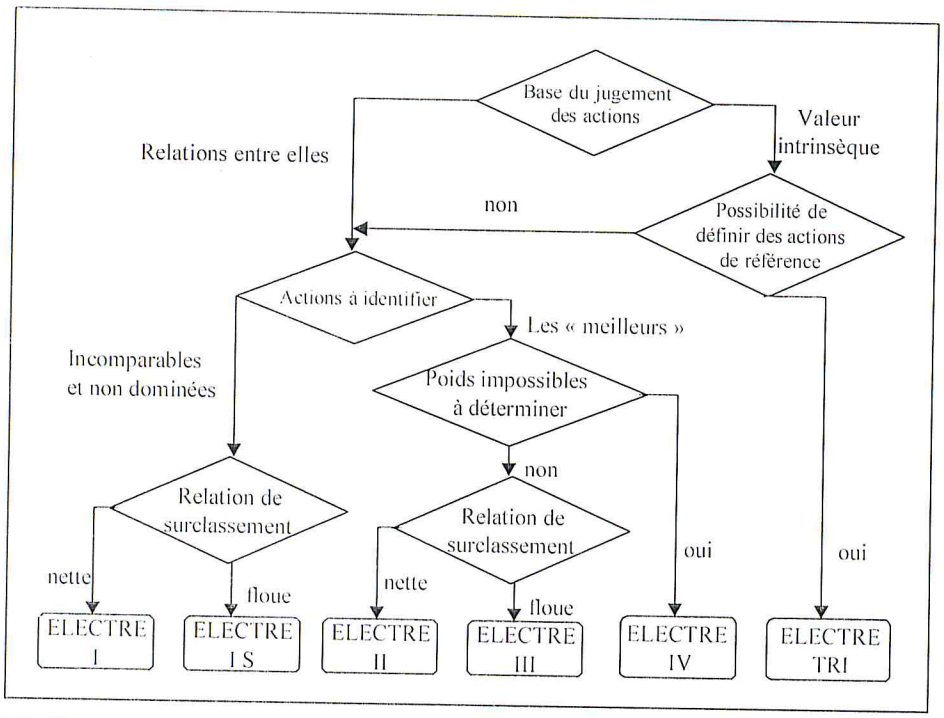


Figure.16 : Procédure de choix d'une méthode ELECTRE [Maystre L. Yves,et al.,1996]

6. Déroulement du processus d'aide a la décision

Le processus d'aide à la décision comprend un certain nombre d'étapes, comme le montre l'exemple détaillé de la figure suivante.

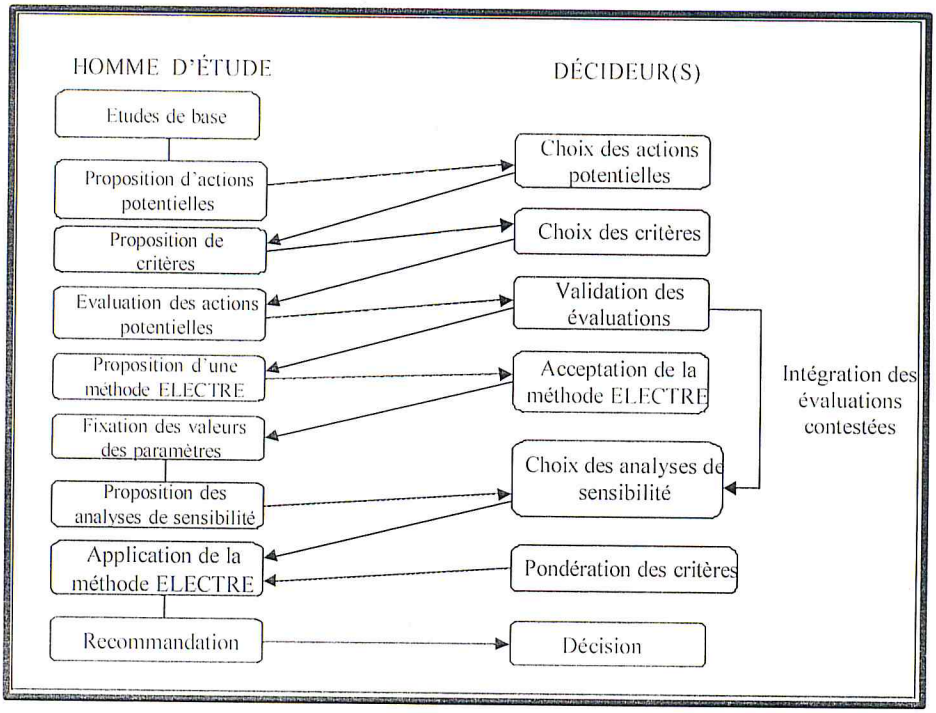


Figure.17 Déroulement du processus d'aide à la décision **[Maystre L. Yves,et al.,1996]**

Cette figure montre bien la nécessité de dialogue entre l'homme d'étude et le décideur. Lorsque le nombre de décideurs est important et leurs positions respectives opposées, le processus d'aide à la décision se transforme en processus de négociation.

Les propositions de l'homme d'étude doivent être connues pour faire réagir le décideur. Habituellement, la discussion sur les actions potentielles précède celle sur les critères, car c'est au moment où les différences entre les actions sont claires qu'il est possible de définir les critères pertinents pour différencier ces actions.

7. Conclusion

Les résultats obtenus avec les méthodes de type Electre ne donnent pas de renseignement sur l'écart qui existe entre deux actions lorsque l'une est préférée à l'autre hormis pour la procédure de tri où le nombre de catégories séparant deux actions affectées différemment peut être significatif.

Cependant, elles permettent de rendre compte de certaines situations apparemment ambiguës (incomparabilité, préférences et indifférences intransitives) mais qui sont la traduction de cas auxquels peuvent être confrontés les décideurs.

L'un des avantages de la famille des méthodes de type Electre est le caractère non totalement compensatoire de l'agrégation multicritère. En effet, à aucun moment l'on a essayé de réaliser une transformation d'échelle permettant de ramener à un critère unique de synthèse, une famille de critères prenant des valeurs de nature a priori hétérogènes. Les poids attribués à chaque critère sont complètement indépendants de l'échelle du critère en question.

La présence de seuil de veto est un moyen supplémentaire de maîtriser le danger de la compensation en sanctionnant des écarts trop importants sur des critères dont les aspects peuvent être plus sensibles.

Chapitre III

Le modèle MDA

I. Introduction

Dans ce chapitre, nous abordons le contexte lié à la conception de la solution que nous proposons afin d'assurer l'interopérabilité des entreprises : le Système d'Information Collaboratif. Le choix d'une approche d'ingénierie appropriée dans une démarche de conception d'un système d'information est évidemment prépondérant. La nature et la complexité de la solution (faire interopérer des systèmes d'information complexes, hétérogènes et répartis) nous obligent à suivre une démarche rigoureuse et structurée, prenant en compte la spécificité du contexte de développement.

Le choix de l'approche de développement basée sur les modèles MDA [**OMG 03**] s'est imposé rapidement : elle facilite la descente en abstraction et permet une séparation nette et délimitée entre le métier et la technique. Dans cette partie de manuscrit, nous discutons le choix de cette approche et ses avantages. Nous définissons ses principes de construction d'applications réparties et ses apports quant à l'interopérabilité des systèmes d'information. Nous montrons aussi les mécanismes de transformation de modèles qui relèvent de l'Ingénierie Dirigée par les Modèles (IDM). En effet, l'intérêt pour l'IDM a été fortement amplifié, lorsque l'OMG a rendu publique son initiative MDA (qui vise à la définition d'un cadre normatif pour l'IDM). Il existe cependant bien d'autres alternatives technologiques aux normes de l'OMG, par exemple Ecore dans la sphère Eclipse. L'IDM dépasse largement MDA, pour se positionner plutôt à la confluence de différentes disciplines.

II. Pourquoi une approche de conception basée sur MDA ?

La conception des applications informatiques, de plus en plus complexes et consommatrices de ressources, a subi une grande mutation marquée par l'apparition de nombreux paradigmes différents. Le paradigme procédural a ainsi laissé la place au paradigme objet, puis au paradigme composant.

Cependant, ces évolutions, bien que majeures, ne permettent plus de faire face à la complexité croissante des systèmes logiciels développés. Les logiciels ne sont pas seulement complexes parce qu'ils doivent manipuler une grande quantité de données, mais aussi parce qu'ils doivent répondre parfaitement aux besoins spécifiques de leurs utilisateurs. De plus, dans un contexte de collaboration, cette complexité s'accroît du fait de l'hétérogénéité des partenaires. En conséquence, le développement de logiciels informatiques ne peut plus

être fait en utilisant une démarche intuitive. D'après **[Lopes 05]**, « *Il devient largement recommandé d'utiliser des méthodologies différentes pour concevoir, analyser, modéliser et produire ces logiciels* ». Le génie logiciel a donc connu sa première vague de structuration de la conception d'applications. Plusieurs méthodologies ont été proposées pour supporter le développement logiciel dans les années 1980, telles que MERISE, OOD (*Object Oriented Design*), OMT (*Object Modeling Technique*) et OOSE (*Object Oriented Software Enginnering*) [**Morley** *et al.* **02**]. En 1997, ces méthodologies ont convergé vers le formalisme UML. Ce langage pseudo-formel est devenu une référence pour le développement orienté-objet d'un système. Cependant, son usage s'est principalement orienté vers la documentation des logiciels, délaissant quelque peu le domaine du « développement ».

L'approche MDA propose alors d'aller encore plus loin avec les modèles de type UML par la création et le maintien des logiciels à partir de ces modèles, avec moins d'efforts que dans le passé. Cependant, l'approche MDA n'est pas une rupture avec l'existant. **[Lopes 05]** affirme que MDA n'est pas une apparition d'idées ou de technologies nouvelles, mais la normalisation de fait, d'un certain nombre de notations consensuelles et de démarches caractérisant les bonnes pratiques du développement du logiciel en technologie des objets et des composants. En effet, MDA met en avant la mise en relation entre un modèle « *exploitable par l'homme* », qui décrit une certaine connaissance métier en capitalisant les besoins spécifiques, en lien avec le logiciel à développer et un modèle « *exploitable par l'informatique* », qui définit la spécification technique de logiciels.

Dans **[Garcia 04]**, on énumère trois raisons pour le choix d'un développement de logiciel avec une approche MDA :

« *tout d'abord pour séparer logique métier et technologies informatiques. Les éléments du métier sont stables et capitalisent le savoir-faire de l'entreprise. Il faut donc* ***modéliser les processus métiers*** *pour les rendre indépendants des technologies et aussi faciliter leur adaptation aux nouvelles technologies.*

Ensuite parce que la technologie des objets et des composants n'a pas tenu ses promesses en termes de simplification de conception. La complexité de ces technologies est croissante et fait régulièrement appel à des techniques (« design patterns », «frameworks »,etc.) pour y remédier.

Enfin, il devient urgent de stopper l'empilement des technologies. Des systèmes hétéroclites, qu'il devient difficile de faire communiquer et qui nécessitent des compétences nombreuses et variées pour maintenir et faire vivre le système d'information ».

En ce qui concerne notre problématique de conception et en se basant sur les raisons de **[Garcia 04]** citées ci-dessus, nous souhaitons évaluer les apports de l'approche MDA pour mener efficacement la démarche de définition du notre système.

III. Présentation de l'approche MDA

MDA promeut une nouvelle voie pour le développement des applications, basée sur l'élaboration d'un modèle indépendant des plateformes **(PIM - *Plateform Independent Model*)** à partir d'un modèle métier **(CIM - *Computation Independent Model*)** qui décrit le besoin métier de l'application à développer, et sur la transformation du PIM en un ou plusieurs modèles dépendants de plateformes **(PSM - *Plateform Specific Model*)**, correspondant à chaque plateforme sur laquelle l'application va être déployée. D'après **[Bézivin** *et al.* **03a]** MDA fournit un processus de conception et met en œuvre des outils pour : spécifier un système indépendamment de la plate-forme qui le supporte, et donc réaliser un PIM, enrichir ce modèle par étapes successives, spécifier les plateformes, choisir une plateforme particulière pour le système, transformer la spécification du système (PIM) en une autre spécification pour une plateforme particulière (PSM), transformer un CIM en un PIM et un PIM en un autre PIM, raffiner le PSM jusqu'à obtenir une implémentation exécutable.

IV. Les types de modèle dans MDA

Dans le processus MDA, tout est considéré comme modèle, aussi bien les spécifications des applications que le code source ou le code binaire. Les quatre types de modèles identifiés sont donc les CIMs, les PIMs, les PDMs (ou PMs) et les PSMs **[Miller** *et al.* **03]** **[Bézivin** *et al.* **02]** :

CIM (*Computation Independent Model*) : appelé aussi modèle de domaine ou modèle métier. Il montre le système dans l'environnement organisationnel dans lequel il va être exécuté. Son but est d'aider à la compréhension du problème. Les exigences exprimées dans le CIM doivent être traçables dans le PIM et le PSM. Comme exemple de CIM : un modèle de processus **[Miller** *et al.* **03]** **[Bézivin** *et al.* **02]**.

PIM (***Platform Independent Model***) : est des modèles qui n'ont pas de dépendance avec les plateformes techniques. Les PIMs représentent par exemple les différentes entités fonctionnelles d'un système avec leurs interactions, exprimées uniquement en termes de la logique d'entreprise. Ils représentent l'intérêt de l'application : le logique métier. Ces modèles sont mis au point par un architecte spécialisé dans le domaine de l'application. Comme exemple de PIM : un modèle d'architecture logique [**Miller** *et al.* **03**] [**Bézivin** *et al.* **02**].

PDM (***Platform Description Model***) ou **PM** (***Platform Model***) : il décrit la plate- forme sur laquelle le système va être exécuté (modèles de composants à différents niveaux d'abstraction : CCM, C#, EJB, EDOC, etc). Actuellement, il est souvent sous forme de manuels de logiciels et de matériels. Dans une démarche MDA, on se base sur les PDMs pour générer les PSMs à partir des PIMs [**Miller** *et al.* **03**] [**Bézivin** *et al.* **02**].

PSM (***Platform Specific Model***) : les PSM quant à eux sont dépendants de plate- formes techniques. Les PSM servent essentiellement de base à la génération de code exécutable vers ces mêmes plateformes techniques. Comme exemple de PSM : un modèle de Services-web [**Miller** *et al.* **03**] [**Bézivin** *et al.* **02**].

V. La transformation de modèles en MDA

La transformation de modèles est le processus qui transforme un modèle en un autre modèle du même système, mais à un niveau différent. Certaines étapes du processus MDA correspondent à la transformation d'un modèle en un autre modèle (de même type ou non), en utilisant éventuellement d'autres modèles. Plus précisément, un PIM suffisamment détaillé est projeté, par l'intermédiaire d'un PDM, vers un PSM. Quatre types de transformations différentes sont ainsi identifiés [**Miller** *et al.* **03**] (Figure 18) :

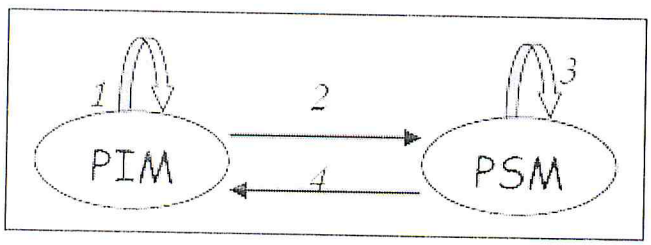


Figure.18 : Les transformations de modèles dans le processus MDA [Miller et al. 03]

V.1. Les transformations en MDA

1. *Transformation de PIM vers PIM* : ces transformations visent à enrichir, filtrer ou spécialiser le modèle sans utiliser d'informations dépendantes d'une plateforme. Les transformations PIM vers PIM sont généralement utilisées pour le raffinement de modèle et ne sont pas toujours automatisables.

2. *Transformation PIM vers PSM* : ces transformations sont effectuées lorsque les PIMs sont suffisamment raffinés pour pouvoir être projetés vers une plateforme d'exécution. Les caractéristiques de cette plateforme peuvent être décrites à l'aide d'un profil UML. L'opération qui consiste à ajouter des informations propres à une plateforme technique pour permettre la génération de code est une transformation PIM vers PSM. A l'heure actuelle, les plateformes techniques visées sont : .Net, J2EE, XML et CORBA. Il apparaît clairement que ce sont les règles qui permettent ces transformations qui sont importantes et qui doivent être généralisées et capitalisées. Ces transformations peuvent donc être fortement automatisées.

3. *Transformation PSM vers PSM* : ces transformations s'appliquent sur un modèle spécifique et génèrent un autre modèle spécifique à la même plateforme. En fait, une unique transformation PIM vers PSM n'est pas toujours suffisante pour permettre la génération de code, il faudra alors parfois transformer les PSM en PSM en utilisant des formalismes intermédiaires. Ces transformations sont donc effectuées pour la réalisation et le déploiement de composants.

