

Université Libre de Bruxelles
Faculté des Sciences Appliquées
Département CoDE-SMG

Directeur de mémoire : Prof. Yves DE SMET
Superviseur : Ir Quantin Hayez

Analyse et conception d'un marketplace basé sur le logiciel D-Sight

Mémoire de fin d'études présenté par Xavier Aupaix en vue de l'obtention du grade
d'ingénieur civil informaticien à finalité Ingénierie informatique

Année académique 2009-2010

Remerciements

Je tiens à remercier ceux qui ont participé à l'aboutissement de ce mémoire :

Mon promoteur Yves De Smet pour sa motivation communicative, son aide et ses conseils promulgués au long de l'année, l'organisation du souper avec le service et surtout pour m'avoir permis de réaliser ce mémoire.

Quantin Hayez pour sa bonne humeur, sa disponibilité constante, son aide dans l'utilisation du logiciel D-Sight et évidemment pour son suivi permanent de mon avancement.

L'ensemble du service d'ingénierie de l'informatique et de la décision pour leurs questions et remarques judicieuses au cours des présentations faites pendant l'année.

Toutes les personnes ayant contribué de près ou de loin à l'élaboration de ce mémoire et en particulier à mes colocataires m'ayant supporté toute l'année et à Valentin Rouquette pour sa relecture et ses idées.

Résumé

Ce document décrit la conception d'une place de marché électronique basée sur la spécification Java Enterprise Edition et utilisant les méthodes PROMÉTHÉE-GAIA d'aide à la décision multicritère. Ces dernières seront intégrées à l'application via le logiciel D-Sight et utilisées afin de fournir une aide originale et nouvelle à l'utilisateur de ce genre de plate-forme.

Ce développement sera réalisé au travers de deux parties complémentaires. Dans la partie "analyse" nous étudierons l'aide à la décision multicritère et les différents modèles de plates-formes d'E-Business afin de mieux aborder la partie "conception". Dans cette dernière nous traiterons des applications web dynamiques, des langages permettant de les construire ainsi que des différents aspects techniques relatifs à ce genre de programme. Ensuite nous expliquerons le cheminement logique emprunté, partant de l'architecture du programme afin d'implémenter peu à peu les différentes classes tout en se basant sur les différents cas d'utilisation possibles.

Mots clés

J2EE, Servlet, JSP, JavaBean, marketplace, E-Business

Table des matières

Remerciements	2
Résumé	3
Table des matières	6
Table des figures	7
Liste des tableaux	8
Notations	9
1 Introduction	10
Contexte	10
Genèse du projet	10
Objectifs	11
Démarche	11
I Analyse	12
2 L'aide multicritere à la décision	13
2.1 L'aide à la décision et l'analyse multicritère	13
2.1.1 Introduction	13
2.1.2 Définitions	14
2.1.3 Problématiques de référence	18
2.1.4 Modélisation des préférences	20
2.1.5 Seuils de préférences	23
2.1.6 Base de toute méthodologie	24
2.2 Traitement des problèmes multicriteres	25
2.2.1 Introduction	25
2.2.2 Théorie de l'utilité multiattribut(MAUT)	25
2.2.3 Les méthodes interactives	26
2.2.4 Les méthodes de surclassement	27

2.3	Les méthodes PROMÉTHÉE-GAIA	29
2.3.1	Introduction	29
2.3.2	Enrichissement de la structure de préférence	29
2.3.3	Relation de surclassement évaluée et flux de surclassement	32
2.3.4	PROMÉTHÉE I	34
2.3.5	PROMÉTHÉE II	34
2.3.6	Le plan GAIA	35
3	Marketplace et E-Business	38
3.1	Contexte et définitions	38
3.1.1	E-Business	38
3.1.2	Marketplace et services proposés	40
3.2	Sécurité et aspects juridiques	43
II	Conception	44
4	Étude technique	45
4.1	Choix des solutions et description du langage choisi	45
4.1.1	PHP [Zer07]	46
4.1.2	Python [Hol08]	47
4.1.3	Ruby [Tho07]	47
4.1.4	J2EE [Keo02]	48
4.1.5	ASP .NET [Mac08]	48
4.1.6	Google Web Toolkit (GWT) [Jab09]	49
4.1.7	Choix final	50
4.2	Description de la spécification J2EE [All02] [Gir02]	51
4.2.1	Architectures Multi-tiers	51
4.2.2	Les servlets	52
4.2.3	Les Java Servlet Pages (JSP)	52
4.2.4	JavaBeans	53
4.2.5	Enterprise JavaBean (EJB) [MH02]	53
4.2.6	Java Standard Tag Library (JSTL)	54
4.2.7	Tomcat [Sek07]	54
4.3	Autres aspects techniques	56
4.3.1	Base de données [Zim08] [Sou02]	56
4.3.2	Sécurité [Pan04]	56

5	Développement et implémentation	60
5.1	Architectures	60
5.1.1	Conception de la base de données [Zim08]	61
5.1.2	Architecture du marketplace	62
5.2	Implémentation	65
5.2.1	Packages et exemple de fonctionnement	65
5.2.2	Librairies utilisées	68
6	Conclusions	70
	Bibliographie	73
	Annexes	75

Table des figures

2.1	Fonctions de préférence	32
2.2	Représentation graphique de la méthode PROMÉTHÉE II et du plan GAIA fournie par le logiciel D-Sight	37
3.1	Intermédiation assurée par une société tierce	41
4.1	Architecture d'une application web J2EE	52
4.2	Schéma récapitulatif du fonctionnement de Tomcat	55
4.3	Table "utilisateur" de la base de données contenant une entrée avec mot de passe haché avec la fonction SHA-256	58
5.1	Modèle entité-relation d'un cas simple avec les entités utilisateur et société	62
5.2	Exemple d'implémentation du pattern DAO via l'utilisation d'une classe générique	63
5.3	Exemple d'implémentation du pattern factory	64
5.4	Architecture finale de l'application web	65
6.1	Modèle entité-relation de la base de données	77
6.2	Page d'inscription du marketplace	78
6.3	Création d'un appel d'offre dans le marketplace	79
6.4	Création d'une offre dans le marketplace	80
6.5	Graphique résultant de la demande de rangement dans le marketplace	80

Liste des tableaux

1	Notations	9
2.1	Différentes actions et critères dans le problème du choix d'une carte graphique .	21
2.2	Modélisation des quatre situations fondamentales de préférences dans la comparaison de deux actions potentielles	22
2.3	Tableau d'évaluation pour n actions et k critères	25
4.1	Avantages et inconvénients des différents langages de programmation dynamiques	50

Notations

API	Application Programming Interface
ASP	Active Server Pages
B2B	Business to business
B2C	Business to Consumer
BLOB	Binary Large Object
DAO	Data Access Object
EJB	Enterprise JavaBean
GWT	Google Web Toolkit
IDE	Integrated Development Environment
J2EE	Java Enterprise Edition
JB	JavaBean
JSP	Java Server Pages
JSTL	JavaServer Pages Standard Tag Library
MD5	Message Digest 5
MVC	Model-View-Controller
PHP	Hypertext Preprocessor
POJO	Plain Old Java Object
RIA	rich Internet application
TIC	Technologies de l'information et de la communication
SGBDR	Système de gestion de base de données relationnelle
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer

TABLE 1 – Notations

Chapitre 1

Introduction

Contexte

Dans la société actuelle les transactions et échanges entre entreprises doivent être effectués de plus en plus rapidement. L'avènement d'internet a permis d'améliorer ces derniers en proposant un nombre important de solutions permettant de faire des affaires directement avec le consommateur ou d'autres sociétés. Un nouveau type d'infrastructure a même vu le jour récemment, permettant aux entreprises de trouver facilement des fournisseurs ou des acheteurs potentiels. Ces sites internet, appelés e-marketplace (Places de marché virtuelles en français), permettent de passer des appels d'offres et de faire monter les enchères afin de trouver l'offre la plus rentable possible, ou encore de se faire connaître facilement pour des petites entreprises.

Genèse du projet

Decision Sights, société spin-off de l'Université Libre de Bruxelles, a développé récemment un logiciel d'aide à la décision nommé D-Sight et basé sur la méthodologie PROMÉTHÉE-GAIA. Ce dernier permet de mettre en oeuvre facilement les méthodes de rangement PROMÉTHÉE et ce afin de guider les utilisateurs dans des processus de décision difficiles. Constatant que les places de marché étaient pour la plupart des plates-formes ne proposant pas d'aider l'utilisateur dans ses choix, l'idée d'utiliser D-Sight et les méthodes PROMÉTHÉE afin de combler ce manque commença à germer. L'innovation voulue par le département CoDE-SMG était de construire un modèle de place de marché qui, au delà des services habituellement proposés, utiliserait le logiciel D-Sight afin de guider l'acheteur dans son choix de fournisseur, lui permettant ainsi de choisir la meilleure alternative.