4. *Transformation PSM vers PIM* : ces transformations permettent d'obtenir un PIM à partir d'une implantation existante sur une plateforme spécifique. Bien que celles-ci ne fassent pas directement partie du processus MDA, au sens où les spécifications OMG de ce processus ne les montrent pas explicitement, elles sont nécessaires à considérer dans toute stratégie de migration. Actuellement, ces transformations sont les plus difficiles à établir et à automatiser. Idéalement, le résultat de cette transformation devrait correspondre à la transformation inverse PIM vers PSM.

Nous pouvons aussi ajouter *la transformation de CIM vers PIM*. Cette transformation peut nécessiter **un enrichissement de modèle** CIM pour obtenir le modèle de PIM. En effet, nous pouvons dire que la distance entre CIM (exigences métiers) et PIM (exigences logiques de système) est plus grande que la distance entre PIM et PSM (exigences

logiques et techniques d'un même système), ce qui rend cette transformation plus compliquée à définir.

V.2. Notions de modèle et métamodèle

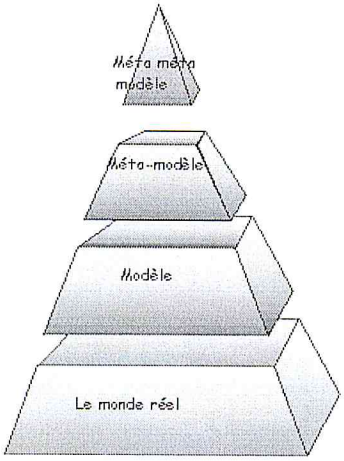


Figure.19 : Modèle et métamodèle [Terasse et al. 03]

D'après [**Terasse** *et al.* **03**], « *le modèle d'un système est la spécification formelle des fonctions, de la structure et / ou du comportement de ce système dans son environnement, dans un certain but* ». Un modèle est souvent représenté par des schémas et du texte. Le texte peut être exprimé dans un langage de modélisation ou en langage naturel.

Un métamodèle d'un modèle est aussi un modèle qui décrit le modèle (Figure 19). Il existe une relation bien précise entre les concepts de modèles et de métamodèles. Chaque élément du modèle est une instance d'un élément de son métamodèle. La relation d'instanciation implique que toutes les informations contenues dans une instance soient décrites dans l'élément correspondant du métamodèle. Réciproquement, toutes les caractéristiques décrites dans l'élément d'un métamodèle peuvent être instanciées par les éléments correspondants du modèle. Nous pouvons dire que :

Un métamodèle correspond à la description d'un modèle, un modèle correspond à l'instanciation d'un métamodèle.

V.3. Mise en œuvre d'une transformation

Nous distinguons trois étapes dans le processus de mise en œuvre d'une transformation MDA [**Bézivin** *et al.* **03**], [**Benguria** *et al.* **06**] :

1. la définition des règles de transformation,

2. le choix d'un outil de transformation,

3. l'exécution des règles de transformation.

VI. Méthodologie de développement MDA

Une méthodologie de développement MDA est « *un processus de développement logiciel basé sur la transformation de modèles et utilisant l'architecture MDA* » [**Bézivin** *et al.* **03**] [**Bézivin** *et al.* **03a**]. Ce processus propose une structuration entre PIM et PSM en application du paradigme de séparation des préoccupations dissociant le logique métier et le code technique.

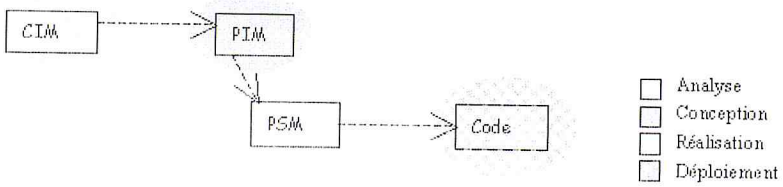


Figure.20: De l'analyse au déploiement en MDA **[Bézivin et al. 03]**

Cette méthodologie est basée sur quatre étapes :

L'**analyse** : elle exprime la logique métier à travers les CIM.

La **conception** : elle révèle les caractéristiques fonctionnelles du système à l'aide des PIMs.

La **réalisation** : elle décrit l'architecture technique et fonctionnelle du système par des PSMs.

Le **déploiement** : il concerne la génération des codes exécutables et de certains artefacts (composants logiciels, documentation, etc.).

VII.Conclusion

Pour que le MDA se diffuse auprès des développeurs, le savoir doit être essaimé et de nouveaux outils doivent être développés. Pour produire ces outils qui font encore aujourd'hui défaut, plusieurs projets ont été lancés pour que l'approche MDA tienne ses promesses. Ces outils permettront l'automatisation des transformations ainsi que la génération automatique de code à partir de modèles, permettant aux architectes de se consacrer pleinement aux tâches de modélisation métier. La démarche MDA deviendra une

architecture viable, dès lors que ces transformations seront concrétisées et intégrées dans les outils.

Le prochain chapitre montre comment nous avons utilisé cette méthode pour concevoir notre système.

Chapitre IV

La démarche de développement

I. Introduction

Notre étude se porte sur l'architecture SOA et plus spécifiquement sur l'intégration de l'aspect aide à la décision dans cette architecture. Dans notre projet on a pris un système qui gère des appels d'offre comme un échantillon des systèmes d'aide à la décision dans une architecture orientée service. Le but de ce chapitre est d'expliquer la conception de l'architecture selon MDA.

II. Méthodologie de développement

Pour la conception de notre solution nous avons fait recourt à la méthodologie 2TUP appelé aussi cycle de développement Y lié au concept MDA que nous avons déjà proposé au chapitre3.

L'utilisation de MDA dans ce contexte permet de mettre en parallèle l'élaboration du PIM et la description de la plate-forme, une jonction entre ces deux constitue le PSM **[Azaiez.S, 92]**.

III. Présentation de la méthodologie 2TUP (Two Track Unified Process)

2TUP (2 Track Unified Process) [Roques et Vallée ,2000], est un processus de développement basé sur le processus UP (Unified Process). Comme tout processus UP, 2TUP se base sur les formalismes de représentation UML. Ce langage s'appuie sur deux aspects complémentaires : la modélisation statique et la modélisation dynamique. La modélisation statique s'appuie sur le diagramme de cas d'utilisation, le diagramme de classes, le diagramme d'objets, le diagramme de composants, et le diagramme de déploiement. La modélisation dynamique s'appuie sur le diagramme d'états, le diagramme d'activité, le diagramme de collaboration et le diagramme de séquence.

Le cycle de développement en Y que nous avons utilisé dans notre projet est celui proposé par S. AZAIEZ, **[Azaiez.S, 92]**, qui s'appui sur l'approche MDA. On y trouve en effet un modèle MDA dans chaque phase de développement jusqu'à la phase de conception. La figure qui suit décrit ce processus de développement.

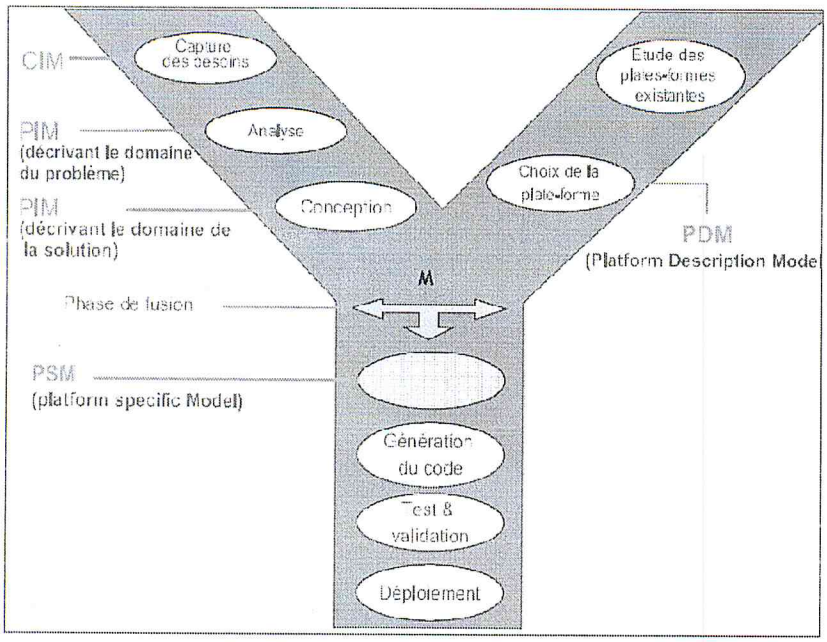


Figure. 21 : Le processus de développement en Y [Azaiez.S, 92]

Selon le processus proposé par MDA la conception et l'analyse sont faites en parallèle avec le choix des plates-formes existantes, durant l'analyse et la conception le PIM et le CIM sont produits comme l'indique le processus fourni par MDA: un premier PIM est élaboré pour décrire le domaine du problème, il est ensuite raffiné pour produire un PIM qui décrit le domaine de la solution. Parallèlement, les aspects techniques de chaque plate-forme sont étudiés et la plate-forme d'implémentation est choisie, les caractéristiques techniques de celle-ci sont modélisés dans le PDM (Platform Description Model). Celui-ci va décrire par exemple, les types supportés par la plate-forme.

Le mapping est définit par l'OMG comme « un ensemble de règles et techniques utilisées pour modifier un modèle afin d'obtenir un autre modèle. Les mappings sont utilisés pour la transformation de PIM vers PIM, de PIM vers PSM, de PSM vers PSM et de PSM vers PIM ».

IV. CIM

Ce modèle exprime les besoins de l'utilisateur, dans cette phase nous allons décrire des cas d'utilisation de notre système puis la modélisation du processus métier avec la notation BPMN.

1) Capture des besoins

Notre besoin débute par la détermination des acteurs et leurs poids (priorité de l'acteur), les cas d'utilisation (fonctions principales du système) puis l'élaboration du diagramme des cas d'utilisation.

a. Acteurs

Maitre d'ouvrage : c'est un agent dans l'entreprise, responsable de lancement des appels d'offre.

Fournisseur : il répond aux appels d'offre

Expert ou Homme d'étude : c'est un expert dans le domaine d'aide à la décision, il a déjà des notions dans le domaine.

Décideur principal : c'est le premier responsable de la décision.

Les décideurs secondaires : c'est un deuxième qui aide le décideur principale.

Administrateur : il supervise le système

b. Cas d'utilisation

Le système doit permettre de réaliser une analyse multicritère pour le choix d'une meilleure soumission. Cette analyse peut suivre une problématique de tri, d'affectation ou de rangement. Le résultat de cette étape sera par la suite négocié entre les différents acteurs concernés par le projet pour aboutir à une solution de consensus ou de compromis.

Donc les deux grandes fonctions du système sont la gestion électronique des appels d'offre, la réponse aux appels d'offre et la négociation assistée entre acteurs.

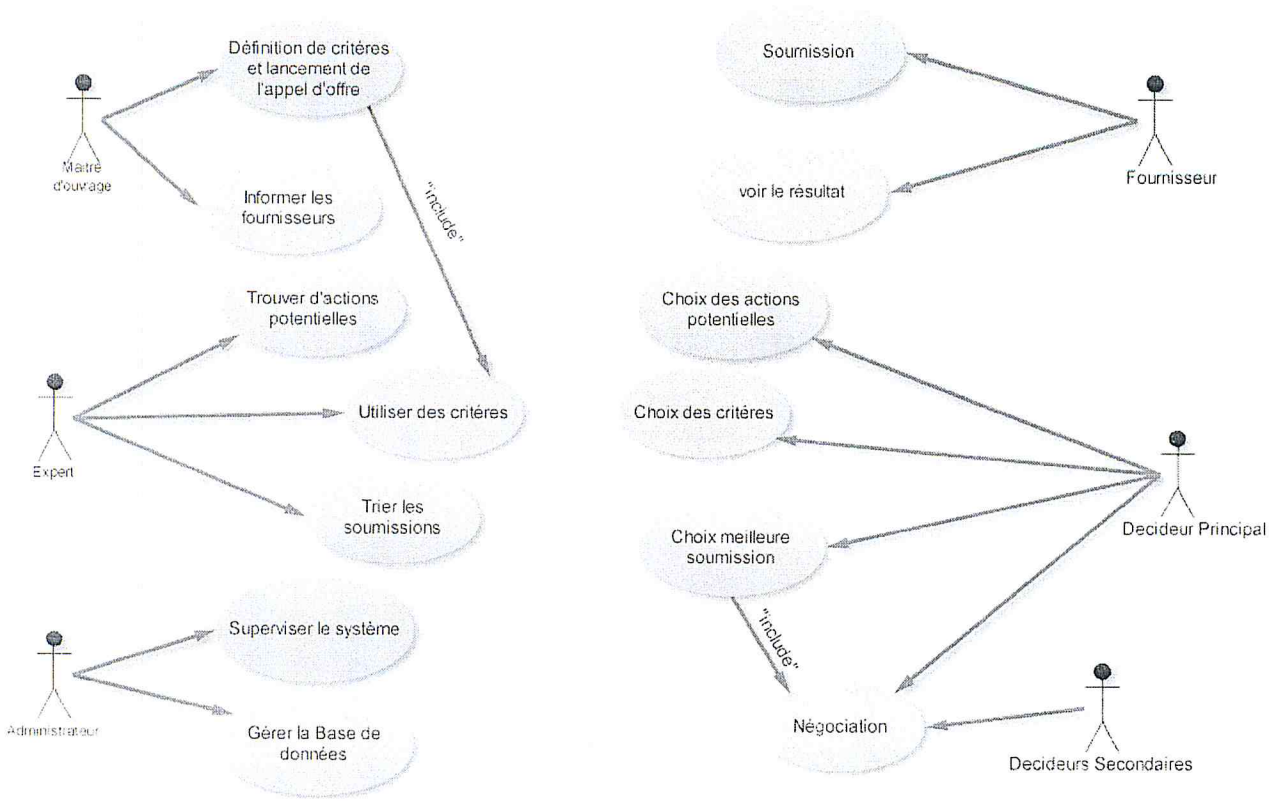


Figure. 22: Diagramme des cas d'utilisation du système

c. Description du processus métier RAO

Un appel d'offre est une procédure dans laquelle un commanditaire (ou maitre d'ouvrage) expose sa demande afin de recevoir des réponses sous forme d'offres. Cette demande peut être matérielle, logistique ou encore humaine. Une fois les offres reçues le commanditaire peut alors choisir parmi celle-ci.

Le décideur principal demande à l'homme d'étude de faire un trie de soumission pour un certain nombre de l'appel d'offre.

L'homme d'étude définit l'ensemble des actions potentielles, les critères (enjeux) et l'évaluation selon les préférences des acteurs potentiels. Le résultat de cette analyse est un ensemble de solutions ordonnées.

Cette solution dépend essentiellement de la problématique et de la méthode multicritère choisie. Il est fort possible que cette solution génère des conflits entre les acteurs. Au lieu de procéder à une phase d'analyse de sensibilité pour essayer de minimiser les conflits, une solution plus efficace est de faire interagir les acteurs entre eux de façon directe, ce qui nécessite une phase de négociation entre les acteurs.

Le décideur principal demande alors à l'ensemble des acteurs (touchés ou intéressés par le projet) d'entrer dans une phase de négociation pour aboutir après un certain nombre d'interactions à une solution de compromis.

Il s'agit de la négociation de la meilleure soumission. L'ensemble des décideurs secondaires entrent en négociation en essayant d'arriver à un compromis. La durée maximale de la négociation est fixée par le décideur principal.

On notera aussi qu'il existe un administrateur ou superviseur du système. Cet administrateur est chargé entre autres d'initialiser le système, de valider et inscrire les acteurs du système, de suivre les différentes phases de la négociation et d'introduire différentes données nécessaires au bon déroulement du processus.

2) La modélisation du processus métier

a) La notation utilisée BPMN (Business Process Modeling Notation)

La notation BPMN est un standard de modélisation des processus métier. Élaborée en 2001, elle a été publiée une première fois en 2004 **[Patrice Briol,08]**. BPMN a pour objectif de proposer un moyen simple et visuel de communication entre les différents intervenants chargés de mettre en œuvre une approche de gestion des processus métiers dans l'organisation.

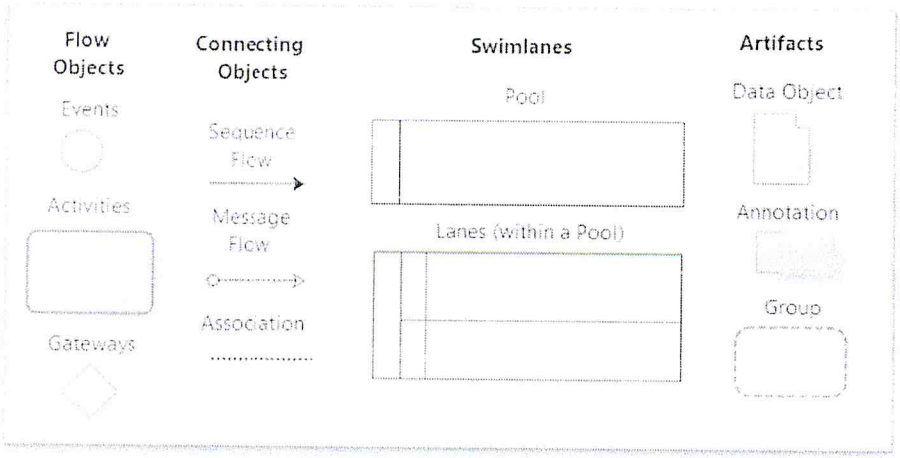


Figure.23: Les éléments graphiques de BPMN.

La notation BPMN couvre uniquement la description des éléments de la notation sans préciser de méthodologie particulière de sa mise en œuvre.

La modélisation des processus métiers s'effectue en rassemblant ces éléments sur un diagramme de processus métier « Business Process Diagram (BPD)».

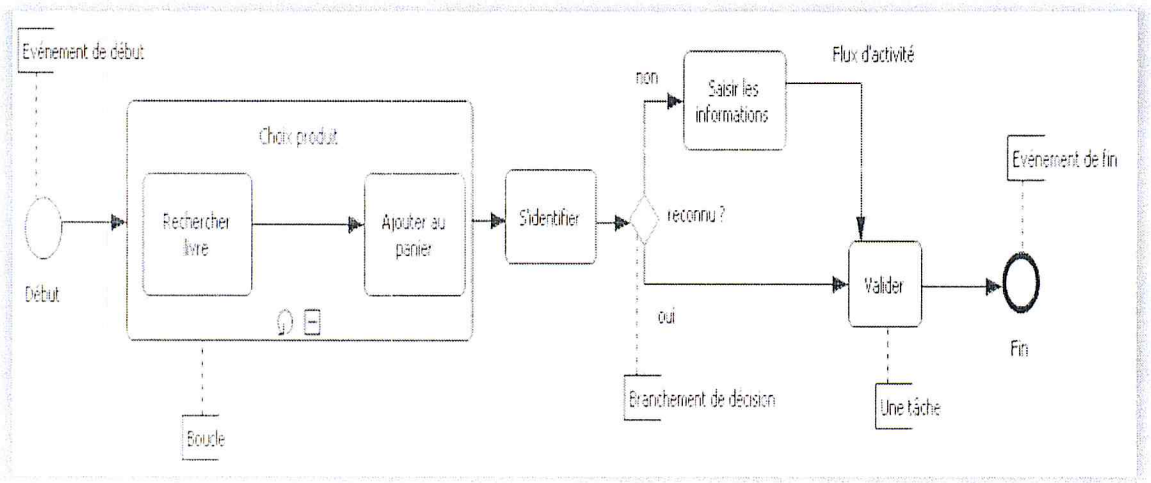


Figure.24: Exemple d'un BPD.

b) La modélisation proposée

L'appel d'offre passe par plusieurs opérations : des mouvements physiques et des opérations logiques ; les interactions entre les acteurs du système et les tâches qu'ils effectuent représentent les processus métiers de base. Dans notre système de « l'appel d'offre » on a ressorti un processus principal qui se focalise sur l'aspect décisionnel et des sous processus réalisés par le décideur qui est le responsable des stocks. Pour représenter les processus métier nous utilisons la notation BPMN (Business Process Modelling Notation).

Les acteurs qui participent dans ce processus sont les suivant :

Acteur	Description
Fournisseur	Celui qui pourra quant à lui répondre à un appel d'offre si le créateur de ce dernier l'y autorise.
Maitre d'ouvrage	Qui fait le Lancement des appels d'offres
Expert	Qui fait l'étude de base
Décideur principal	Qui fait la sélection de la meilleure soumission
Décideur secondaire	Qui fait la négociation avec décideur principal

Tableau. 5: Les acteurs de processus métier.

La figure suivante illustre le processus principale de traitement d'un appel d'offre

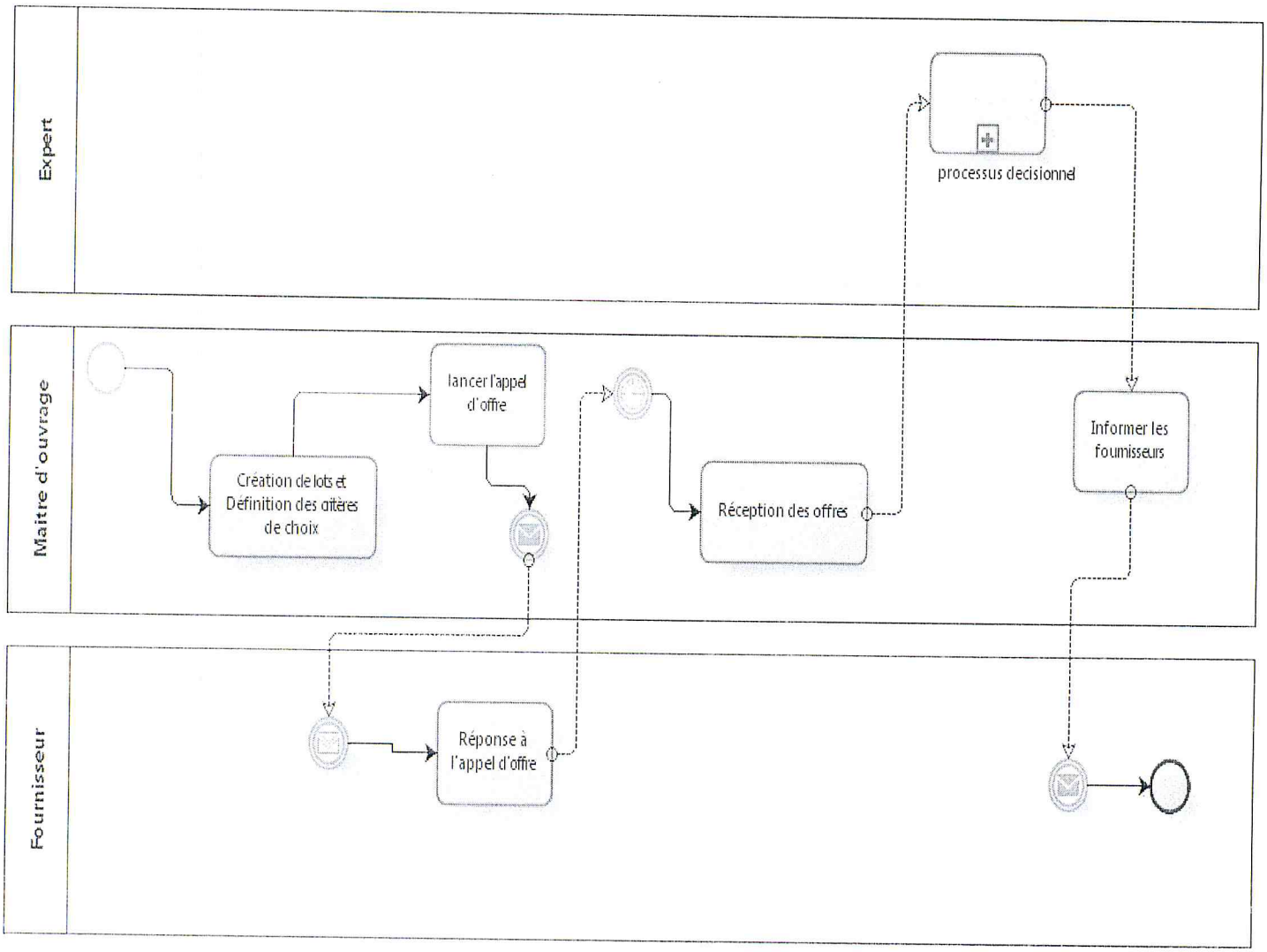


Figure. 25: Processus principal « appel d'offre ».

- Sous-processus«Processus d'aide à la décision»

Cette figure qu'elle montre le sous-processus d'aide à la décision

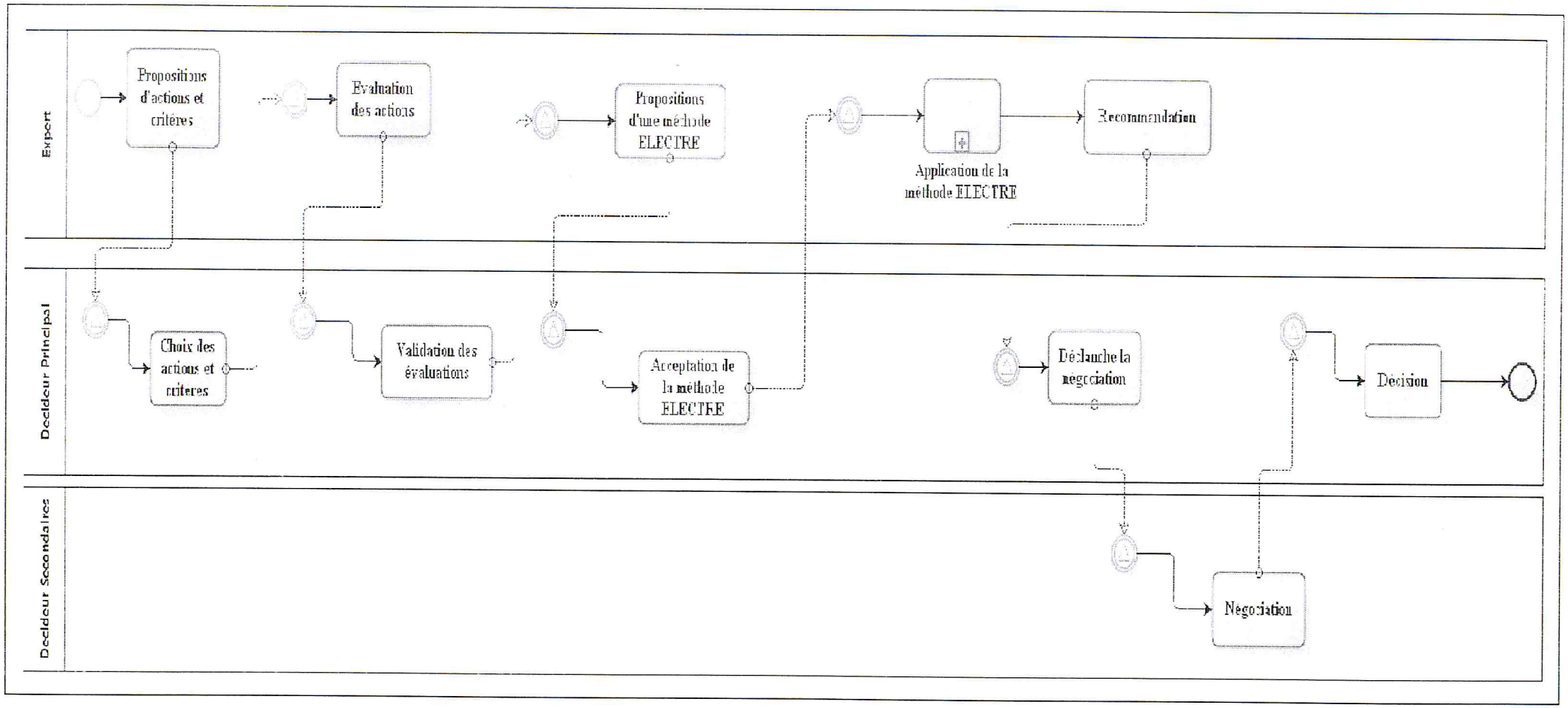


Figure. 26: Sous-processus « aide à la décision ».

- Sous-processus«La méthode ELECTRE»

Et à la fin, la figure suivante décrit le sous-processus de la méthode ELECTRE

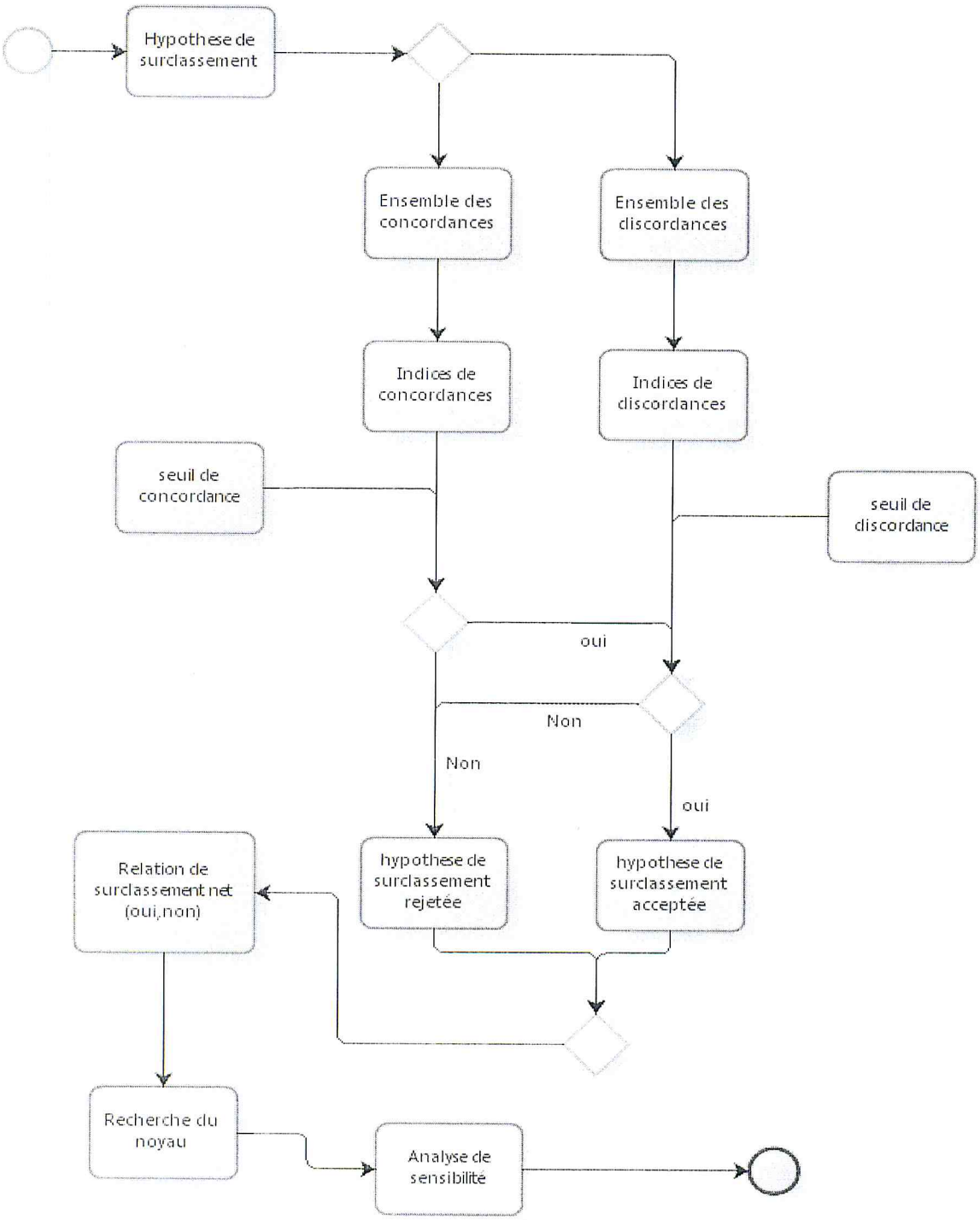


Figure. 27: Sous-processus « ELECTRE ».