Objectifs

L'objectif 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 places de marché virtuelles afin de concevoir un prototype d'application web communiquant avec le logiciel D-Sight. Les objectifs seront donc de développer une analyse poussée du contexte afin de mieux construire l'application. Cette dernière devra permettre de se connecter en ayant un rôle : acheteur, fournisseur ou expert (permettant de fournir une note aux fournisseurs), chaque rôle pouvant interagir d'une façon différente avec l'application.

Démarche

Nous commencerons par étudier l'aide à la décision multicritère et plus particulièrement les méthodes PROMÉTHÉE-GAIA afin de pouvoir construire intelligemment la base de données qui sera utilisée dans l'application. Ensuite, nous dresserons un bilan des modèles d'E-Business actuels pour arriver à définir le rôle exact joué par les places de marché virtuelles ainsi que leurs contraintes, dans le but de concevoir l'application de la meilleure des manières.

Dans la partie conception nous commencerons par proposer une comparaison des différents langages web actuels et détaillerons celui choisi. Après nous passerons en revue les autres aspects techniques dont nous avons tenu compte dans le cadre du développement. Enfin nous expliquerons la conception de l'application web partant de l'architecture choisie pour arriver au prototype demandé.

Première partie

Analyse

Chapitre 2

L'aide multicritere à la décision

Introduction

Que se soit dans sa vie domestique ou professionnelle l'homme a et sera toujours amené à prendre des décisions. Ces dernières, bien qu'elles soient prises le plus souvent par une seule personne, sont le plus souvent la résultante d'une interaction entre les préférences des membres d'un groupe (Conseil d'administration, gouvernement, ...). Ces intervenants (ou acteurs) font des choix qui les amènent à exécuter ou non une action, la faire d'une manière ou d'une autre. Ces choix peuvent être sans importance ou, à l'opposé, influencer sur la vie d'un collectif (famille, société, pays, ...). Ces décisions, bien qu'elles soient la résultante d'une réflexion (ou processus de décision), sont également le résultat de la confrontation permanente entre ces différents acteurs et ne sont pas toujours les meilleures.

Nous verrons dans cette partie comment l'aide multicritère à la décision peut appuyer ces intervenants dans leur processus de décision. Nous détaillerons en particulier la méthodologie PROMÉTHÉE-GAIA qui sera utilisée via le logiciel D-Sight ¹ dans la suite de ce mémoire.

2.1 L'aide à la décision et l'analyse multicritère

2.1.1 Introduction

Malgré sa relative jeunesse, l'aide à la décision est devenu très vite un domaine primordial dans le contexte socio-économique. Une de ses nombreuses définitions peut être :

"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 un processus de décision, éléments

1. <http://www.decision-sights.com/>

concourant à éclairer la décision et normalement à prescrire (au sens médical), 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."[Roy85]

En d'autres terme l'aide à la décision permet, dans une situation de choix, de prendre la solution la plus avantageuse en se basant sur une modélisation des réalités humaines.

Dans cette partie, nous définirons les termes spécifiques à ce domaine de recherche en 2.1.2 afin de nous familiariser avec ce dernier. Ensuite, nous présenterons les différentes problématiques de références en 2.1.3 avant de déterminer laquelle sera traitée dans le cadre de ce mémoire. Après cela, nous introduirons en 2.1.4 les situations fondamentales de préférence via une mise en situation. En outre, en 2.1.5 nous parlerons des seuils de préférences, éléments cruciaux intervenant dans les méthodes de surclassement PROMÉTHÉE. Enfin, en 2.1.6 nous terminerons par dresser une liste d'étapes servant à toutes méthodologies.

2.1.2 Définitions

Modèle

La modélisation du problème de décision est la base de toute méthodologie d'aide à la décision. La définition la plus générale d'un modèle peut être :

"Un schéma qui, pour un champ de questions, est pris comme représentation d'une classe de phénomènes, plus ou moins habilement dégagés de leur contexte par un observateur pour servir de support à l'investigation et/ou à la communication."[Roy85]

Dans tous les domaines scientifiques des modélisations sont en effet imaginées afin de s'approcher au plus près de la réalité en la simplifiant sur certains points. Dans le cas de la recherche opérationnelle (et donc de l'aide à la décision) la modélisation a exactement le même but : transposer une réalité (ici humaine) en un modèle mathématique sur lequel le chercheur pourra se baser afin de développer une méthodologie. Ces réalités humaines peuvent être économiques, industrielles, sociologiques, etc.

Imaginons par exemple la problématique suivante : une société envisageant l'implantation d'une filiale dans un pays en voie de développement. Avant de s'engager tête baissée elle fera appel à un expert qui modélisera les différentes réalités humaines : est-ce que l'investissement

sera vite rentabilisé? (point de vue économique), Quelle production devons nous atteindre? (point de vue industriel), Quelle hiérarchie allons-nous instaurer? (point de vue sociologique)

Ces modèles sont appelés des approches qualitatives, en opposition aux approches quantitatives, se basant sur sa connaissance de la réalité, son intuition. Bien qu'elles soient différentes, ces deux approches ne sont pas incompatibles et sont parfois utilisées en complément.

Il faut cependant comprendre que ces réalités humaines sont complexes et leurs modélisations ne seront jamais parfaites. Les méthodologies d'aide à la décisions héritent directement de cette caractéristique et n'auront sans doute jamais valeur de vérité absolue. Cependant, le résultat de ces méthodologies est un atout important dans le processus de décision car il donne une idée générale quant aux meilleures solutions.

il sera évidemment nécessaire de nommer les personnes responsables et de leur donner un rôle dans le processus de décision afin de construire et d'exploiter ce modèle. Ces dernières se nommeront les intervenants.

Intervenants

Comme écrit dans l'introduction, un processus de décision résulte de l'interaction entre les préférences des membres d'un groupe. Ces derniers sont nommés les intervenants (ou acteurs) du processus de décision. Nous listerons ici les deux types d'intervenants importants :

- **Le décideur** : L'aide s'adresse à l'un des intervenants du processus de décision identifié comme étant le décideur. Ce dernier joue un rôle déterminant dans la conduite du processus, la décision s'exerçant en son nom. Cela ne signifie cependant pas que seules ses opinions, stratégies ou préférences soient à modéliser à l'exception des autres intervenants. Il évalue les choix possibles et leurs finalités, exprime ses préférences à l'homme d'étude et les fait prévaloir dans l'évolution du processus. Cependant, il se peut que le décideur ne soit pas une seule et unique personne mais soit un ensemble d'intervenants. ;
- **L'homme d'étude** : L'homme d'étude est le plus souvent un spécialiste (chercheur opérationnel, économiste, etc) distinct du décideur mais pas toujours du processus de décision. Son rôle est de concevoir le modèle, de l'explicitier, de trouver des éléments de réponses afin d'éclairer le décideur sur les conséquences de tel ou tel choix. ;

Il se peut également qu'une tierce personne fasse le lien entre le décideur et l'homme d'étude. Cette dernière est alors appelée demandeur.

Comme expliqué ci-dessus, le décideur doit évaluer les différents choix possibles. Dans un processus de décision et afin de montrer que ce choix aura une influence importante ces derniers sont appelés des actions.

Actions et actions potentielles

D'après [Roy85], une action 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”

En d’autres termes, une action est un objet, une décision, un candidat,... qui, dans un problème de décision, apporte un choix possible et ce le plus souvent en concurrence avec d’autres actions. Ces actions font parties du modèle. Reprenons par exemple d’implantation dans un pays en voie de développement. Imaginons maintenant que trois sites d’implantation soient accessibles afin de créer la filiale. Les différentes actions seront le site A, le site B ou le site C, chacun d’entre eux ayant des caractéristiques propres permettant de les comparer.

Une action potentielle peut se définir, toujours d’après [Roy85] comme :

”Une action réelle ou fictive provisoirement jugée réaliste par un acteur au moins ou présumée telle par l’homme d’étude en vue de l’aide à la décision ; L’ensemble des actions potentielles sur lequel l’aide à la décision prend appui au cours d’une phase d’étude est notée A”

L’ensemble A des actions potentielles peut être :

- **globalisé** : Chaque élément de A est exclusif de tout autre
- **fragmenté** : Les résultats du processus de décision font intervenir des combinaisons de plusieurs éléments de A

Cette caractéristique explique la partie de définition ci-dessus : ”le plus souvent en concurrence”. Dans le cadre de ce mémoire nous travaillerons avec des actions globalisées.

Si nous reprenons notre exemple nous pouvons facilement imaginer que le site B soit jugé irréaliste par un acteur à cause d’un dénivelé de terrain trop élevé. Ce site ne se retrouvera donc pas dans les actions potentielles. Par conséquent, l’ensemble A contiendrait le site A et le site C.

Afin de déterminer quel site sera le meilleur parmi les deux restant il faudra les comparer. Afin de réaliser cela le décideur se basera sur ses préférences propres dans le but de créer des échelles de jugement : les critères.