V. PIM

Ce modèle décrit l'analyse et l'architecture du système pour identifier les services de notre système, nous utilisons les diagrammes SOAML. Dans cette phase nous allons décrire le diagramme de classe et les diagrammes d'architectures (SOAML) après la transformation de CIM vers PIM.

a. Diagramme de classe

Pour exprimer de manière générale la structure statique de notre système, le diagramme de classes est utilisé, ce présent diagramme permet de décrire les liens entre les objets du système.
Pour le cas de notre système, nous avons élaboré le diagramme de classes suivant :

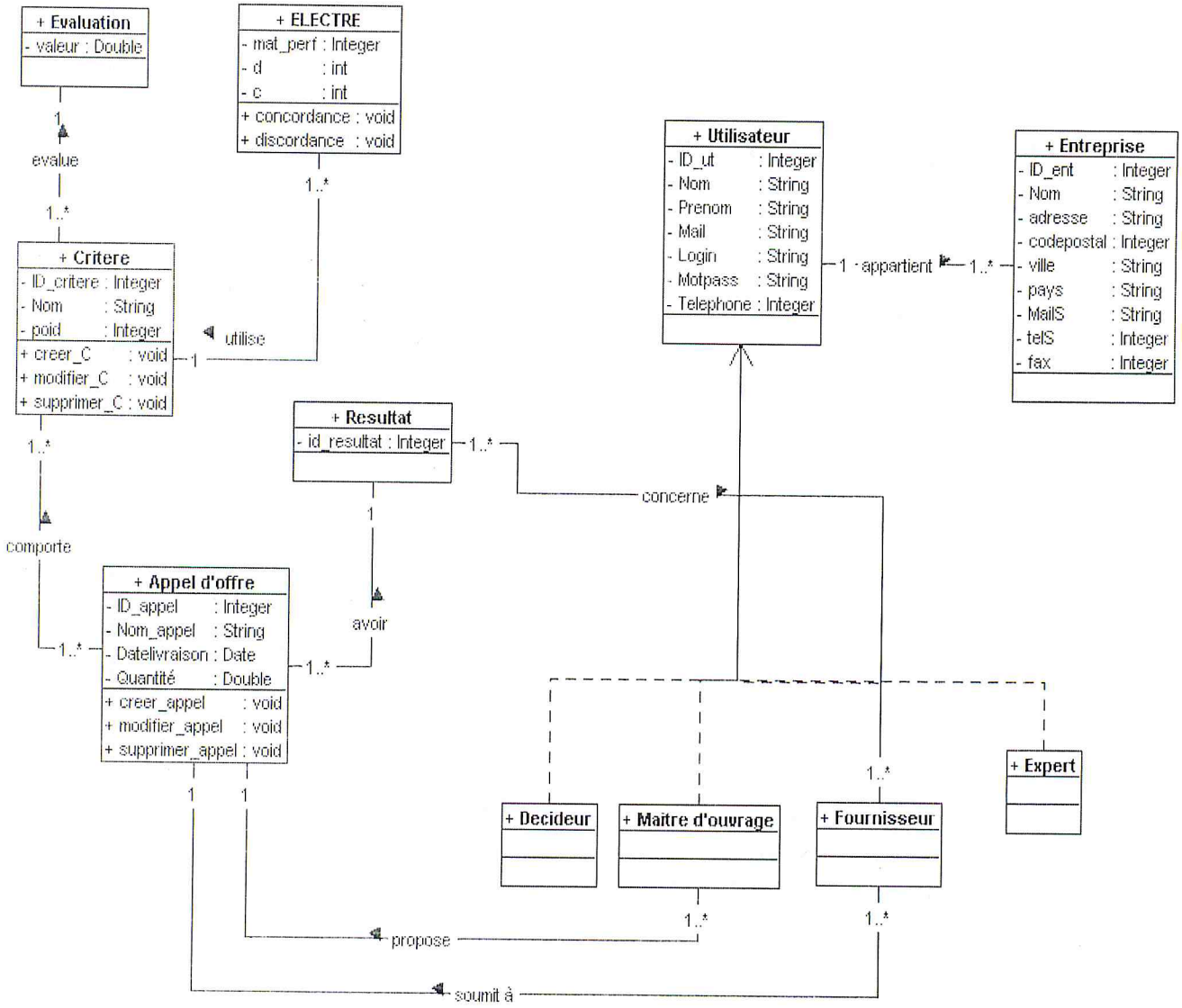


Figure. 28: Diagramme de classe du système.

Description des classes

Classes	Attributs/Méthodes	Rôles des attributs/méthodes
Critère	**Attributs:**	
	ID-critere	Identifiant du critère
	Nom	Nom du critère
	poid	poid du critère
	Méthodes:	
	créer_C()	créer un critère
	modifier_C()	modifier un critère
	supprimer_C()	supprimer un critère
Resultat	**Attributs:**	
	ID_Resultat	Identifiant du resultat
ELECTRE	**Attributs:**	
	Mat_perf	Matrice des performances
	c	Seuil de concordance
	d	Seuil de discordance
	Méthodes:	
	Concordance()	Calculer la matrice de concordance
	Discordance()	Calculer la matrice de discordance
utilisateur	**Attributs:**	
	ID_ut	Identifiant de l'utilisateur
	Nom	Nom de l'utilisateur
	Prenom	Prénom de l'utilisateur
	Mail	L'adresse mail de l'utilisateur
	Login	Pseudo de l'utilisateur
	Motpass	Mot de passe de l'utilisateur
Entreprise	**Attributs:**	
	ID_ent	Identifiant de l'entreprise
	Nom	Nom de l'entreprise
	adresse	Adresse de l'entreprise
	codepostal	Code postal de l'entreprise
	ville	Ville de l'entreprise
	pays	Pays de l'entreprise
	MailS	Adresse mail de l'entreprise
	telS	Numéro téléphone de l'entreprise
	faxS	Numéro Fax de l'entreprise
Appel d'offre	**Attributs:**	
	ID_appel	Identifiant de l'appel d'offre
	Nom_appel	Nom de l'appel d'offre
	Datedebut	Date début de soumissionnement
	Datefin	Date fin de soumissionnement
	Quantité	Quantité
	Méthodes:	
	créer_appel()	créer un appel d'offre
	modifier_appel()	modifier un appel d'offre
	supprimer_appel()	supprimer un appel d'offre
Evaluation	**Attributs:**	
	Valeur	Valeur du critère

Tableau. 6: Description des classes.

b. Les services par SoaML :

Le diagramme Architecture des services (ServicesArchitecture) : ce diagramme permet de décrire la manière dont les participants (acteurs) s'échanges des services, ces dernières sont exprimés par des contrats de services **[Jean Marie B,12].**

Chaque service est représenté par un **"ServiceContract"** qui définit le rôle joué par chaque participant du service (consommateur et fournisseur) ainsi que les interfaces qu'ils mettent en œuvre pour compléter ce rôle **[Jean Marie B,12].**

Les schémas ci-dessous représentent l'architecture des services ainsi que les contrats des services :

- **Diagramme architecture des services du processus principal**

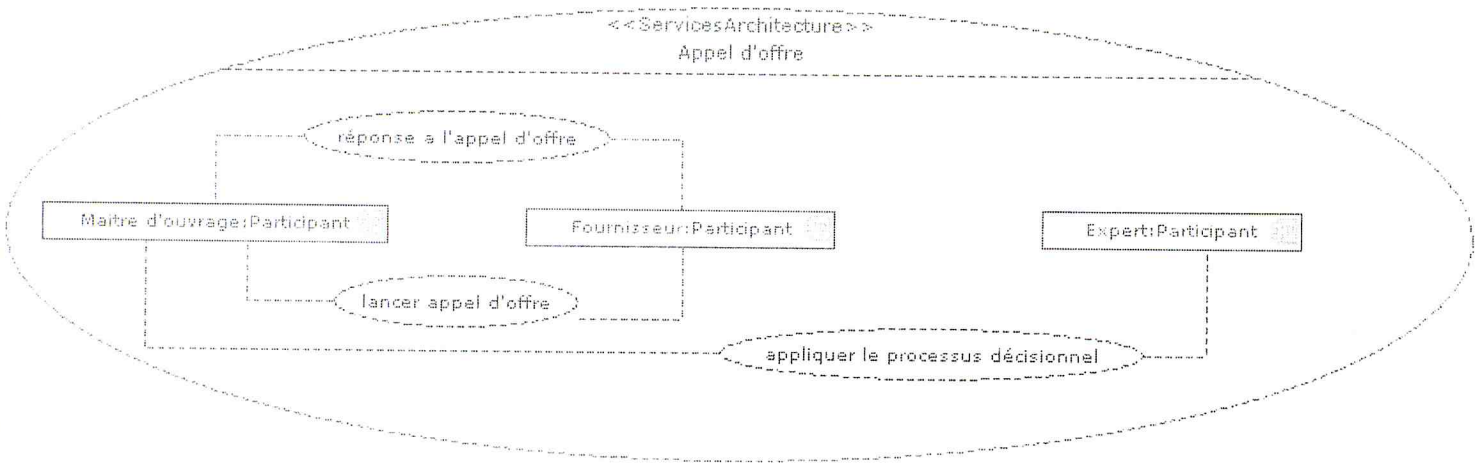


Figure. 29: Diagramme architecture des services « appel d'offre ».

➢ **Diagramme des contrats de services du processus principal**

Les diagrammes suivant expliquent les contrats de services qui interviennent dans notre système

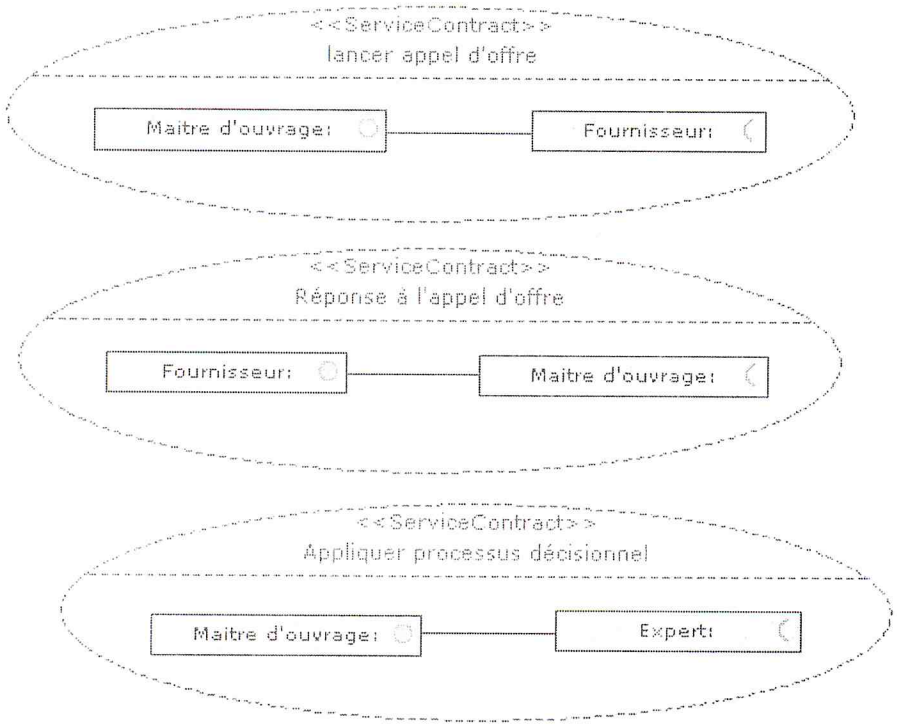


Figure. 30: Diagramme contrat de services « appel d'offre ».

➢ **Diagramme architecture des services du processus d'aide à la décision**

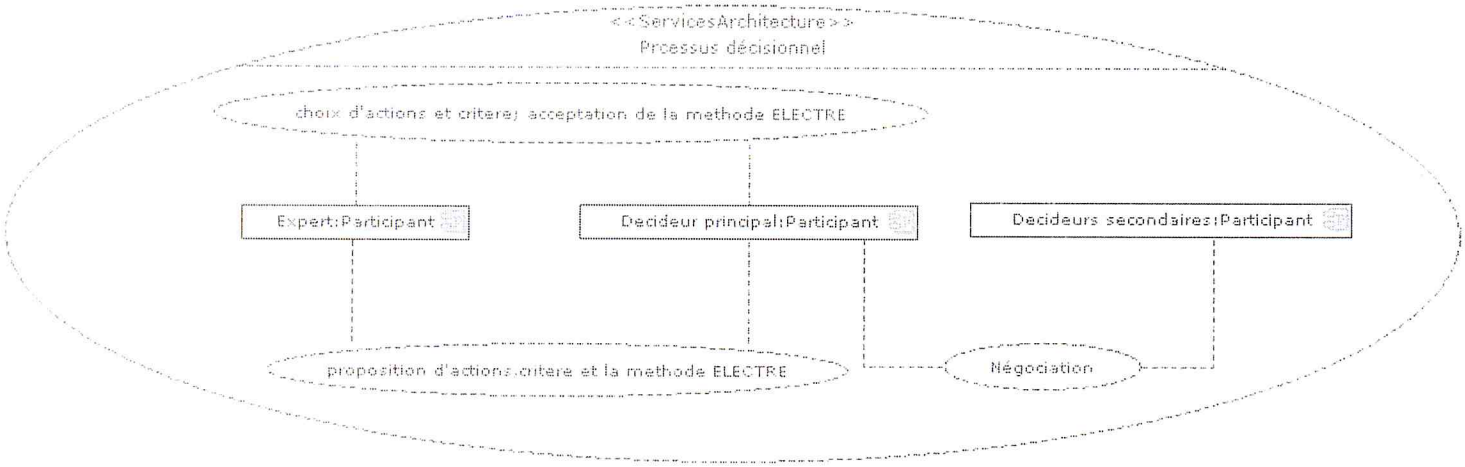


Figure. 31: Diagramme architecture des services « aide à la décision ».

➢ **Diagramme des contrats de services du processus d'aide à la décision**

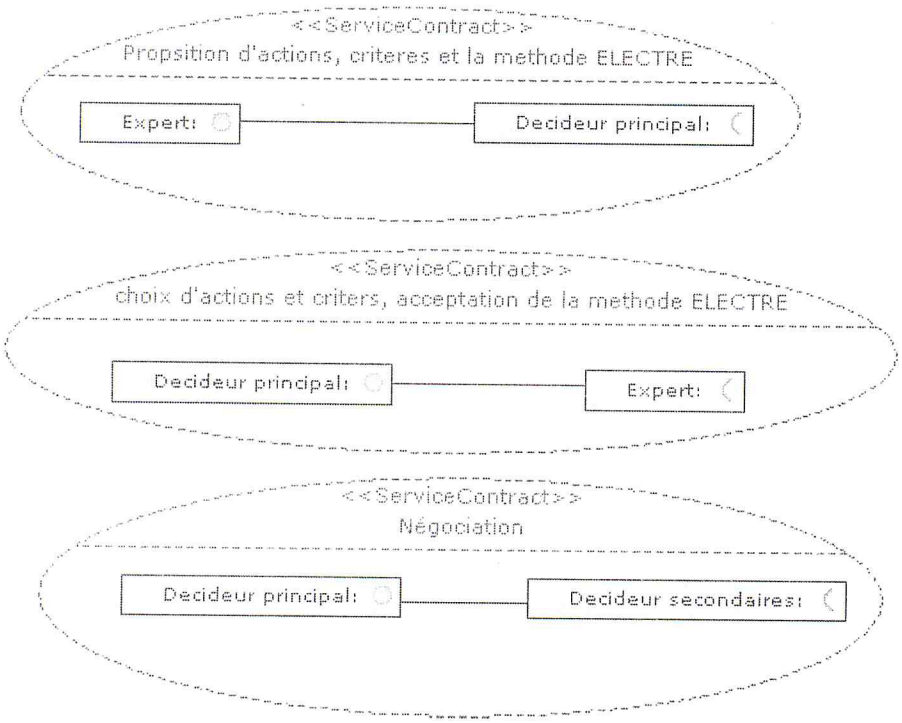


Figure. 32: Diagramme contrat de services « aide à la décision».

Les règles de correspondance entre le BPMN et SOAML

Règle 0 : Le processus à l'architecture de services : L'architecture de services est alignée avec le processus métier. Les participants sont dérivés des Pools ou bien des lignes et les contrats de services sont dérivés des activités ou bien des interactions entre participants.

Règle 1:La Tâche à l'action UML:

Une tâche décrit une activité qui être consommé par les acteurs du processus. Elle représente une 'interface abstraite pour le travail accompli sans donner plus de précisions sur le flux de travail exécuté. Le tableau 1 illustre la correspondance des notations.

Table 1. Task to UML Action

	BPMN	SoaML
Construct	Task	Action
Notation		

Règle 2: Sous-processus à L'architecture de services

Un sous-processus représente une fonction plus complexe que la tâche. Il est à mentionner que l'architecture des services correspondant au sous-processus n'est pas de faible granularité. Le tableau 2 illustre la correspondance des notations.

Table 2. Sub-Process to Services Architecture

	BPMN	SoaML
Construct	Sub-process	Services Architecture
Notation		

Règle 3 : Les groupements (Pool) au participant

Un Pool représente un participant important dans un processus en BPMN. Il peut également être structuré à l'égard des autres participants du processus, créant ainsi un des participants hiérarchiques. On représente le Pool dans le modèle SoaML par un rôle dans une architecture de services qui dispose d'un participant de type Pool. Le tableau 3 illustre la correspondance des notations.

Table 3. Pool to Participant (Community-level)

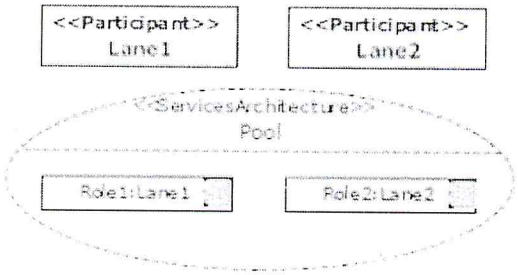
	BPMN	SoaML
Construct	Pool	Participant Role in a Community-level Services Architecture
Notation		

Règle 4: La Ligne (Lane) au participant

Une ligne représente un participant ou un département en BPMN et est situé dans un Pool, montrant ainsi la hiérarchie à deux niveaux. Afin de montrer la possibilité pour une nouvelle subdivision,

la ligne est représentée par un rôle dans un participant au niveau de l'architecture des services de type ligne. L'architecture des participants du type ligne doit adhérer à l'architecture des services dont les participants du type Pool (les lignes sont contenues dans ces Pool). Le tableau 4 illustre la correspondance des notations.

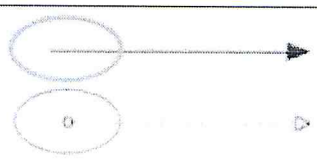
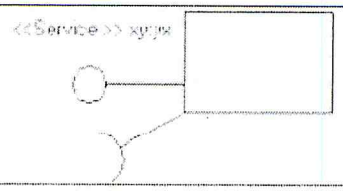
Table 4. Lane to Participant

	BPMN	SoaML
Construct	Lane	Participant
Notation		

Règle 5: Le début de Flux de message au service

Le point de départ de chaque message dans BPMN ,il représente un canal de transmission de données entre deux participants ou deux Pools. Dans le SoaML ça représente un port d'un service. Le tableau 5 illustre la correspondance des notations.

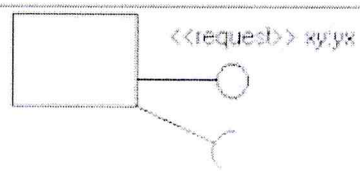
Table 5. Message "Begin" to Service

	BPMN	SoaML
Construct	Message "Begin"	Service
Notation		

Règle 6: La fin de Flux de message à la demande :

Le point de fin de chaque message dans BPMN, il représente un canal de transmission de données entre deux participants ou deux Pools. Dans le SoaML ça représente un port de demande d'un service. Le tableau 6 illustre la correspondance des notations.

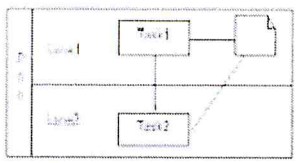
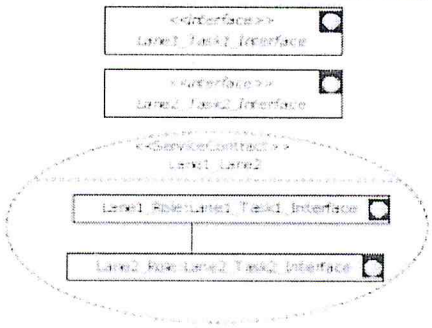
Table 6. Message "End" to Request

	BPMN	SoaML
Construct	Message "End"	Request
Notation		

Règle 7: fragment de processus (modèle) au contrat de service :

Il n'ya pas de construction unique en BPMN qui ressemble à un contrat de service . Vous avez besoin d'analyser les processus BPMN et d'identifier les fragments de processus qui peuvent être mappées à des contrats de service. Un contrat de service définit un cahier des charges de service qui définit le rôle joué par chaque participant dans le service, et les interfaces qu'ils mettent en œuvre pour compléter ce rôle. On peut cependant, définir un modèle BPMN qui peut être associé à un contrat de service. Le tableau 7 illustre la correspondance des notations.

Table 7. Process fragment (pattern) to Service Contract

	BPMN	SoaML
Construct	Lane1 → Lane2	Service Contract
Notation		

c. **La conception Globale**

Dans cette phase nous présentons notre architecture qui prend en charge le processus d'aide à la décision. L'architecture proposée ci-dessous est composée de quatre couches conformément au modèle IBM [Dodani, 2006] et l'architecture de [Cui Lin et al, 2009], de [Bonnet, 2006] et de [Raymond, 2006], et de [Xavier, 2006] :

Cette nouvelle architecture doit permettre de prendre en considérant le processus métier de l'entreprise et certain aspects décisionnel exprimés par Electre. Elle est ainsi séparée en quatre couches, comme suit :

- La couche « **Données** » : elle contient deux sous couches, la première c'est « Métier » qui englobe les services métiers « CRUD » du processus métier exécuté dans l'entreprise. La deuxième sous couche et « Information » a pour but de sauvegarder les services indice.
- La couche « **Technique** » : elle contient deux sous couches : la sous couche « Fonction » qui représente les services fonctions du processus métier. La sous couche « Evaluation », qui inclut les services.
- La couche « **Processus** » : elle contient deux sous couches : Applicatifs et Electre. La première regroupe les services applicatifs du processus métier, et la deuxième assure le processus de désignation ou en utiliser Electre.
- La couche « **Présentation** » : elle contient des interfaces, et assure la communication entre l'utilisateur et le système, soit pour exécuter le processus métier, ou pour prendre une décision.

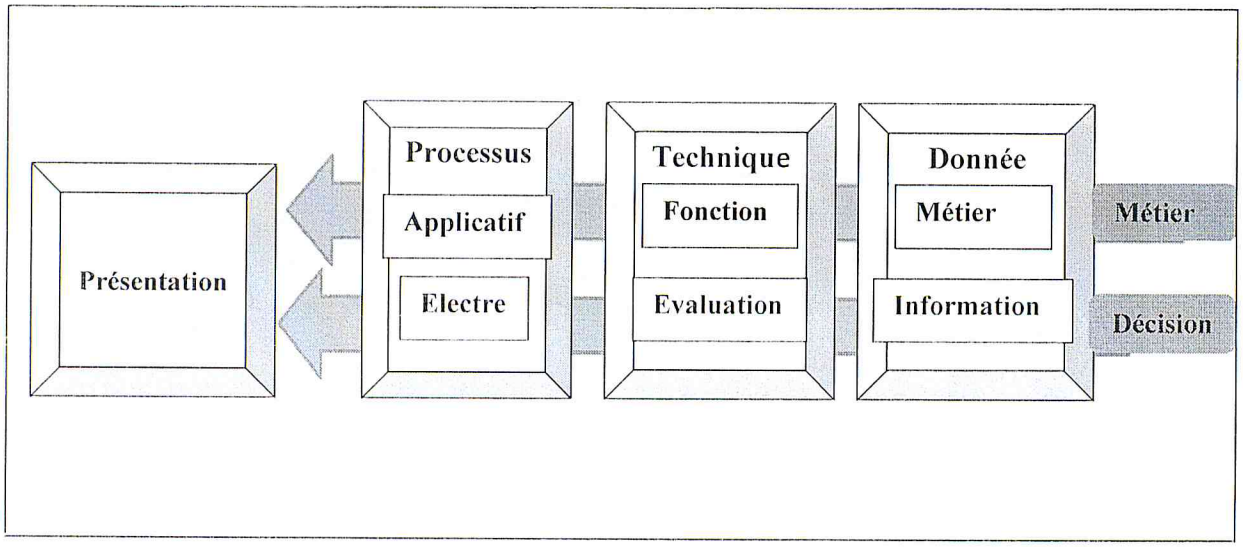


Figure. 33: L'architecture SOA proposée

Application de l'architecture SOA proposée sur notre système :

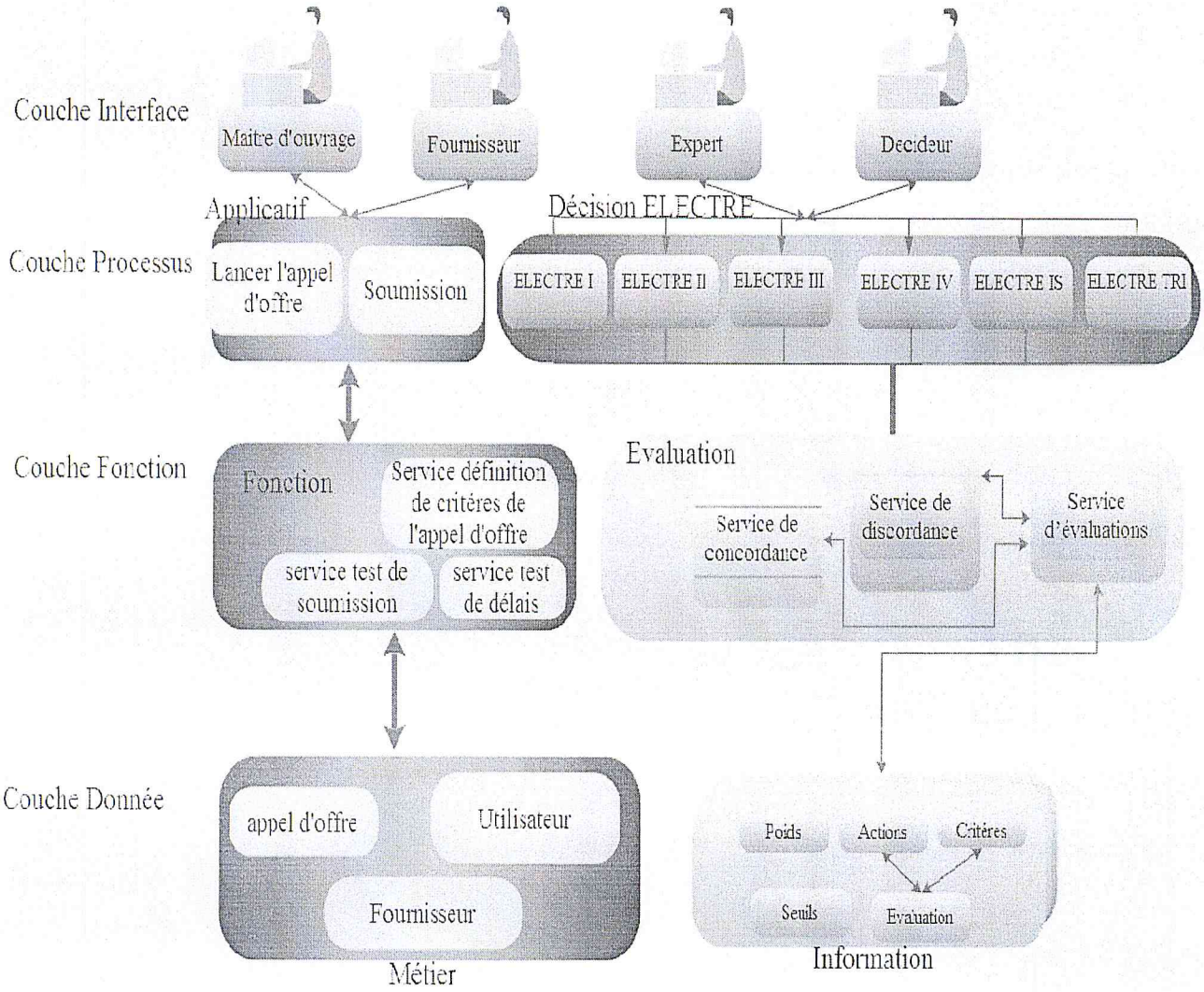


Figure. 34:Application de l'architecture proposée sur le système

VI. PDM :

Dans cette phase nous allons décrire la plate-forme d'implémentation choisie ainsi que les caractéristiques techniques de celle-ci. La description de cette plate-forme est matérialisée par nos choix techniques.

a. La plateforme de développement:

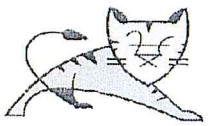
Actuellement,les principaux environnements d'exécutions des services sont représentés essentiellement par les infrastructures à base de composants distribués(orientés services) du marché : Microsoft.Net et J2EE(Java 2 Entreprise Edition).

Notre choix s'est fait sur le Framework J2EE qui est un ensemble de spécifications pour le langage de programmation Java de Sun, plus particuliérement distiné aux application d'entreprises.

b. Serveur de composants :

Pour notre part nous avons utilisé le serveur de composants TOMCAT 6.0 puisque il implimente les spécifications des servelets et des JSP. Il est paramétrable par des fichiers XML et inclut des outils pour la configuration et la gestion. Il comporte également un serveur http.

⇨ **Présentation du conteneur Web TMOCAT 6 :**



Apache Tomcat est un conteneur libre de servlets et JSPs Java EE. Issu d'un projet principal de **l'Apache Software Foundation**, Jakarta. Tomcat ne constitue qu'un conteneur web, et non un serveur Web : il gère spécifiquement les servlets et les JSPs. Il peut être également parfois désigné comme un moteur de servlets.

Tomcat est utilisé en combinaison avec un serveur Web Apache ou d'autres serveurs Web comme : JBoss,WebSphere… etc

Tomcat a été écrit en langage Java. Il peut donc s'exécuter via la JVM, il peut donc s'exécuter sur n'importe quel système d'exploitation la supportant.

c. Spécification des interfaces Homme/Machine :

Pour le développement des composants graphyques de notre système nous proposons de le faire avec la technologie JSF. Le choix de ce dernier se justifie par son utilisation avec des langages Web dynamiques (JSP) à l'aide d'outils relativent simples d'utilisation. Il permet ainsi de développer des applicatios dynamiques sans connaissance préalable des langages de programmations.

➢ **Pourquoi JSF ?**

JSF fait partie du standard J2EE, c'est un framework d'interface utilisateur pour les applications web, basé sur les technologies JSP et les Servlets.

JSF permet :

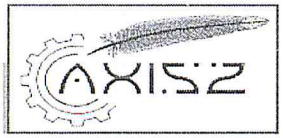
- une séparation nette entre la couche de présentation et les autres couches.
- le mapping HTML/Objet.
- un modèle riche de composants graphiques réutilisables.
- une gestion de l'état de l'interface entre les différentes requêtes.
- une liaison simple entre les actions côté client de l'utilisateur et le code Java correspondant côté Serveur.

d. L'implémentation des services et leurs contrats :

L'implementation d'un service doit étre décrite en XML, vu qu'il correspond à un composant logique, pour cela nous avons opté pour le serveur de déploiement AXIS 2. Notre choix dépond de son fonctonnement particulier avec TOMCAT; en effet AXIS est installé comme une application Web au sien du moteur de servelets et de JSPs Apache TOMCAT.

Il est capable de fournir des outils pour créer automatiquement les WSDL correspondant à des classes Java ou inversement pour créer les classes Java sur la bas d'un WSDL.

⇨ **Présentation du serveur de déploiement des services AXIS 2 :**



Axis est un projet de l'Apache ftware Foundation. C'est un package Java libre qui fournit plusieurs fonctionnalités :

- un environnement qui peut fonctionner comme un serveur indépendant soit comme un plug-in de moteurs de servlets (en particulier Tomcat),
- une API pour développer des services web SOAP,
- des outils pour déployer, tester et monitorer des web-services.

Axis2 est une réécriture complète pour objectif d'être plus efficace, plus modulaire et plus orienté XML que la version précédente. Un certain nombre de modules sont en cours de développement concernant la sécurité, les transactions...etc.

e. Orchéstration des services :

Le BPM est devenu un intermédiaire incontournable entre l'IT et le métier. **Intalio|BPMS** est le premier dans le monde Open Source. Il se base sur le standard BPMN pour la modélisation des processus métiers. A partir du BPMN il génère du BPEL pour l'orchestration des processus. Il est conçu pour les utilisateurs ayant peu d'expérience technique..

⇨ **Présentation de la solution INTALIO|BPMS**

INTALIO Techniquement la solution Intalio|BPMS est architecturée autour des composants suivants :

- Intalio|Designer est l'outil de modélisation et de conception des processus métiers fondé sur la plate forme Eclipse. Permet de créer les diagrammes de processus à l'aide de la notation BPMN et de compléter ces modèles en intégrant les Services Web et autres détails nécessaires à l'exécution des processus métiers. Cet outil traduit automatiquement les diagrammes BPMN en code source BPEL d'exécution des processus.

- Intalio|Server comporte le moteur d'exécution des processus métiers en interprétant le langage BPEL. Il propose également l'ensemble des fonctionnalités d'administration.

- Intalio|Workflowest basé sur la nouvelle extension BPEL4People il supporte l'exécution des activités humaines par le biais d'interfaces utilisateur AJAX générées automatiquement par le moteur XForms. Intalio|Workflow utilise Intalio|Server pour l'exécution des processus BPMN.

⇨ **Formulaires**

Qui dit processus d'entreprise dit aussi interface graphique orientée utilisateur final. On parle donc ici de formulaires web.

- Outre l'édition des processus, Intalio offre aussi une interface graphique permettant de dessiner des XForms et d'appliquer un certain nombre de contraintes (champs obligatoires, types de champ...). Dans le cadre de notre projet, nous préférons d'utiliser les formulaires AJAX munies d'un ensemble de composant interface d'utilisateur plus riche que les XForms, ainsi l'environnement de réalisation de ces formulaires est aussi intégré dans Intalio|Designer.

f. **Le choix d'un système de gestion de base de donnée** :

Le système de Gestion de Base de Données que nous avons choisi est PostgresSql qui est un SGBD relationel fonctionant pratiquement sur tous types de système. Notre choix de PostgresSql version 9.0 se justifie par la disponibilité gratuite et aussi son fonctionnement selon l'architecture client/serveur.

g. **L'environnement de développement :** La figure suivante montre notre choix technique

Notre choix s'est porté vers l'EDI, qui permet de créer des applications Java EE et Web.