Critères

Aider à décider, c'est avant toutes choses, aider à clarifier une situation afin de faire comprendre au décideur quelles sont ses préférences. À ce niveau, le concept-clé est celui du critère. Selon [Roy93] :

"Pour l'essentiel, un critère vise à résumer, à l'aide d'une fonction, les évaluations d'une action sur diverses dimensions pouvant se rattacher à un même "axe de signification", ce dernier étant la traduction opérationnelle d'un "point de vue" au sens usuel du terme"

En d'autres termes, un critère est une fonction à valeurs réelles définie sur l'ensemble A des actions potentielles afin de pouvoir décrire le résultat de la comparaison de deux actions a et b et de fonder la proposition :

$$f_b \geq f_a \implies b S_f a$$

où S_f est une relation binaire signifiant "au moins aussi bon que", relativement au critère f .

Nous reviendrons sur ce point dans la section 2.1.4.

Dans un problème d'aide à la décision, la liste des critères que l'on prend en compte (la famille de critères) est étroitement liée au décideur et à son échelle des valeurs. En effet, selon qu'il porte une attention particulière ou nulle à un aspect de la question, ce dernier se retrouvera dans la famille de critère ou non, cela ouvrant sur un problème de subjectivité du choix des critères.

Cette subjectivité, bien qu'admise, ne signifie pas que la liste des critères puisse être remplie à la légère. Cette dernière doit avant toutes choses être exhaustive et l'homme d'étude à pour but d'éclairer le décideur dans son raisonnement. Une des méthodes courantes est de hiérarchiser ses critères en construisant un "arbre des critères" ou "value-tree" : en premier lieu on demande au décideur de définir les objectifs fondamentaux de la prise de décision. Ensuite on essaye de décomposer chaque objectif en critères.

Lorsque la famille de critères respectent certaines exigences on parlera alors de famille **cohérente** de critères. Ces dernières sont au nombre de trois :

- Exigence d'exhaustivité : les critères ne doivent pas être trop peu nombreux. Si on a la situation $f_j(a_i) = f_j(a_k) \quad \forall j \in F$ où F représente la famille de critères alors l'affirmation

" a_i est indifférente à a_k " doit être émise ; Dans le cas contraire cela signifierait que certains éléments d'appréciation n'ont pas été pris en compte dans la famille de critères.

- Exigence de cohérence : cohérence entre les préférences locales de chaque critère et les préférences globales : si l'évaluation de a_i est égale à celle de a_k sur tous les critères sauf un et qu'elle est meilleure sur cet unique critère, alors on peut émettre l'affirmation : " a_i est préféré à a_k "
- Exigence de non-redondance : il ne faut pas qu'il y ait des critères qui se multiplient et donc plus nombreux que nécessaire. Le nombre idéal doit être tel que la suppression d'un des critères laisserait une famille qui ne satisfait plus à une au moins de deux exigences précédentes.

Pour finir, nous verrons en 2.2.2 que ces critères peuvent être fusionnés en un critère unique menant à une analyse mono-critère du problème. Dans le cadre de notre mémoire nous nous intéresserons cependant plus à l'analyse multicritère, devenue populaire depuis le début des années 1970 et reflétant mieux les réalités humaine d'après [Vin89].

2.1.3 Problématiques de référence

Dans les problèmes multicritères plusieurs problématiques peuvent se poser. Ces dernières sont bien évidemment liées à la phase d'étude et à l'état d'avancement du processus de décision. Il est à noter que une des quatre problématiques citées ci-dessous ne s'impose pas forcément et l'homme d'étude peut se retrouver à devoir faire un choix parmi plusieurs d'entre elles. Ces quatre problématique sont chacune désignées par une lettre grecque.

Problématique du choix $P.\alpha$

- Objectif : aider à choisir une "meilleure" action ou à élaborer une procédure de sélection.

Cette problématique est la plus courante dans le domaine de l'aide à la décision. Le problème à résoudre est posé comme celui du "meilleur choix". Les procédures d'optimisation ont été créées pour ce type de problématique. L'investigation se tourne vers la mise en évidence d'un sous-ensemble A' de A , le plus restreint possible et ce dans un des buts suivants :

- Indiquer le plus précisément et rigoureusement possible au décideur une action à favoriser.

- Proposer la sélection d'une méthodologie comportant une procédure de sélection (d'une meilleure action) en accord avec une éventuelle utilisation répétitive et/ou automatisée.

En fait cette méthodologie consiste surtout à justifier le "non choix" du plus grand nombre d'action, l'idéal étant de réduire l'ensemble A à une unique action. Cependant l'obtention d'une telle action optimale dans les problèmes de décisions multicritères est le plus souvent irréaliste étant donné le caractère antagoniste des critères.

exemple : Le choix d'un site pour l'implémentation d'un commerce.

Problématique du choix $P.\beta$

- Objectif : aider à trier les actions d'après des normes ou à élaborer une procédure d'affectation.

Cette méthodologie est utilisée dans le but de classer les différentes actions potentielles dans des catégories définies à l'avance, ces dernières orientant les actions qu'elles sont destinées à recevoir.

exemple : L'admission d'un candidat pour un poste. Les catégories pourraient être "dossiers préconisant l'admission du candidat sans entretien", "dossiers préconisant un entretien avec le candidat avant de décider", "dossiers préconisant le refus direct du candidat".

Problématique du choix $P.\gamma$

- Objectif : aider à ranger les actions selon un ordre de préférence décroissante ou à élaborer une procédure de classement.

L'investigation s'oriente ici vers la mise en évidence d'un classement défini sur un sous-ensemble A' de A. Ce classement se destine à distinguer entre-elles les actions se trouvant être "suffisamment satisfaisante" en fonction d'un modèle de préférence. Ce type de rangement étant destiné à aider le décideur et devant être le reflet d'une priorité plus ou moins grande que ce dernier attache à chaque action potentielle de A' . Ce rangement doit aider à guider la réflexion du décideur, à orienter sa discussion avec les autres intervenants.

exemple : Le remplissage d'une grille de programmes pour une chaîne télé.

Problématique du choix P. δ

- Objectif : aider à décrire les action et/ou leurs conséquence de façon systématique et formalisée ou à élaborer une procédure cognitive.

Dans ce cas il s'agit d'avantage pour l'homme d'étude de formuler un problème que de le résoudre. En effet l'investigation sert essentiellement à mettre en valeur des informations relatives aux actions potentielles. Cela doit aider le décideur à les comprendre, à les estimer en lui présentant une description des actions et de leurs conséquences tant qualitatives que quantitatives.

Dans le cadre de notre mémoire le type de problématique traité sera plutôt celui du rangement p. γ . En effet la personne passant l'appel d'offre devra indiquer ces critères afin de pouvoir ranger les offres et d'avoir une idée de la valeur de chacune d'elles. Ce genre de problématique sera solutionnée par la méthode PROMÉTHÉE II vue en 2.3.5 et permettant de réaliser un rangement complet des différentes actions.

Afin de solutionner ces problématiques une structure de préférence doit être construite afin de pouvoir comparer les différentes actions et ainsi commencer à trouver une solution au problème. Dans la prochaine section nous introduirons les 4 situations fondamentales de préférence permettant de comparer deux actions entre-elles.

2.1.4 Modélisation des préférences

Considérons une entreprise spécialisée dans l'infographie et désirant renouveler les cartes graphiques de leurs postes de travail. Pour cela le comptable de l'entreprise débloque un budget moyen B par carte graphique. Il est imaginable qu'afin de trouver un fournisseur le chargé des achats passe un appel d'offre sur un marketplace et reçoive 5 réponses (cet exemple est repris dans le cadre d'une démonstration de l'application et se trouve en annexe). Le tableau 2.1 montre les différentes offres reçues.

Étant donnée les valeurs propres à l'entreprise celle-ci affiche une préférence marquée de a_1 et a_2 par rapport à a_3 . On exprimera cette préférence en disant que a_1 (ou a_2) est **strictement préféré** à a_3 .

La comparaison entre a_1 et a_2 est quant à elle plus complexe. Étant donné que le critère de prix est celui sur lequel la société pose le plus de valeur et qu'elle considère que la mémoire de a_1 est suffisante on dira alors que a_1 est **légèrement préféré** à a_2 . Nous entendons par là que

Modèle de carte	Mémoire	Fréquence	unités de traitement	Prix d'achat
a_1	1 Go	725 MHz	3000	0,87 B
a_2	2 Go	650 MHz	3100	0,95 B
a_3	500 Mo	650 MHz	3100	0,99 B
a_4	2 Go	750MHz	3000	Vraisemblablement Supérieur à B

TABLE 2.1 – Différentes actions et critères dans le problème du choix d'une carte graphique

la société hésite entre a_1 strictement préféré à a_2 et a_1 **indifférent** de a_2 .