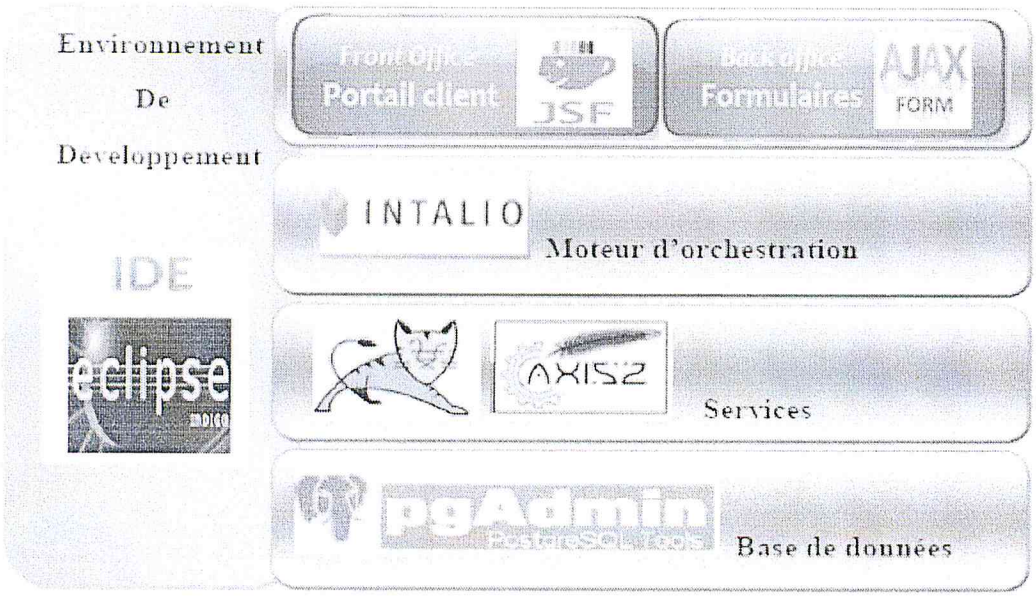


Figure. 35: Les choix techniques

VII. Conclusion :

Dans ces trois premiers modéles (CIM, PIM, PDM) nous avons présenté la démarche de développement de notre application, en appliquant la méthode MDA, suivant le dernier modéle (PSM) qui exprime les étapes d'utilisation de ses aspects.

VIII. PSM

Après avoir fixé les choix techniques dans l'étape précédente, nous présentons dans cette étape un aperçu général sur la réalisation de notre solution. Ensuite, nous illustrons les différentes vues de l'application implémentée.

Nous présentons dans cette section les étapes de réalisation de notre application strate par strate, de bas en haut.

1) La couche métier

Pour chaque machine logique (MLM) de la couche métier correspond un service CRUD (acronyme anglais qui signifie Create Read Update Detete), qui permet de créer, rechercher, mettre à jour et supprimer l'information dans les référentiels du système d'information.

Ces services sont réalisés par des classes JAVA simple, on évitera donc d'alourdir l'application par les appels à des services web réalisant de simples opérations.

DemandeurBDService.java

```
+ import entitesBD.Demandeur;

  public class DemandeurBDService {
      ConnexionBD bd;

+     public DemandeurBDService() throws Exception {

      /** Ajouter un demandeur*/
+     public void addDemandeur(Demandeur demandeur) throws Exception {

      /** Modifier un demandeur*/
+     public void updateDemandeur(Demandeur demandeur) throws Exception {

      /** Supprimer un demandeur*/

+     public void deleteDemandeur(int id) throws Exception {

      /** Obtenir list demandeur */

+     public ArrayList<Demandeur> getAllDemandeur() throws Exception {

      /** Obtenir un demandeur */
+     public Demandeur getDemandeur(int code) throws Exception {
```

Figure 36 : Service CRUD.

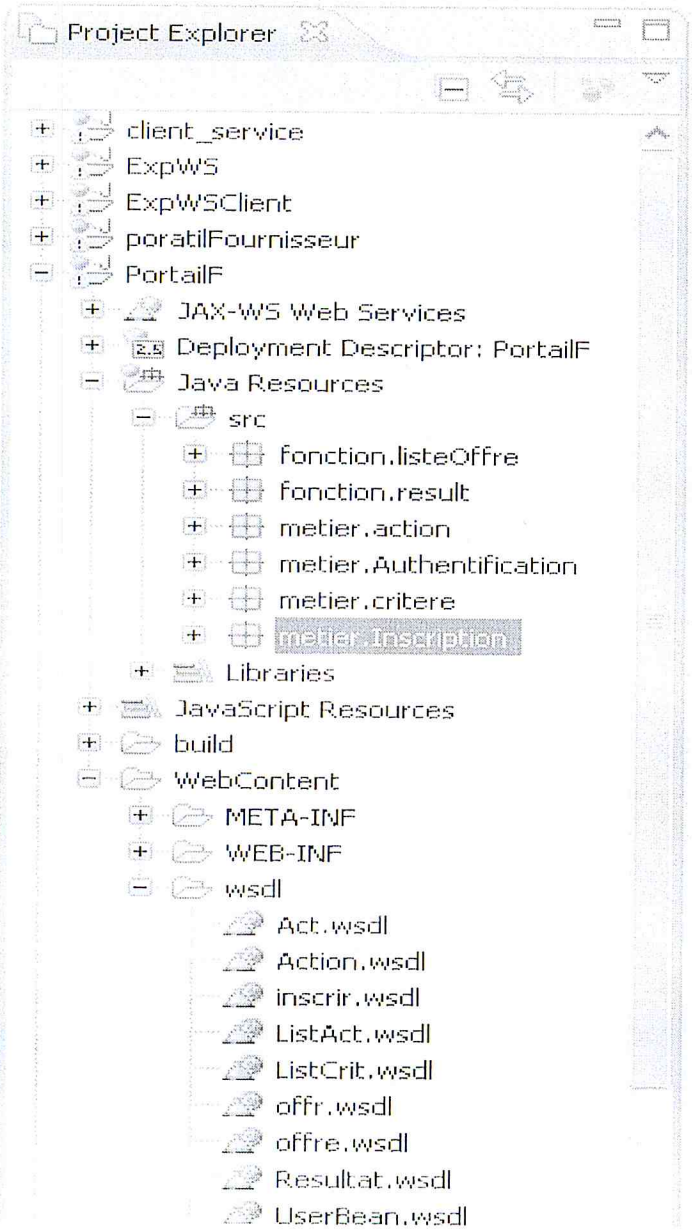


Figure 37 : Les couches métier et fonction.

2) La couche fonction

Les machines logiques (MLO) de la couche organisation (Figure 37) sont réalisés par des services web, ces derniers seront par la suite orchestrés par le processus BPM, ou appelés directement par le portail client. Le processus BPM à son tour devient un WS.

a) Les Services Web

Deux approches peuvent être utilisées pour la création de Web Service :

➢ Bottom-Up

Création du logique métier dans un langage donné, puis génération du WSDL et des classes nécessaires. Cette approche masque la partie définition WSDL, elle permet surtout de réutiliser du code existant.

➢ Top-Bottom

Définition d'un fichier WSDL puis génération de code vers un langage donné. Cette approche est indépendante de la plateforme de destination, mais nécessite une bonne maîtrise de la notion de langage de définition d'interface.

Les Services Web de notre application sont créés suivant l'approche Bottom-Up à l'aide de l'Axis runtime et déployés sur le serveur Axis.

➢ Exemple d'un fichier WSDL

Un fichier WSDL contient une description de tout ce qui est nécessaire à l'appel d'un Service Web :

- L'URL et l'espace de noms du service.
- Le type de Service Web.
- La liste des fonctions disponibles.
- Les arguments de chaque fonction.
- Le type de données de chaque argument.
- La valeur de retour de chaque fonction et le type de données de chaque valeur de retour.

Figure 38 : Exemple d'un fichier WSDL.

b) Les processus métier

Les modèles produits durant l'étape de conception adressent une réponse organisationnelle orientée « métier » et de gestion des individus, alors il faut les transposés en modèles de processus exécutables représentés avec la notation BPMN en utilisant l'outil choisi « Intalio|Designer ».

Ces modèles sont automatiquement traduits en langage d'exécution des processus métiers BPEL et exécutés directement sur un système de gestion des processus métiers « Intalio|BPMS ».

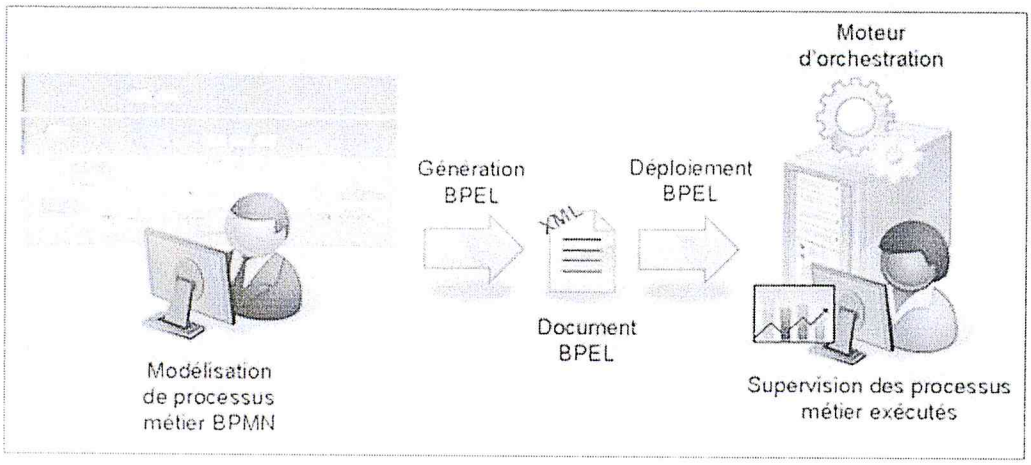


Figure 39 : Les étapes de transposition des processus métiers.

La mise en œuvre des processus métier avec Intalio|BPMS est réalisée en plusieurs étapes :

1. La définition des rôles des intervenants du Workflow se traduits en pool dans le diagramme BPD. La Figure suivante illustre la représentation des intervenants des processus dans sa définition graphique.

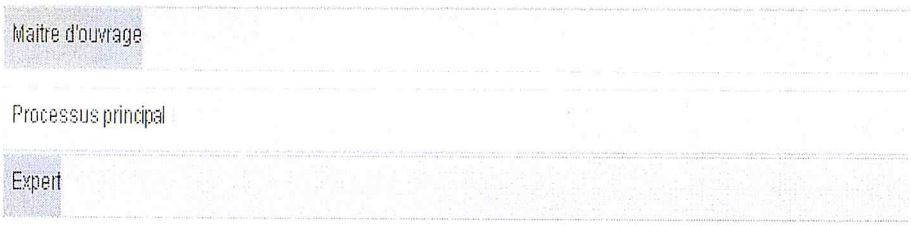


Figure 40: Définitions des rôles.

2. La définition du flux d'activités du processus principal.

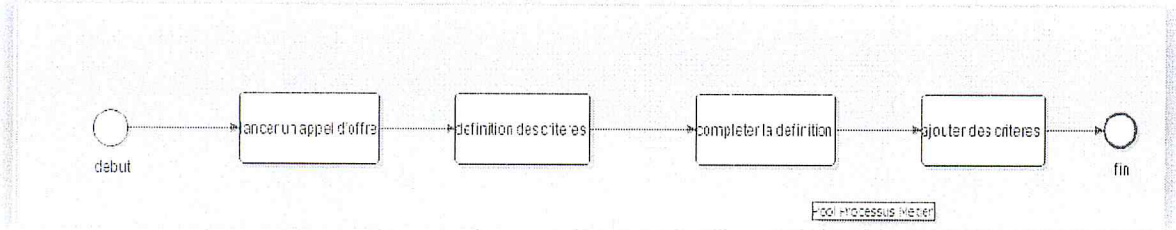


Figure 41: Partie du processus principal

3. Intégration des formulaires utilisés dans les étapes du Workflow. Ces formulaires représentent les tâches à réaliser ou les notifications provenant de l'exécution du processus.

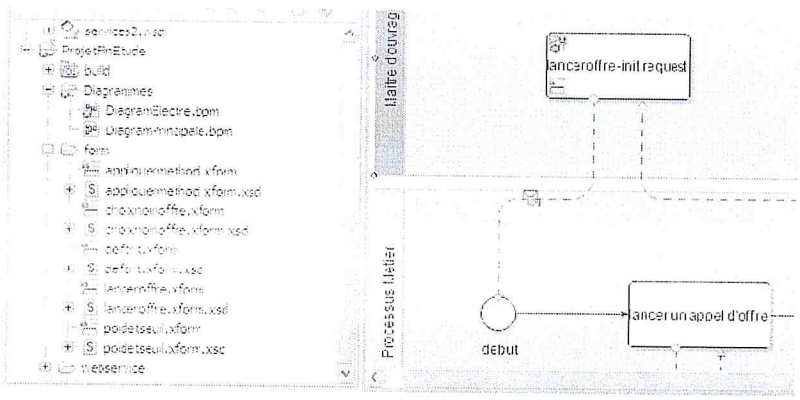


Figure 42: Intégration des formulaires.

4. L'intégration des Services Web impliqués dans l'exécution du processus métier. L'interface de description de Service Web WSDL est utilisée comme moyen de référencement d'intégration.

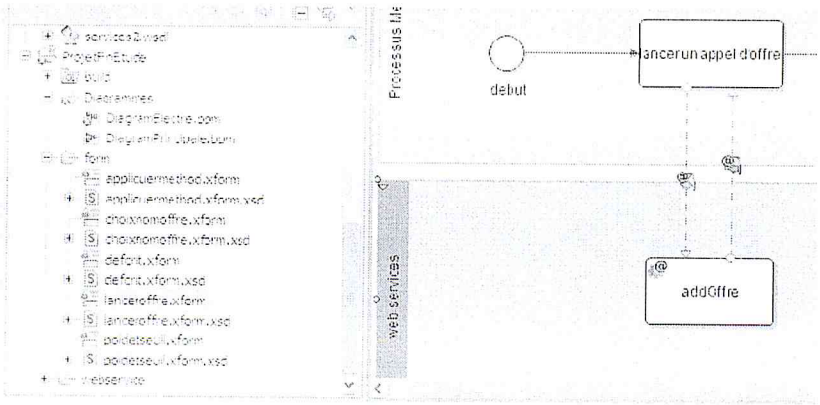


Figure 43 : Intégration des Services Web

5. Le déploiement des diagrammes BPD sur Intalio|Server. Un diagramme BPD déployé sur le serveur devient lui-même un service Web. Il est éventuellement appelé depuis le portail client.

3) La couche présentation

a) Les formulaires

Ces formulaires représentent les tâches à réaliser ou les notifications provenant dans l'exécution du processus. Les données saisies sont véhiculées tout au long de l'exécution des tâches du processus.

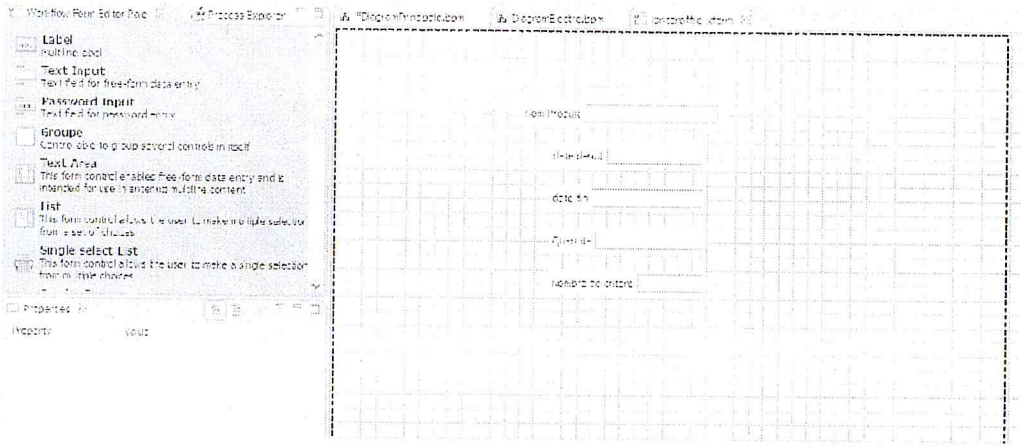


Figure 44 : Formulaire Workflow Form « Créer un appel d'offre».

b) Le portail client

Le portail client regroupe un ensemble de vues réalisées à l'aide des pages JSP, ces pages font appel aux services web et processus de la couche fonction.

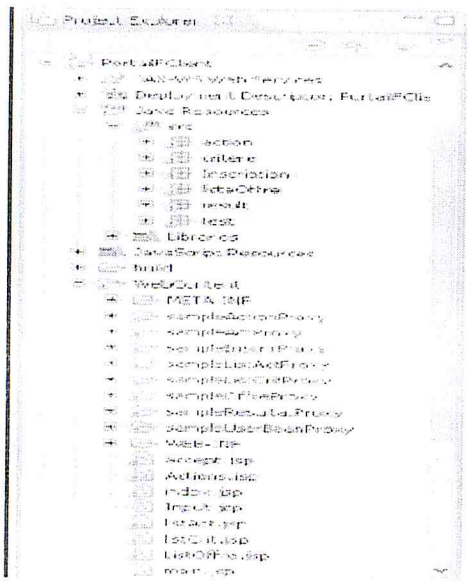


Figure 45 : L'application web « Portail client ».

IX. Démonstration

Dans cette section nous présentons les différentes vues d'exécution de notre application.

1) Le portail Fournisseur

La figure suivante illustre « la page d'authentification » du portail fournisseur :

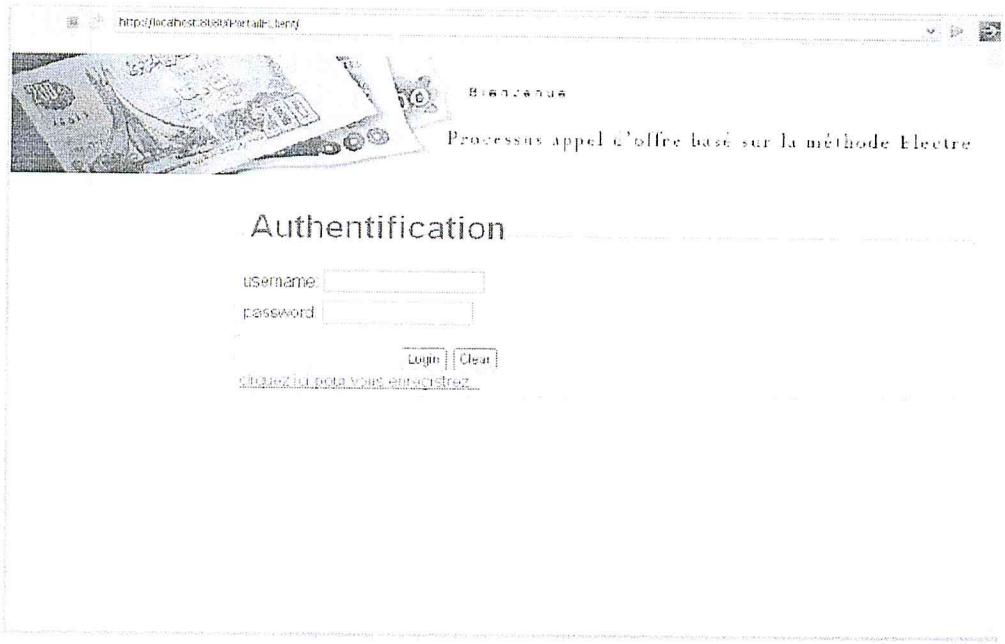


Figure 46 : Portail fournisseur « Authentification ».

2) Les formulaires

Les agents peuvent procéder au traitement des demandes en accédant à leurs comptes via l'interface Intalio|Workflow :

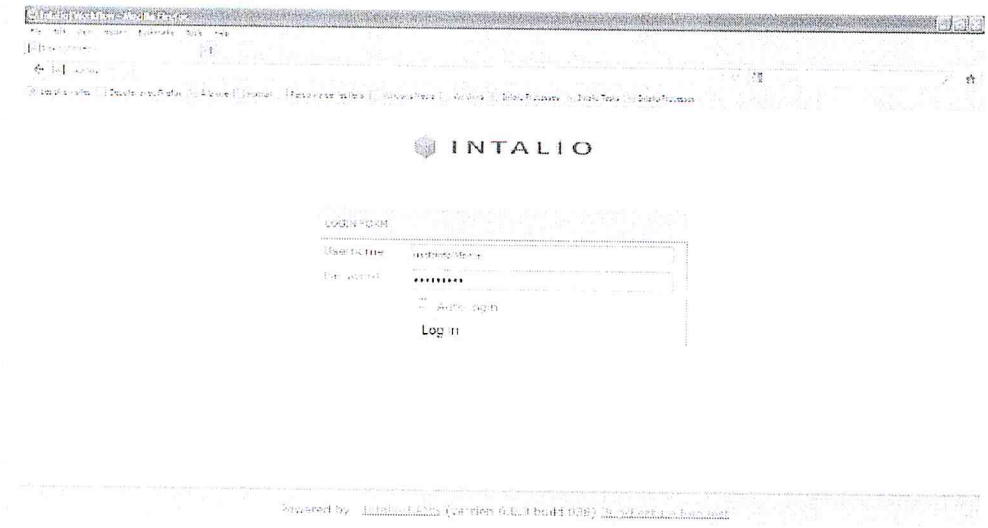


Figure 47 : Intalio|Workflow «Authentification».

Après l'authentification, la fenêtre suivante apparaît à l'agent :

Figure 48 : Intalio|Workflow «Liste des tâches».

Cette fenêtre présente les tâches associées à cet utilisateur, les messages qu'il a reçus ainsi que les processus qu'il peut déclenchés.

Un formulaire apparaît à l'utilisateur dès qu'il a choisi la tâche à réaliser. Il peut ainsi:

- Enregistrer le formulaire en choisissant le bouton « Save », la tâche reste alors en attente.

- Accomplir la tâche en choisissant le bouton «Complete », dans ce cas le processus continue à s'exécuté.

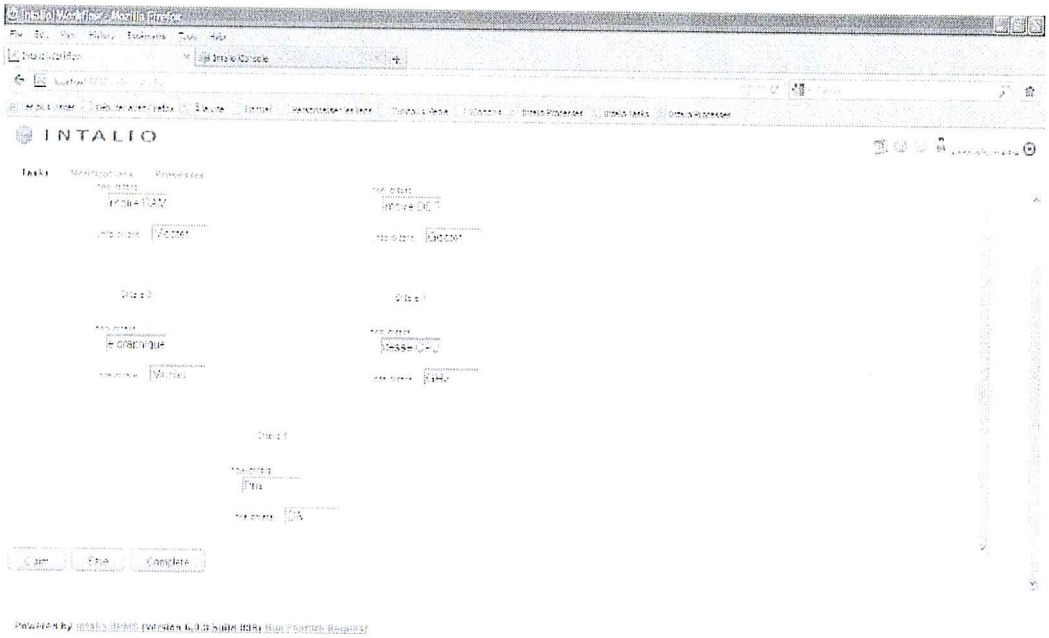
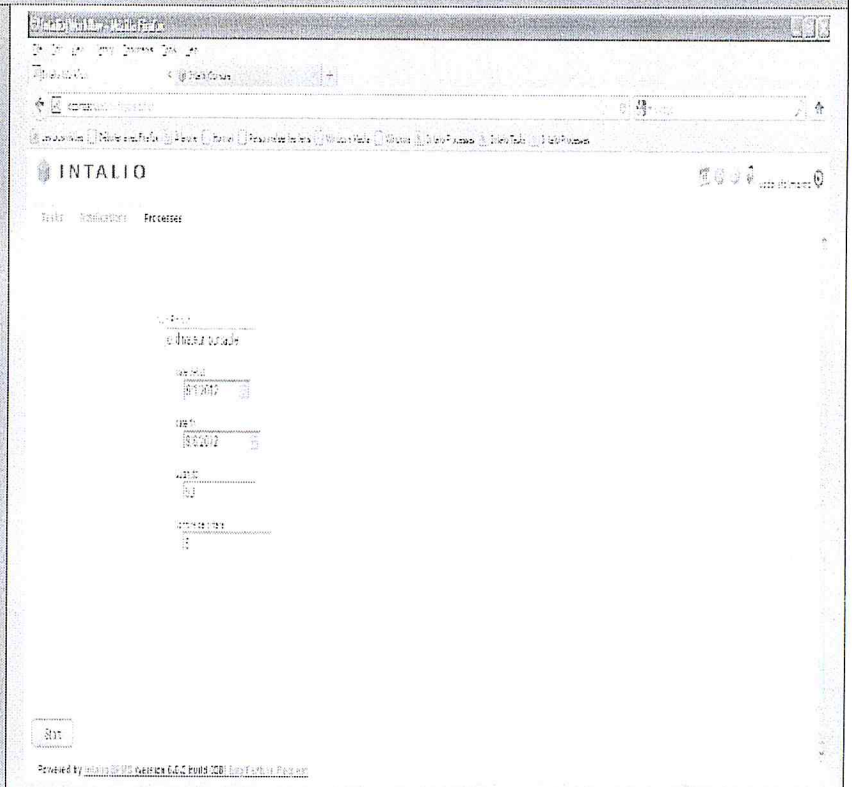
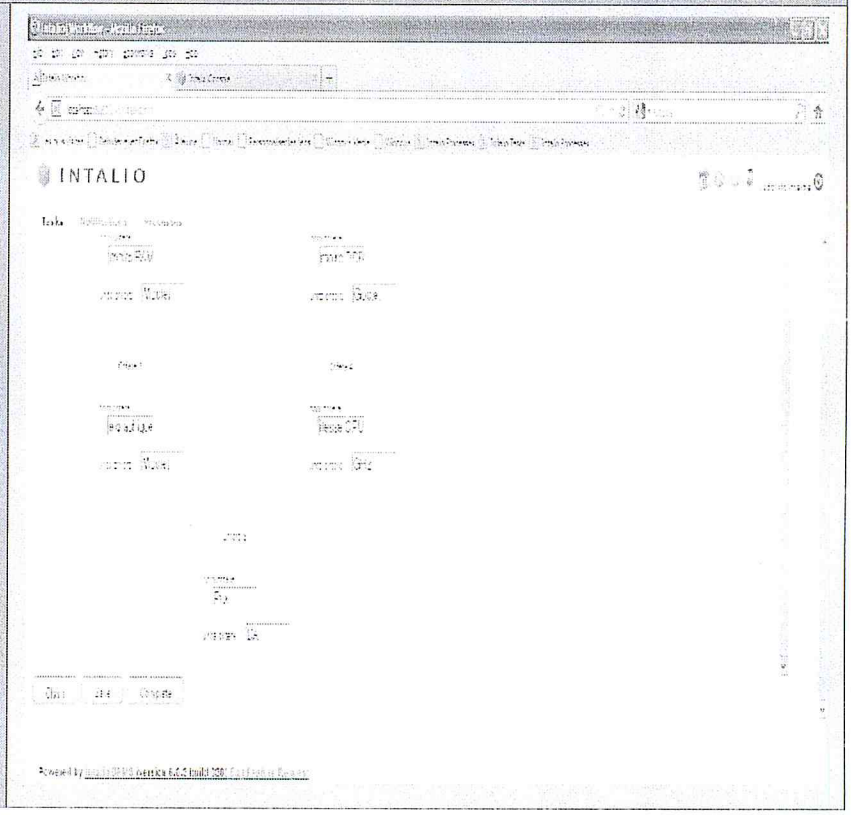
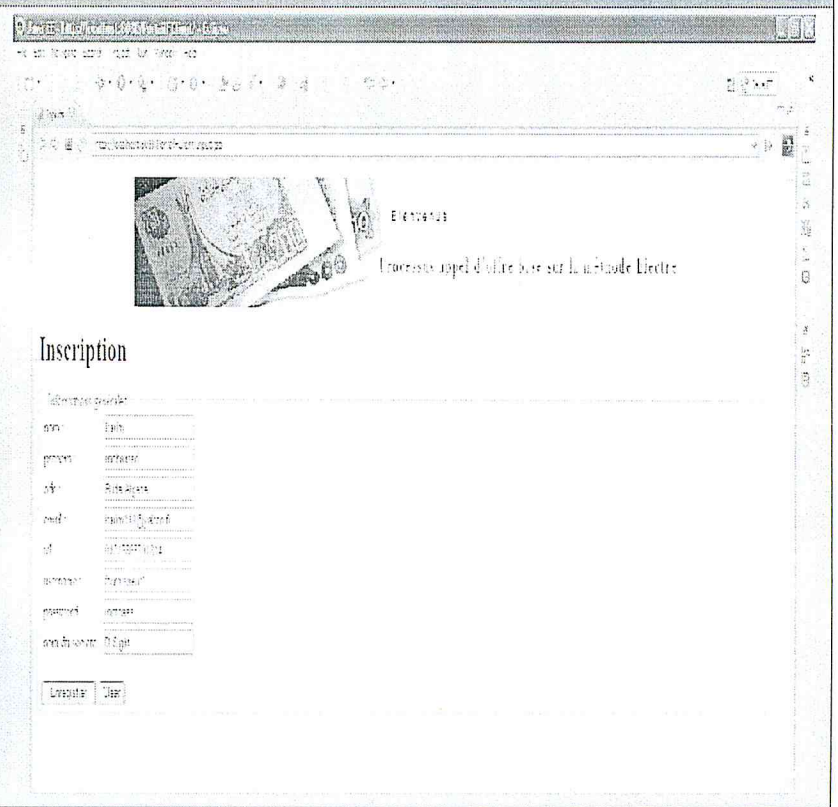
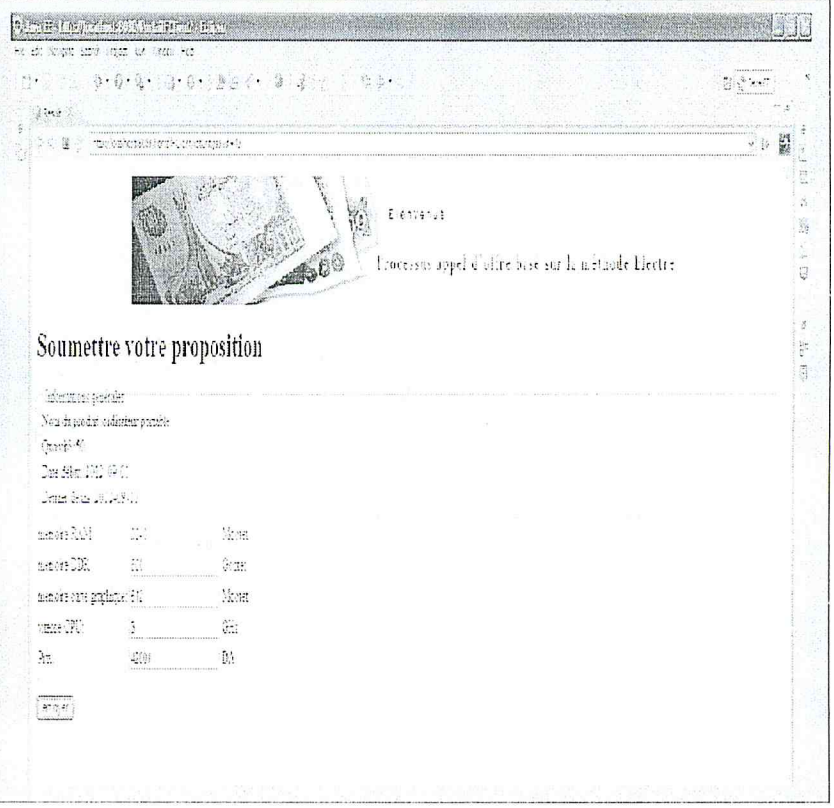