Enfin la carte a_4 dont on ne connaît pas exactement le prix pourrait être vendue à 0,9 B, auquel cas la société la préférerait strictement à a_1 . Si, au contraire le prix est de 1,2 B ou supérieur la préférence serait inversée. La valeur du prix de vente qui assurerait l'indifférence entre a_1 et a_4 lui paraît difficile à préciser. Nous parlerons ici de **situation d'incomparabilité**.

Le tableau 2.2 donne la définition et les propriétés de ces situations [Roy85].

Une relation R sur le domaine E est dite symétrique si et seulement si :

$$\forall (x, y) \in E^2, (xRy) \iff (yRx)$$

Une relation R sur le domaine E est réflexive si et seulement si tout élément de E est en relation avec lui-même :

$$\forall (x) \in E, (xRx)$$

La relation d'indifférence I étant symétrique et réflexive on pourra donc écrire, aIa , aIb et bIa sans aucun problème(sous condition que les actions a et b soient indifférentes pour les deux dernières relations). D'après [Vin89] :

"Les trois relations P,I,R constituent une structure de préférence sur A si elles ont les propriétés indiquées dans le tableau ci-dessus et si, étant donné deux éléments quelconque a et b de A, une et une seule des situations suivantes est vérifiée : aPb , bPa , aIb , aRb ".

Mathématiquement un problème à k critères devant être maximisés peut donc être ramené à une relation de dominance sur A :

Situation	Définition	Relation binaire(propriétés)
Indifférence	Cette situation existe à partir du moment où les critères et le jugement de valeur du décideur justifient une équivalence entre les deux actions.	I : relation symétrique réflexive
Préférence stricte	Cette situation existe à partir du moment où les critères et le jugement de valeur du décideur justifient une préférence significative de l'une des deux actions.	P : relation asymétrique irréflexive
Préférence faible	Cette situation existe à partir du moment où les critères et le jugement de valeur du décideur ne justifient pas une préférence stricte de l'une des deux actions sans pour autant en déduire une préférence stricte en faveur de l'autre action ou une indifférence entre ces deux actions.	Q :relation asymétrique irréflexive
Incomparabilité	Cette situation existe à partir du moment où les critères et le jugement de valeur du décideur ne justifient aucunes des trois situations précédemment citées.	R : relation symétrique irréflexive

TABLE 2.2 – Modélisation des quatre situations fondamentales de préférences dans la comparaison de deux actions potentielles

$$\forall a, b, \in A : \begin{cases} aP_D b \iff \begin{cases} f_j(a) \geq f_j(b), \forall j = 1, \dots, k \\ \exists h : f_h(a) > f_h(b) \end{cases} \\ aI_D b \iff f_j(a) = f_j(b), \forall j = 1, \dots, k \\ aR_D b \iff \begin{cases} \exists h : f_h(a) > f_h(b) \\ \exists h' : f_{h'}(a) < f_{h'}(b) \end{cases} \end{cases}$$

Cette dernière relation n'est qu'une façon de voir le problème, basée sur une structure de préférence déclarant la préférence stricte d'une action a par rapport à une action b lorsque a se distingue sur au moins un critère tout en étant au même niveau que b sur les autres. Afin de

construire d'autres structures de préférences nous pouvons introduire les seuils de préférence.

2.1.5 Seuils de préférences

Le modèle le plus classique permettant de séparer deux actions en fonction d'un critère revient à supposer que :

$$\forall a, b \in A : \begin{cases} aP_fb \iff f(a) > f(b) \\ aI_fb \iff f(a) = f(b) \end{cases}$$

On parle alors de "vrai-critère" étant donné qu'il permet à la structure de préférence (I_f, P_f) d'être un préordre complet (structure de classement avec possibilité d'ex-æquo). Dans ce modèle, toute différence, aussi faible soit-elle, est révélatrice d'une situation de préférence stricte. De plus nous remarquerons que la préférence et l'indifférence sont transitives. En effet une relation R définie sur un ensemble E est transitive si et seulement si :

$$\forall x, y, z \in E [xRy \wedge yRz \implies xRz]$$

Nous constatons que si aPb et bPc alors aPc . Du point de vue de l'indifférence cette propriété pose problème. En effet si nous reprenons l'exemple de la tasse de thé cité dans [Vin89] nous comprenons pourquoi : imaginons n tasses de thé T_i dont le taux de sucre est progressivement augmenté d'un milligramme. Si une personne goute les tasses les unes après les autres elle en conclura qu'elles sont identiques. Cependant en comparant la première tasse et la dernière elle se rendra compte que la différence en sucre existe. Cet exemple contredit la transitivité de l'indifférence. Il convient alors de prendre en compte les petits écarts en introduisant un seuil d'indifférence. Lorsque $f_a \geq f_b$:

$$\begin{cases} aP_fb \iff f(a) - f(b) > q(f(b)) \\ aI_fb \iff f(a) - f(b) \leq q(f(b)) \end{cases}$$

où $q(f(.))$ est un seuil dit d'indifférence représentant le plus grand écart $f_a - f_b$ compatible avec une situation d'indifférence entre a et b . On parle alors de "quasi-critère" et la structure de préférence (I_f, P_f) est alors appelée un quasi-ordre.

Dans ce modèle tout écart légèrement supérieur au seuil d'indifférence est révélateur d'une situation de préférence stricte. Cela est parfois considéré comme une transition fort brusque. Pour éviter cela nous introduirons encore le concept de "pseudo-critère" : lorsque $f_a \geq f_b$ on a :

$$\begin{cases} aP_fb \iff f(a) - f(b) > p(f(b)) \\ aQ_fb \iff q(f(b)) < f(a) - f(b) \leq p(f(b)) \\ aI_fb \iff f(a) - f(b) \leq q(f(b)) \end{cases}$$

Ce concept introduisant le seuil de préférence $p(f(.))$ en plus du seuil d'indifférence $q(f(.))$, permettant une transition plus douce en passant par la préférence faible. La structure de préférence (I_f, Q_f, P_f) est alors un pseudo-ordre.

Ces seuils de préférences sont énormément utilisés dans les méthodes de surclassement PROMÉTHÉE vues au chapitre 2.3

2.1.6 Base de toute méthodologie

Nous avons maintenant toutes les informations nécessaires afin de dresser les bases de toute méthodologie d'aide à la décision. Les étapes sont les suivantes :

Dresser la liste des actions potentielles

Nous avons vu la définition d'une action potentielle en 2.1.2. La première étapes dans toutes méthodologie d'aide à la décision est de déterminer ces actions potentielles afin de construire l'ensemble A. Cette étape est cruciale car l'identification des actions potentielles est parfois l'une des étapes les plus difficiles dans un processus d'aide à la décision.

Dresser la liste des critères à prendre en considération

Cette deuxième étape revient à identifier les critères important comme détaillé dans la section 2.1.2.

Établir le tableau des performances

Le tableau des performances est une construction prenant comme lignes les n actions potentielles de A et en colonnes les k critères établis à l'étape précédente. La valeur introduite dans une case (i,j) n'est autre que l'évaluation du critère j dans le cas de l'action j notée $f_j(a_i)$. Le tableau 2.3 comprend donc $n \times k$ évaluations :

Nous pouvons nous apercevoir sur ce tableau que chaque action potentielle de A peut se voir comme un vecteur k dimensions dans l'espace des critères.

	$f_1(.)$	$f_2(.)$...	$f_j(.)$...	$f_k(.)$
a_1	$f_1(a_1)$	$f_2(a_1)$...	$f_j(a_1)$...	$f_k(a_1)$
a_2	$f_1(a_2)$	$f_2(a_2)$...	$f_j(a_2)$...	$f_k(a_2)$
...
a_i	$f_1(a_i)$	$f_2(a_i)$...	$f_j(a_i)$...	$f_k(a_i)$
...
a_n	$f_1(a_n)$	$f_2(a_n)$...	$f_j(a_n)$...	$f_k(a_n)$

TABLE 2.3 – Tableau d'évaluation pour n actions et k critères

Établir un modèle de préférences globales

En dernier lieu il faut créer une représentation formalisée de préférences globales relativement à l'ensemble A des actions potentielles. Par préférences globales il faut entendre les préférences qui mettent en jeu la totalité des conséquences à prendre en compte en vue de l'aide à la décision. C'est sur cette dernière étape que grands nombres de méthodes d'aide à la décision multicritère diffèrent.

2.2 Traitement des problèmes multicriteres

2.2.1 Introduction

Dans cette partie nous évoquerons la plupart des principales méthodologies existantes à ce jour. Bien que ces méthodologies soient très différentes les unes par rapport aux autres elles ont cependant un même but : aborder au mieux un problème de décision pour arriver à une modélisation réaliste de la réalité comme décrit en 2.1.2. Nous verrons que chaque modélisation a ses avantages comme ses inconvénients. En 2.2.4 nous parlerons des relations de surclassement dont font partie les méthodes PROMÉTHÉE abordées dans le chapitre suivant.

2.2.2 Théorie de l'utilité multiattribut(MAUT)

Cette théorie d'inspiration anglo-saxonne est couramment utilisée aux États-Unis et ce dans de nombreux domaines. Elle repose sur la substitution du problème multicritère

$$\text{Max } \{ f_1(x), f_2(x), \dots, f_j(x), \dots, f_k(x) \mid x \in A \}$$

en une fonction U

$$U = U(f_1(x), f_2(x), \dots, f_k(x))$$

modélisant les préférences du décideur. Il est à noter que dans notre exemple nous avons considéré que tous les critères étaient à maximiser dans le but de simplifier l'écriture.

Dans cette méthode deux problématiques principales sont à considérer :

- La fonction U devant représenter le mieux possible les préférences du décideur il faut que ces dernières aient des propriétés spécifiques. Un des buts actuels des chercheurs est de définir ces propriétés.
- Ces fonctions devant être définies de manière spécifique l'étude de leur construction et l'estimation des paramètres intervenant dans celles-ci doivent également être étudiées.

Il existe de nombreux modèles construits autour de cette théorie dépendant surtout de la forme analytique de la fonction U (additive, multiplicative, mixte, ...). Cette méthodologie, proposant de transformer un problème multicritère et ses solutions incomparables en un seul problème uni-critère ayant quant à lui des solutions comparables a séduit beaucoup de personnes. Ainsi de nombreuses méthodes de construction de fonctions d'utilité ont vu le jour. Bien que cette théorie soit basée sur un corps d'axiomes et d'hypothèses extrêmement précis [Mar02] émet des réserves quant à la qualité de celle-ci :

”Substituer la fonction d'utilité au problème multicritère de base en affirmant traiter du même problème est à la limite de l'honnêteté scientifique.”

Le modèle additif

Citée précédemment, la forme analytique la plus simple est évidemment la forme additive :

$$U(a) = \sum_{j=1}^n U_j(f_j(a))$$

Les fonctions U_j ayant pour simple utilité de transformer les critères initiaux de manière à uniformiser leurs échelles.

2.2.3 Les méthodes interactives

Dans sa définition simple une méthode interactive est un cycle de phases de calculs et de phases de dialogues entre l'homme d'étude et le décideur. Bien que le dialogue entre les deux soit essentiel dans toutes méthodes d'aide à la décision celui-ci est un outil principal dans une méthode interactive : après chaque phase de calcul le décideur réagit à la solution en exprimant ses préférences tout en essayant de faire des concessions sur certains critères. Bien que

de nombreuses méthodes interactives comme la méthode STEM, la méthode de GEOFFRION, DYER et FEINBERG ou encore la méthode du point de mire soient citées dans la littérature et détaillées([Vin89]) nous ne les décrivons pas en détail mais expliquerons comment la méthode se déroule en générale. Celle-ci commence généralement par une phase de calcul dans laquelle l'homme d'étude détermine une première solution de compromis basée sur les préférences du décideur. Une fois celle-ci trouvée le décideur la juge dans une phase de décision. A partir de ce point plusieurs cas de figure peuvent se produire :

- Il est satisfait de la solution obtenue et la méthode s'arrête immédiatement.
- Il veut améliorer tous les critères. Ceci est évidemment impossible si l'homme d'étude a bien mené sa méthodologie et la méthode s'arrête.
- Il identifie un critère f_g sur lequel il est prêt à faire une concessions Δ_g . L'homme d'étude peut alors utiliser cette nouvelle information dans une nouvelle phase de calcul.

Ce cycle recommence de cette manière jusqu'au moment où la solution est rejetée ou acceptée par le décideur. Cette méthodologie, bien qu'efficace dans son ensemble est assez longue à mettre en place étant donné le caractère cyclique de sa définition. De plus le décideur étant plus important et plus fréquemment sollicité dans cette méthode il faut qu'il soit d'accord pour intervenir à chaque étape.

2.2.4 Les méthodes de surclassement

Le leitmotiv de base des méthodes de surclassement était de trouver un juste milieu entre la relation de dominance, trop pauvre dans sa définition pour être d'une utilité quelconque, et la fonction d'utilité multiattribut, trop compliquée dans sa conception et surtout trop riche dans ses relations (un préordre complet formé où toutes les actions sont comparables). Cette méthode se déroule en deux phase :

- La construction d'une relation de surclassement
- L'exploitation de cette relation

La plupart des méthodes de surclassement utilisent la notion de "poids" des critères afin de refléter l'importance de ces derniers. La détermination de ces poids fait également l'objet

d'études afin d'aider le décideur dans cette démarche. Nous ne décrivons pas ces méthode dans la suite de ce mémoire.

La méthode ELECTRE I [Vin89] [May94]

Cette méthodologie vise à obtenir un sous-ensemble N d'actions tel que toute action externe à N soit surclassée par au moins une action de N et que les actions appartenant à N soient incomparables entre elles. Ce sous-ensemble est souvent comparé au noyau d'un graphe de surclassement possédant les actions comme sommets, ces derniers étant reliés entre eux par une flèche partant de l'action surclassant et arrivant vers l'action surclassée. On peut, à partir de cette première description, comprendre la philosophie générale des méthodes de surclassement. N n'est sans doute pas l'ensemble des bonnes actions mais l'ensemble dans lequel se trouve certainement le meilleur compromis. La méthode ELECTRE ("Élimination Et Choix Traduisant la Réalité") fût la première méthode de surclassement proposée dans la littérature et à depuis été maintes et maintes fois utilisée. Elle sert généralement dans les problèmes de choix. Nous ne décrivons pas ici la méthodologie complète mais la manière dont est construite la méthode de surclassement. L'homme d'étude et le décideur attribuent à chaque critère un poids p_j reflétant l'importance de celui-ci. Une relation de concordance est ensuite construite pour chaque couple (a,b) d'actions :

$$c(a,b) = \frac{1}{P} \sum_{j=f_j(a) \geq f_j(b)} p_j, \text{ où } P = \sum_{j=1}^n p_j$$

$c(a,b)$, variant de 0 à 1 est, d'une certaine manière, une mesure des arguments favorisant l'affirmation : "a surclasse b". Un autre indice, dit de "discordance", a également été introduit afin de vérifier si un des critères favorisait tellement b que l'indice de concordance pouvait être remis en cause.

$$d(a,b) = \begin{cases} 0 & \text{si } f_j(a) \geq f_j(b), \forall j \\ \frac{\max_j [f_j(b) - f_j(a)]}{\max_{c,d,j} [f_j(c) - f_j(d)]} & \text{sinon} \end{cases}$$

Cet indice est d'autant plus proche de 1 que la préférence de b sur a est forte sur au moins un critère. Cependant cette définition est valable uniquement si les différences $f_j(b) - f_j(a)$ ont un sens et sont comparables d'un critère à l'autre. Sinon on définira, pour chaque critère j , un ensemble de discordance D_j formé des couples (x_j, y_j) tels que si $f_j(a) = x_j$ et $f_j(b) = y_j$, alors on refuse le surclassement de b par a. Ayant défini un seuil de concordance \hat{c} et, si nécessaire, un seuil de discordance \hat{d} on peut définir la relation de surclassement S :

$$aSb \text{ ssi } \begin{cases} c(a, b) \geq \hat{c} \\ d(a, b) \leq \hat{d} \end{cases}$$

ou

$$aSb \text{ ssi } \begin{cases} c(a, b) \geq \hat{c} \\ (f_j(a), f_j(b)) \notin D_j, \forall j \end{cases}$$

A partir de cette relation de surclassement S l'homme d'étude peut alors construire le graphe de surclassement cité plus haut. Afin de trouver le noyau du graphe il essaiera donc de trouver un sous-ensemble de N actions tel que :

$$d(a,b)= \begin{cases} \forall b \in A \setminus N, \quad \exists a \in N : aSb, \\ \forall a, b \in N, \quad a \not S b \end{cases}$$

Des algorithmes existent afin de trouver le noyau du graphe plus facilement. A partir du noyau il est alors possible de trouver une solution acceptable au problème de décision en effectuant une analyse plus fine des actions du noyau.

2.3 Les méthodes PROMÉTHÉE-GAIA

2.3.1 Introduction

Les méthodes PROMÉTHÉE-GAIA, introduites pour la première fois au début des années 80, se proposent d'agir en trois temps. Premièrement un enrichissement de la structure de préférence permet d'éliminer les problèmes d'échelle liés aux unités des critères en définissant des fonctions de préférences intégrant les seuils de préférences vus en 2.1.5. Dans un deuxième temps elle propose d'enrichir la relation de dominance vue en fin de section 2.1.4 afin que celle-ci prenne en compte la structure de préférence nouvellement construite. Enfin la relation de surclassement ainsi obtenue est utilisée afin d'aider le décideur.