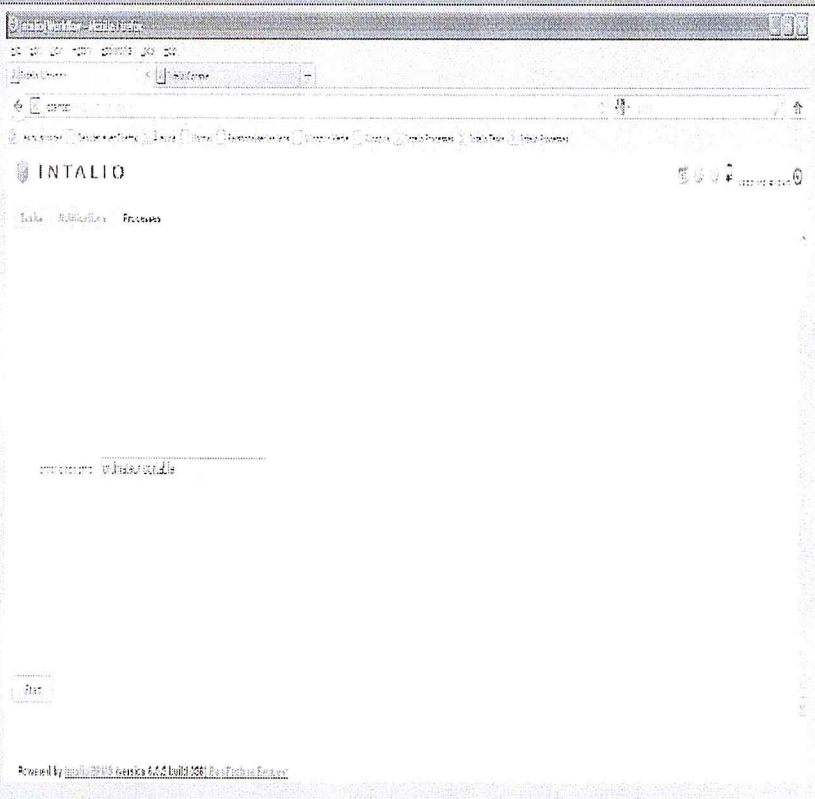
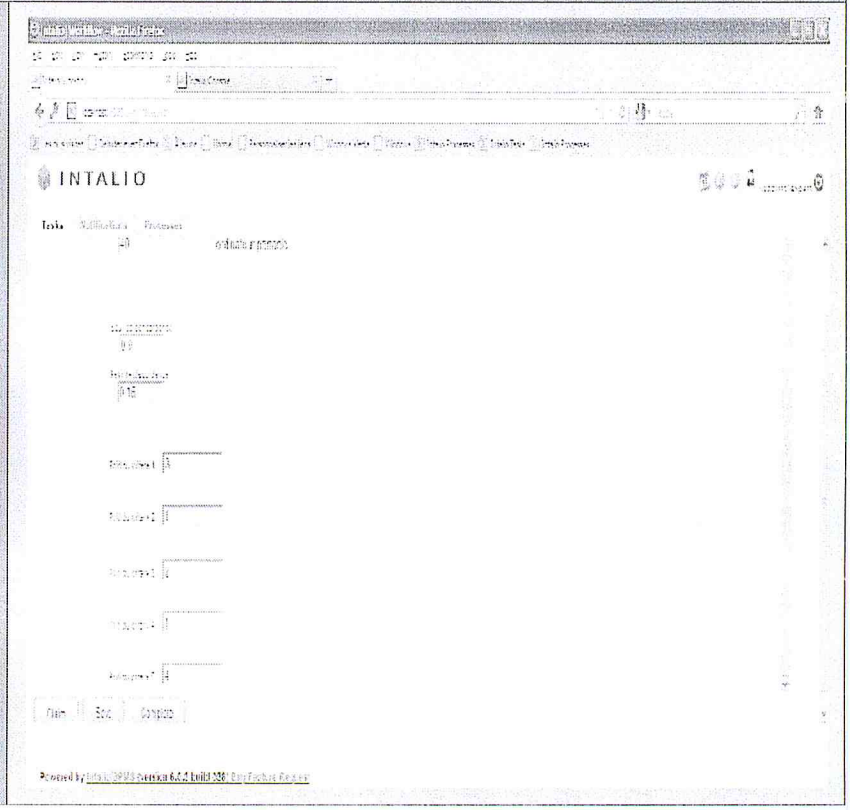
Figure 49 : étape 2 « créer les critères ».

X. Processus d'exécution de l'application

Le tableau suivant montre l'enchainement des étapes d'exécution de notre application suivant le processus métier, étape par étape, depuis la création d'un appel d'offre et jusqu'à la consultation des résultats.

Maitre d'ouvrage :	1. Créer un appel d'offre
Le maitre d'ouvrage accède au formulaire « créer un appel d'offre », il lance l'appel et après il complète la deuxième tache.	
Maitre d'ouvrage :	2. Définition des critères
Le maitre d'ouvrage accède au formulaire « définition des critères », et il complète le lancement de l'appel d'offre.	

Fournisseur :	3. Inscription des fournisseurs
Un fournisseur accède au portail fournisseur pour authentifier il faut faire l'inscription d'abord.	
Fournisseur :	4. Soumission
Apres l'authentification il accède au liste des offres pour soumettre leur proposition.	

Expert :	5. Choisir l'offre
L'expert accède au formulaire « choix le nom d'offre » pour choisir le nom d'offre qu'il s'applique la méthode Electre I.	
Expert :	6. Définition des poids et seuils
Apres le choix, l'expert va définir les poids de chaque critère et les seuils de concordance et de discordance.	

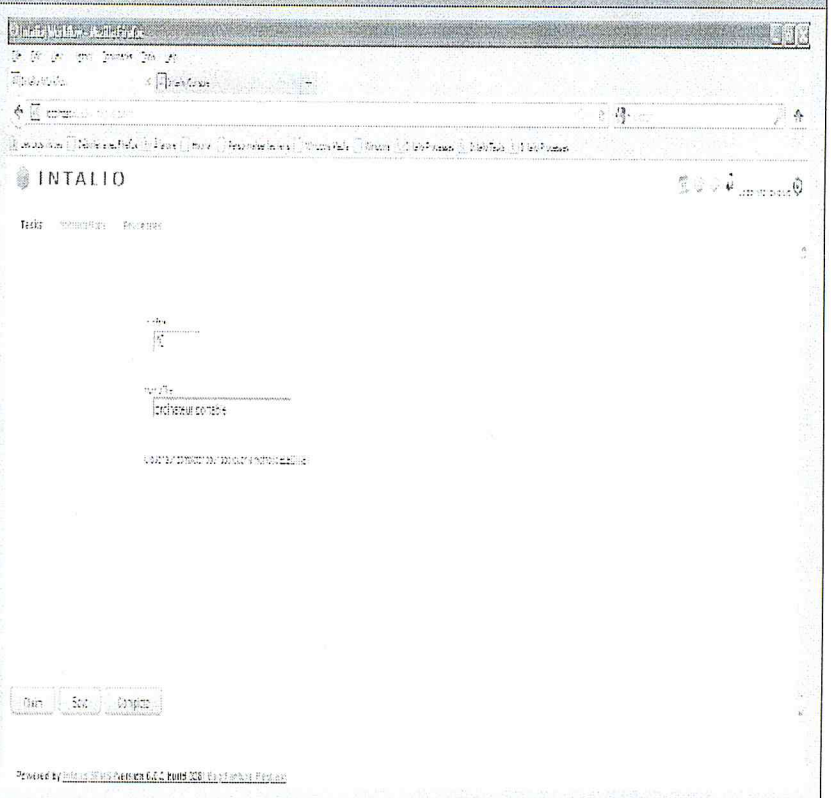
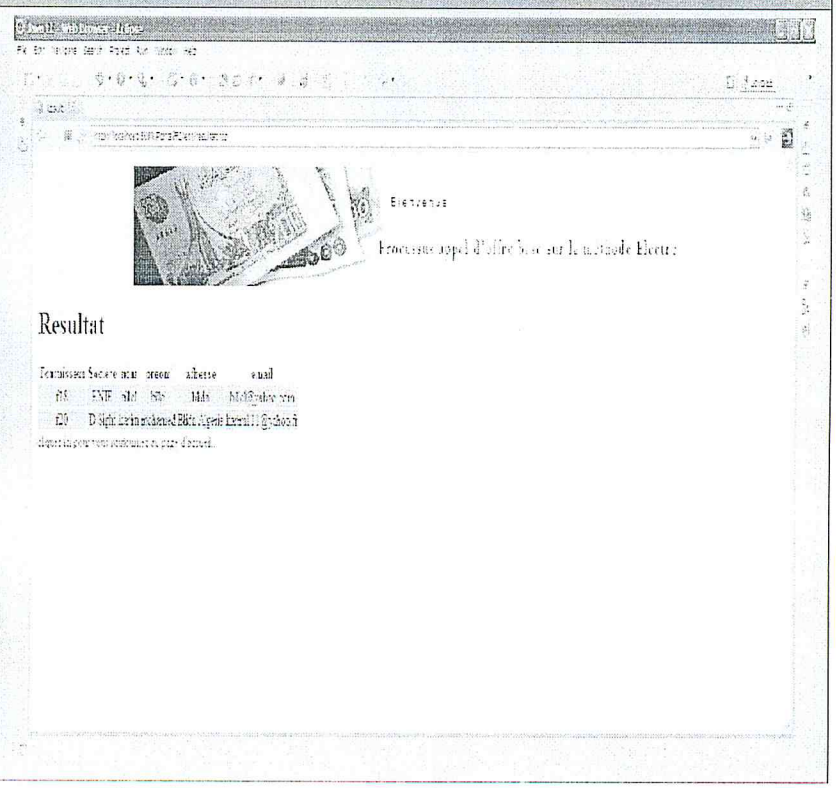
Expert :	7. Application de la méthode
L'expert complète la tache pour appliquer la méthode Electre 1.	
Fournisseur :	8. Consultation du résultat
Quand le délai de l'offre est terminé, il peut voir le résultat de cet offre « la liste des fournisseurs acceptée ».	

Tableau 7 : Processus d'execution de l'application.

La console d'administration des processus

La figure suivante illustre la console d'administration des instances des processus exécutées sur le serveur.

Figure 50 : Intalio console.

XI. Conclusion

L'exécution de ce processus informatisés reflète le déroulement du processus métier modélisé lors de la conception. On doit noter que les diagrammes BPD présentés dans ce chapitre ne représentent qu'une solution. En effet, la transposition d'un diagramme BPD en diagramme BPMN exécutable est réalisable suivant plusieurs directions tout en conservant les liens entre l'exécution, la modélisation et les objectifs à atteindre.

Conclusion générale

Le projet qui nous a été confié consistait à réfléchir à une architecture orientée service dédiée au processus métier RAO, qui doit prendre en charge l'aspect d'aide à la décision.

Pour cela, nous avons effectué un travail indispensable de recherche et d'analyse sur l'architecture orientée services, l'automatisation de processus métier et les différentes méthodes Electre comme Electre I, II, III…etc.

La présence d'un processus de développement formalisé, bien défini et bien géré est un facteur de réussite d'un projet. Pour cette raison nous avons suivi les préceptes de la méthode MDA.

Pour la conception de notre solution nous avons fait recourt à la méthodologie 2TUP appelé aussi cycle de développement Y lié au concept MDA. Le premier modèle CIM modèle exprime les besoins de l'utilisateur ; le deuxième modèle PIM décrit l'analyse et l'architecture du système pour identifier les services de notre système; le troisième modèle PDM décrire la plate-forme d'implémentation et le dernier PSM contient les étapes de réalisation de notre application.

Ce projet très enrichissant, nous a donné l'occasion de nous plonger dans un problème concret, où nous avons pu mettre à profit les connaissances acquises durant notre cursus, et avons acquis de nouvelles compétences, plus particulièrement dans les architectures des systèmes d'information.

Le début a été très difficile, ce qui est principalement dû à la multiplicité des concepts SOA, BPM, Service Web, orchestration…etc. et des notations BPMN et BPEL. Cependant, ces difficultés nous ont permis d'acquérir un esprit de synthèse, ce qui est loin d'être négligeable.

Le projet nous a de plus permis d'approfondir nos connaissances sur la programmation en JAVA, Axis et le langage SQL, ainsi que l'acquisition de connaissance sur des techniques récentes de modélisation et exécution de processus métiers (Intalio).

Néanmoins comme tout projet, le notre n'est pas exhaustif, il serait intéressant d'enrichir ce travail par :

- L'intégration des autres méthodes d'aide à la décision comme Prométhée,
- Compléter la programmation des autres méthodes d'Electre à savoir : II, III, TRI, IV, IS,

Bibliographie

Bibliographie :

- **[Amir NAFI , Caty WEREY.10]** :introduction aux méthodes d'analyse multicritère de type ELECTRE, 2010.
- **[Azaiez.S, 92]** : Approche dirigée par les modèles pour le développement de systèmes multi-agents, Thèse de doctorat, Polytech'Savoie - Université de Savoie, Mars 1992.
- **[Benguria *et al.* 06]** G. Benguria, X. Larrucea, B. Elveseater, T. Neple, A. Beardsmore, M. Friess. *A Platform Independent Model for Service Oriented Architectures.* 2éme conférence sur l'interopérabilité, IESA'06-Bordeaux, Mars 2006.
- **[Bernard Dubs,07]** : Bernard Dubs, Christophe Toulemonde, BIT Group, De l'éditeur de logiciel au partenaire Métier : L'offre SOA de Microsoft, (2007).
- **[Bézivin *et al.* 02]** J. Bézivin et X. Blanc, *MDA : standards et travaux*, développeur Référence, 2 octobre 2002.
- **[Bézivin *et al.* 03]** J. Bézivin, G. Dupé, F. Jouault, G.Pitette,J. E. Rougui, *First Experiments with the ATL Model Transformation Language: Transforming XSLT into Xquery*, 2nd OOPSLA Workhop on Generative Techniques in the context of Model Driven Architecture, October 2003.
- **[Bézivin *et al.* 03a]** J. Bézivin et S. Gérard, *A Preliminary Identification of MDA Components*, 2003.
- **[Garcia 04]** A. Garcia, *L'approche guidée par les modèles de l'OMG*, Tectosages, rapport interne, 2005.
- **[Gilbert Raymond, 07]**: Gilbert Raymond, SOA : Architecture Logique, Principes, structures et bonnes pratiques, Version 1.0, (2007).
- **[Jean Marie B,12]** : Jean Marie BOMY, consulté janvier 2012 ;
- **[Jeremy Fierstone,02]**: Jeremy Fierstone, les services web, (2002).
- **[Lopes 05]** D. Lopes, *Étude et applications de l'approche MDA pour des plate-formes de Service Web*, thèse de doctorat, Université de Nantes, 2005.
- **[M. Belaunde]** M. Belaunde - France Télécom R&D
- **[Maystre L. Yves,et al.,1996]** : Maystre L.Y., Pictet J., Simos J., 1996. Méthodes multicritères ELECTRE, Presses Polytechniques et Universitaires Romandes, Lausanne.
- **[Mickaël Baron,10]** : Mickaël Baron, Introduction SOA, (2010).
- **[Miller *et al.* 03]** J. Miller et J. Mukerji, *MDA Guide Version 1.1*, , Document number omg/,12 Juin 2003.
- **[Morley *et al.* 02]** C. Morley, J. Hugues et B. Leblanc, *UML pour l'analyse d'un système d'informations*, 2éme édition, Dunod, 2002.

- **[OMG03]** OMG, *MDA Guide Versions 1.0.1*,
- **[Patrice Briol,08]** :Patrice Briol, Ingénierie des processus métiers, de l'élaboration à l'exploitation, (2008).
- **[Pierre Bonnet,05]** : Pierre Bonnet, Cadre de Référence Architecture SOA Meilleures pratique, (2005).
- **[Roger et al., 2000]** :Rogers,M., Bruen,M. and Maystre,L-Y. 2000. ELECTRE and decision support, method and applications in engineering and infrastructure investment. Kluwer Academeic Publisher, ISBN 0-7923-8647-7, USA.
- **[ROY, 1985]** :Roy B., 1985, Méthodologie multicritère d'aide à la décision, Economica, Paris.
- **[Roy, 1996]** :Roy, B., 1996.Multicriteria Methodology for Decision Aiding Dordrecht : Kluwer Academic Publishers.
- **[Tanguy Crusson,08]** : Tanguy Crusson, Business Process Management de la modélisation à l'exécution, (2003).
- **[Terasse *et al.* 03]** M.N. Terrasse, M. Savonnet, G. Becker, and E. Leclercq. *UML-Based Metamodeling for Information System Engineering and Evolution*. In Proc. Of the 9th International Conference on Object-Oriented Information Systems, OOIS'03, LNCS 2817, pp. 83–94, 2003.
- **[Xavier Fournier-Morel,06]** : Xavier Fournier-Morel, Pascal Grojean, Guillaume Plouin, Cyril Rognon, SOA: Le guide de l'architecte, DUNOD, (2006).
- **[Yann Letanou,07]** :Yann Letanou, Urbanisation & Intégration de Systèmes «THINK SERVICE », Version 1.2, (2007).