2.3.2 Enrichissement de la structure de préférence

Partons de la relation :

$$d_j(a, b) = f_j(a) - f_j(b)$$

exprimant l'écart entre les évaluations des actions a et b pour le critère j . Pour l'enrichir nous introduisons une fonction normée

$$P_j(a, b) = P_j[d_j(a, b)]$$

pouvant prendre les valeurs suivantes

$$\left\{ \begin{array}{ll} P_j(a, b) = 0 \text{ si } d_j(a, b) \leq 0 & (\text{pas de préférences}) \\ P_j(a, b) \approx 0 \text{ si } d_j(a, b) > 0 & (\text{préférence faible}) \\ P_j(a, b) \approx 1 \text{ si } d_j(a, b) \gg 0 & (\text{préférence forte}) \\ P_j(a, b) = 1 \text{ si } d_j(a, b) \ggg 0 & (\text{préférence stricte}) \end{array} \right.$$

Il est clair que si $d_j(a, b) \leq 0$ alors $P_j(a, b) = 0$ mais $P_j(a, b)$ peut être positive. Pour cette raison nous introduisons la fonction $H_j(d_j)$ telle que :

$$H_j(d_j) = \begin{cases} P_j(a, b) & \text{si } d_j(a, b) \geq 0 \\ P_j(b, a) & \text{si } d_j(a, b) \leq 0 \end{cases}$$

Cette fonction varie également entre 0 et 1. Cependant selon les seuils voulus 2.1.5 pour un critère le décideur pourra choisir son type de fonction :

- Critère usuel : Ce type, évoqué sous le nom de "vrai critère" en 2.1.5, est modélisé par la fonction :

$$H_j(d_j) = \begin{cases} 0 & \text{si } d_j = 0 \\ 1 & \text{si } d_j \neq 0 \end{cases}$$

Il n'y a donc indifférence que si $f_j(a) = f_j(b)$. Cette fonction permet de faire usage de la notion de critère dans son commun.

- Quasi-critère ou critère en U :

$$H_j(d_j) = \begin{cases} 0 & \text{si } |d_j| \leq q_j \\ 1 & \text{si } |d_j| > q_j \end{cases}$$

Les actions a et b sont indifférentes aussi longtemps que l'écart $d_j(a, b)$ ne dépasse pas un seuil q_j

- Critère à préférence linéaire (ou critère en V) :

$$H_j(d_j) = \begin{cases} \left| \frac{d_j}{p_j} \right| & \text{si } |d_j| \leq p_j \\ 1 & \text{si } |d_j| > p_j \end{cases}$$

Ce type de fonction de préférence permet au décideur de préférer progressivement a à b en fonction de l'écart entre $f_j(a)$ et $f_j(b)$. Un seul paramètre p_j est ici nécessaire. Généralement on demandera au décideur de le fixer assez haut avant de le descendre jusqu'au moment où sa préférence n'est plus stricte.

– Critère à palier :

$$H_j(d_j) = \begin{cases} 0 & \text{si } |d_j| \leq q_j \\ \frac{1}{2} & \text{si } q_j < |d_j| \leq p_j \\ 1 & \text{si } |d_j| > p_j \end{cases}$$

Cette fonction représente un pseudo-critère décrit en 2.1.5. Deux paramètres doivent être définis : q_j par le bas et p_j par le haut. Lorsque l'écart entre $f_j(a)$ et $f_j(b)$ est inférieur à q_j nous sommes en situation d'indifférence. Lorsque cette différence se situe entre q_j et p_j on se situe à un stade de préférence faible en faveur de l'action a . Enfin si l'on dépasse p_j on préférera strictement a à b .

– Critère à préférence linéaire avec zone d'indifférence :

$$H_j(d_j) = \begin{cases} 0 & \text{si } |d_j| \leq q_j \\ \frac{|d_j| - q_j}{p_j - q_j} & \text{si } q_j < |d_j| \leq p_j \\ 1 & \text{si } |d_j| > p_j \end{cases}$$

Cette fonction est similaire à la précédente excepté le fait que la préférence de a envers b croît linéairement entre q_j et p_j alors qu'elle est fixée dans le cas précédent.

– Critère gaussien :

$$H_j(d_j) = 1 - e^{-\frac{d_j^2}{2s_j^2}}$$

Dans cette fonction la préférence de a par rapport à b croît de manière continue en fonction de d_j . Un seul paramètre s_j doit être fixé et contrôle l'aplatissement de la fonction de

préférence gaussienne : il correspond à un degré de préférence moyen et se situe donc entre un seuil d'indifférence q_j et un seuil de préférence stricte p_j .

Ces fonctions sont les plus courantes dans la méthodologie PROMÉTHÉE. Il n'est cependant pas prohibé d'utiliser d'autres fonctions de préférences. La figure 2.1 résume les différents types de critères et leur représentation graphique (uniquement la partie droite de ceux-ci).

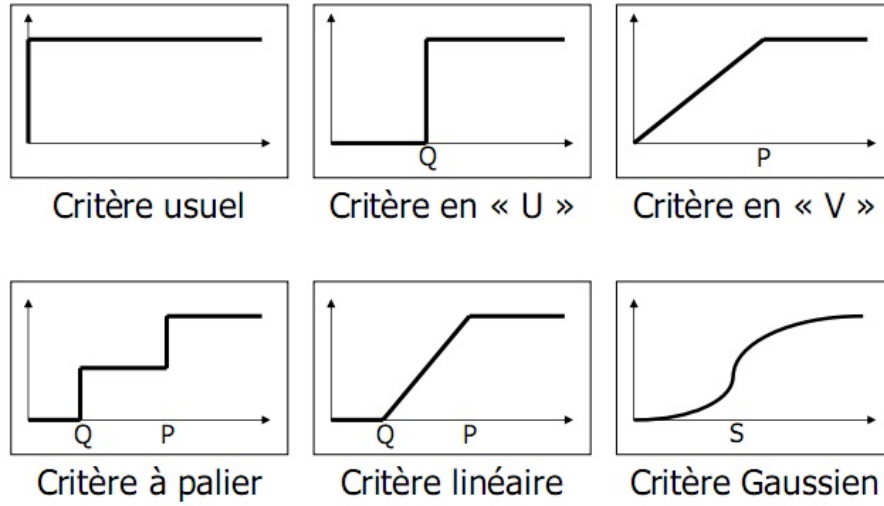


FIGURE 2.1 – Fonctions de préférence

[Mar02]

2.3.3 Relation de surclassement évaluée et flux de surclassement

Indice de préférence multicritère

Reprenons le tableau d'évaluation défini en 2.1.6 et supposons qu'un critère généralisé ait été associé à chaque $f_j(\cdot)$. Nous pouvons à partir de là définir l'indice de préférence multicritère :

$$\pi(a, b) = \sum_{j=1}^k P_j(a, b) * w_j \quad \left(\sum_{j=1}^k w_j = 1 \right)$$

où les $w_j > 0$, $j = 1, 2, 3, \dots$ sont les poids des critères. Ces derniers représentent l'importance du critère au regard du décideur et constitue une information additionnelle enrichissant la structure de préférence entre les critères. L'indice de préférence $\pi(a, b)$ est une mesure de préférence de a sur b sur l'ensemble des critères et possède les propriétés suivantes :

$$\pi(a, a) = 0$$

$$0 \leq \pi(a, b) \leq 1$$

Plus cet indice est proche de 0 et plus la préférence globale de a sur b est faible. Inversement le fait qu'il se rapproche de 1 signifie que a est fortement préféré à b .

flux de surclassement sortant

Ce nouvel indice se définit par :

$$\phi^+(a) = \frac{1}{n-1} \sum_{x \in A} \pi(a, x)$$

Ce flux exprime le caractère surclassant de l'action a par rapport aux $n-1$ autres actions (sa puissance). La somme étant pondérée il est également compris entre 0 et 1.

flux de surclassement entrant

Inversement au flux précédant nous avons ici un indice normé exprimant le caractère surclassé de l'action a par rapport aux $n-1$ autres actions (sa faiblesse).

$$\phi^-(a) = \frac{1}{n-1} \sum_{x \in A} \pi(x, a)$$

flux de surclassement net

Le flux de surclassement net n'est autre que la différence entre le flux de surclassement sortant et le flux de surclassement entrant, pouvant être positif ou négatif :

$$\phi(a) = \phi^+(a) - \phi^-(a)$$

flux unicritères

Ces flux sont calculés séparément pour chaque critère. Nous y retrouvons la notion de flux sortant et entrant.

$$\phi_j^+(a) = \frac{1}{n-1} \sum_{x \in A} P_j(a, x), j=1,2,\dots,k$$

$$\phi_j^-(a) = \frac{1}{n-1} \sum_{x \in A} P_j(x, a) j=1,2,\dots,k$$

Les flux normaux se retrouvent en sommant les flux unicritères multipliés par le poids w_j du critère. De plus nous pouvons retrouver le flux net unicritère :

$$\begin{aligned} \phi_j(a) &= \phi_j^+(a) - \phi_j^-(a) \\ \phi(a) &= \sum_{j=1}^k \phi_j(a) w_j \end{aligned}$$

A partir de ces divers éléments l'homme d'études peut commencer à ranger les différentes actions.

2.3.4 PROMÉTHÉE I

Cette méthode de rangement partiel utilise les flux sortant et entrant afin de construire la structure de préférence. Nous construisons d'abord les deux préordres complets (S^+, I^+) et (S^-, I^-) induits par ceux-ci :

$$\begin{cases} aS^+b \iff \phi^+(a) > \phi^+(b) \\ aI^+b \iff \phi^+(a) = \phi^+(b) \end{cases}$$

Une action est d'autant meilleure que son flux sortant est élevé

$$\begin{cases} aS^-b \iff \phi^-(a) < \phi^-(b) \\ aI^-b \iff \phi^-(a) = \phi^-(b) \end{cases}$$

Une action est d'autant meilleure que son flux entrant est faible

PROMÉTHÉE I construit enfin le rangement partiel final en faisant l'intersection de ces deux préordres :

$$\begin{aligned} aP^{(1)}b &\iff \begin{cases} aS^+b \text{ et } aS^-b \\ aS^+b \text{ et } aI^-b \\ aI^+b \text{ et } aS^-b \end{cases} \\ aI^{(1)}b &\iff aI^+b \text{ et } aI^-b \\ aR^{(1)} &\quad \text{sinon} \end{aligned}$$

Nous retrouvons une structure de préférence $(P^{(1)}, I^{(1)}, R^{(1)})$ désignant la préférence, l'indifférence et l'incomparabilité dans PROMÉTHÉE I. Deux actions a et b sont incomparables si a est nettement supérieure à b sur un sous-ensemble de critères et qu'à son tour b est nettement meilleur que a sur un autre sous-ensemble de critère. Le modèle ne peut alors trancher en faveur d'une action ou l'autre.

2.3.5 PROMÉTHÉE II

Cette méthode fournit un rangement complet en utilisant une structure de préférence reposant sur les flux nets :

$$aP^{(2)}b \iff \phi(a) > \phi(b)$$

$$aI^{(2)}b \iff \phi(a) = \phi(b)$$

L'incomparabilité que l'on pouvait avoir dans PROMÉTHÉE I ne se retrouve plus dans PROMÉTHÉE II étant donné l'utilisation des flux nets. Cependant il est à noter qu'une partie de l'information est perdue, le flux net étant moins "informatif" que les flux entrant et sortant couplés.

2.3.6 Le plan GAIA

GAIA est une méthode de visualisation géométrique servant de complément aux méthodes PROMÉTHÉE. Elle permet de repérer les critères conflictuels, de chercher les bons compromis et de faciliter la compréhension du problème de décision auquel le décideur se trouve confronté. Sur ce plan on retrouve les actions potentielles, les critères et les poids liés représentés respectivement par des points, des axes et un vecteur particulier π . La base de cette méthode est la construction de la matrice des flux unicritères vus au point 2.3.3 :

$$\Phi = \begin{pmatrix} \Phi_1(a_1) & \Phi_2(a_1) & \dots & \Phi_j(a_1) & \dots & \Phi_k(a_1) \\ \Phi_1(a_2) & \Phi_2(a_2) & \dots & \Phi_j(a_2) & \dots & \Phi_k(a_2) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \Phi_1(a_i) & \Phi_2(a_i) & \dots & \Phi_j(a_i) & \dots & \Phi_k(a_i) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \Phi_1(a_n) & \Phi_2(a_n) & \dots & \Phi_j(a_n) & \dots & \Phi_k(a_n) \end{pmatrix}$$

Chaque action a_i sera donc représenté par un point

$$\alpha_i : (\Phi_1(a_i) \ \Phi_2(a_i) \ \dots \ \Phi_j(a_i) \ \dots \ \Phi_k(a_i))$$

d'un espace euclidien à k dimensions dont les axes représentent les k critères. Le but de la méthode GAIA est de projeter orthogonalement ces points sur un plan passant par l'origine et ce en essayant de perdre le moins d'information possible. Les points ainsi obtenus sont notés A_i . Pour ce faire, une analyse en composantes principales sera utilisée. Cette analyse est reprise dans les annexes.

Représentation dans le plan GAIA

Comme expliqué précédemment les actions sont représentées par des points sur le plan GAIA. Voyons ce qu'il en est pour les critères et les poids.

- Les critères : Contrairement aux actions les vecteurs sont unitaires (chaque dimension de R^k représentant un critère). Le résultat de la projection d'un critère j noté c_j sera donc la projection d'un vecteur e_j de coordonnées

$$(0,0,...,1,...,0)$$

- Les poids : Il s'agit ici d'un vecteur unique \bar{w} dont les coordonnées sont en fait les différents poids attribués à chaque critère. Ce vecteur est également appelé "stick de décision" car il permet de facilement voir dans quelle direction (et donc vers quelles actions) va la préférence lorsqu'on change les poids. Sa projection donne un vecteur unique Π . Plus celui-ci est court et plus son pouvoir de décision est faible. En effet cela signifie alors que le vecteur \bar{w} est pratiquement orthogonal au plan et que la simple modification d'un poids peut entraîner un changement de direction de ce dernier.

Lorsque tous ces éléments sont représentés sur le plan GAIA il est simple pour le décideur d'analyser la situation :

- Une action i dont la représentation A_i sur le plan GAIA possède la meilleure projection orthogonale sur c_j est l'action optimale pour ce critère.
- Une action i dont la représentation A_i sur le plan GAIA possède la meilleure projection orthogonale sur Π est celle permettant d'avoir le meilleur compromis.
- Si deux axes représentants des critères ont un produit scalaire négatif, ceux-ci sont antagonistes

Nous pouvons voir sur la figure 2.2 la représentation graphique de la méthode PROMÉTHÉE II et du plan GAIA fournie par le logiciel D-Sight.

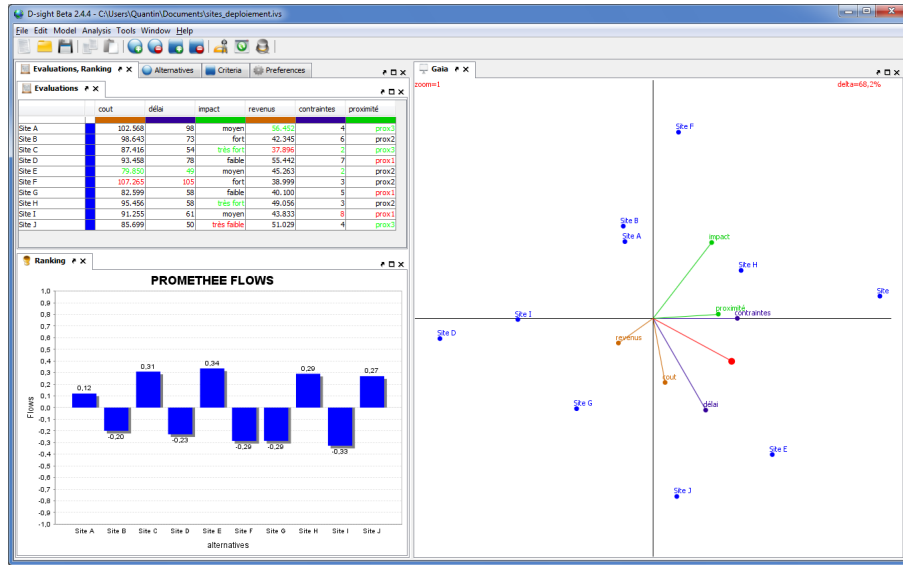


FIGURE 2.2 – Représentation graphique de la méthode PROMÉTHÉE II et du plan GAIA fournie par le logiciel D-Sight

Conclusions

Cette étude de l'aide multicritère à la décision et en particulier des méthodes PROMÉTHÉE fût l'une des premières tâche effectuée dans le cadre de ce mémoire. Elle permit par la suite de visualiser plus facilement l'architecture de la base de données du marketplace en se basant sur les paramètres à implémenter et la relations entre ceux-ci. De plus la compréhension des méthodes PROMÉTHÉE fût essentielle afin d'utiliser le code source du logiciel D-Sight et de savoir quelles méthodes étaient importantes. Dans le cadre du développement et de l'implémentation 5 nous reviendrons sur certaines notions importantes vues dans cette partie.

Chapitre 3

Marketplace et E-Business

Introduction

Dans ce chapitre nous nous intéresserons aux modèles actuels d'E-business afin d'arriver à définir correctement le rôle d'un marketplace et les besoins sécuritaires de telles infrastructures. Cette partie a pour rôle principal d'initier le lecteur au concept de marketplace afin qu'il comprenne le but et les enjeux de ce mémoire. Nous définirons tout d'abord les principales formes d'E-Business existantes et en particulier la notion d'intermédiation dont le marketplace est une entité spécifique. Ensuite nous passerons en détails les différentes mesures de sécurité à appliquer dans le cadre d'une telle infrastructure.

3.1 Contexte et définitions

Afin de comprendre l'environnement dans lequel se situe les places de marché nous devons d'abord définir les différentes notions propres à son contexte. En effet nous savons qu'un marketplace est une forme d'E-Business mais quelles sont les caractéristiques de telles infrastructures ? En quoi ces structures sont elles avantageuses pour les entreprise ? Nous répondrons à ces questions dans cette partie.

3.1.1 E-Business

D'après l'agence wallonne des télécommunications¹ l'E-Business est une notion :

"recouvrant les différentes applications possibles de l'informatique faisant appel aux technologies de l'information et de la communication (TIC) pour traiter de façon performante les relations de communication d'information d'une entreprise telle un PME avec des

1. <http://www.awt.be/index.aspx>

organisations externes ou des particuliers. Les technologies utilisées sont principalement celles de l'internet et du web.”

En termes simplifiés le terme générique d'E-Business désigne toute activité économique réalisée par internet. Les deux moteurs de ce concept sont la gestion orienté client et le recours aux TIC. Ce terme recouvre, entre autres, trois notions distinctes que nous allons dès à présent détailler :

Le commerce B2C(Business to Consumer)

La forme la plus connue d'E-Business. Ce type de commerce concerne les entreprises utilisant le web afin de communiquer directement avec l'acheteur et de lui vendre ses produits. Contrairement à ce que l'on pourrait croire à la vue des nombreux sites présents sur internet il représente moins de 10% du chiffre d'affaire attribué à l'E-Business. Le commerce B2C facilite grandement le quotidien des entreprises et de nombreux avantages peuvent être cités par rapport à un commerce traditionnel :

- Il permet une information générale disponible et de nombreux sites permettant aux consommateurs de comparer prix et produits.
- Il permet de garder une trace des habitudes de consommation du client et donc de lui proposer des produits susceptibles de l'intéresser.
- Il permet à l'utilisateur un achat direct et sans contraintes

Ce type de commerce électronique est donc intéressant pour des commerces voulant faciliter les transactions avec le client tout en réalisant des économies d'infrastructure. Cependant la plupart des échanges commerciaux actuels se déroulant entre les entreprises un autre modèle de plates-formes s'est vu créé : le B2B.

Le commerce B2B(Business to Business)

Il s'agit ici de commerce électronique entre entreprises, les supports électronique permettant des échanges d'informations entre-elles : fournisseurs, sous-traitant, entreprises clientes. Ce mode peut prendre de nombreux aspects comme la sous-traitance, le service après-vente, l'acquisition d'équipements et la maintenance, etc. Les avantages de ce genre de commerce sont nombreux : d'une part la réduction des coûts de transactions commerciales et coûts internes

(moins d'intervenants) permettent aux entreprises de réaliser de nombreux bénéfices et aux plus petites d'entres-elles de pouvoir se lancer dans le commerce électronique. De plus la rapidité des transactions permettent à celles-ci de travailler en flux tendu sans se soucier à l'avance de l'approvisionnement.

Ce genre de commerce permettra donc à de nombreuses entreprises d'avoir un champ d'action agrandi tout en permettant une plus grande liberté dans les échanges commerciaux. Cependant une entreprise voulant se lancer dans ce genre de commerce devra construire elle-même sa plate-forme de commerce B2B afin de fournir ses services.

Parfois des entreprises collaborant via des plates-formes B2B sont amenées à faire appel à des entreprises extérieures afin de travailler ensemble. Ces dernières sont appelées plates-formes d'intermédiation.

L'intermédiation

D'après l'agence wallonne des télécommunications :

"Dans le cadre des échanges électroniques d'informations relatifs à une application e-business, l'intermédiation concerne les activités que les acteurs émetteurs ou récepteurs de ces informations ne peuvent pas ou ne veulent pas assumer et dont ils confient la réalisation à une entité tierce. L'intermédiation correspond à ces activités ou métiers nouveaux engendrés par l'e-business et plus globalement par les usages des TIC".

La figure 3.1 prise sur le site internet de l'agence Wallonne des télécommunications² montre le schéma d'une société d'intermédiation transformant des bons de commande et des factures afin d'assurer la bonne communication entre deux entreprises.

Ces sociétés existent donc uniquement grâce à l'avènement du commerce électronique et permettent aux entreprises de ne pas se soucier de certaines étapes dans l'établissement d'échanges commerciaux. Une forme particulière d'intermédiation consiste à installer sur le web un site de rencontre et d'échange entre sociétés afin de favoriser la communication et faciliter le processus d'appel d'offre. Cette définition est évidemment celle des places de marché virtuelles.

3.1.2 Marketplace et services proposés

Un marketplace est défini précisément par l'agence wallonne des télécommunications :

2. <http://www.awt.be/index.aspx>

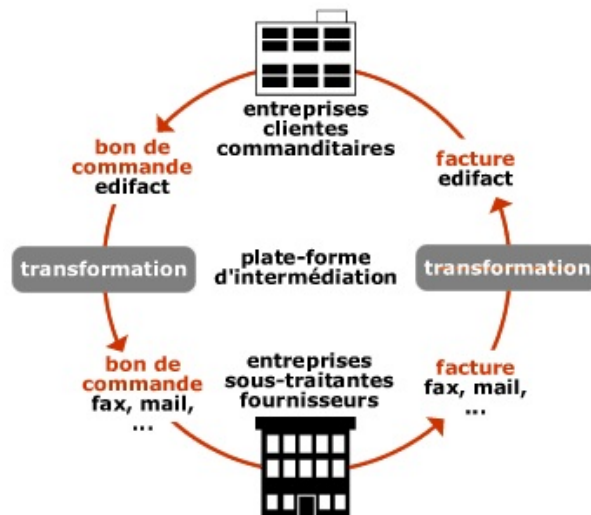


FIGURE 3.1 – Intermédiation assurée par une société tierce
Agence Wallonne des télécommunications

”Un endroit permettant la mise en présence sur le Web, par une tierce partie, d’offres et de demandes en vue d’aboutir à l’établissement d’une transaction.”

Cette description, bien que simple, résume parfaitement le but même de l’existence de ces plateformes. Afin d’approfondir cette description nous listerons la liste des services inhérents à un marketplace. Cependant le concept d’appel d’offre doit lui aussi être décrit dans le but de mieux comprendre ce modèle économique.

Appel d’offre

Un appel d’offre est une procédure dans laquelle un commanditaire 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 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 marketplace 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. De plus les marketplace fournissent aux entreprises de nombreux outils secondaires. Généralement les appels d’offres sont décomposés en un certain nombre de rounds.

Ces derniers sont des échéances avant lesquelles les fournisseurs doivent avoir répondu à l’appel d’offre. A la fin d’un round, le nombre de fournisseurs déterminé par le commanditaire

est sélectionné afin de pouvoir participer au round suivant. Cette utilisation de deadline permet au commanditaire de pousser les fournisseurs à faire des efforts afin de rester ”en course”.

Services proposés

Les marketplaces sont des sites tentaculaires regroupant de nombreux services (optionnels ou non) mis en place afin de faciliter les transactions :

- un support pour poster la description d’une offre ou d’une demande
- un outil de recherche parmi les offres ou les demandes
- un outil de recherche d’offres répondant aux caractéristiques d’une demande
- un outil de recherche de demandes pour une offre déterminée
- une gestion d’appel d’offres
- un système d’alerte lors de l’ajout d’une offre
- une gestion de mise en vente de produits ou services
- un support de négociation en ligne
- Un support de transaction en ligne

De nombreux marketplaces existent sur le web. Citons par exemple Oracle³, Bobex⁴ ou encore Windows Marketplace⁵. Cependant ces marketplace n’appuie pas vraiment le commanditaire dans son choix à l’aide de méthodes de rangement des offres.

Introduction des méthodes PROMÉTHÉE

L’innovation voulue par le département CoDE-SMG est la construction d’un modèle de marketplace qui, au delà des services habituellement proposés, utiliserait le logiciel D-Sight afin d’aider l’acheteur dans son problème de décision, lui permettant ainsi de choisir la meilleure alternative (offre). Cet ajout serait un réel avantage pour le commanditaire puisque celui-ci pourrait passer une offre tout en prévenant les fournisseurs de ses critères de choix ainsi que de leurs pondérations.

Les fournisseurs ainsi prévenus essayeraient alors de faire des efforts en terme de qualité ou de prix afin que leur offre réponde aux attentes du commanditaire. Les méthodes de rangement multicritère permettraient également au commanditaire de se faire une opinion générale vis-à-vis des offres proposées.

3. <https://oracle.corporateperks.com/login>

4. <http://www.bobex.com/>

5. <http://www.windowsmarketplace.com/>

3.2 Sécurité et aspects juridiques

Les places de marché sont des sites permettant aux entreprises un réel gain d'argent. Une cause est que la concurrence se rassemble en un endroit spécifique (avantage pour les acheteurs) et que des petites entreprises peuvent participer au marché plus facilement (avantage pour les fournisseurs). Le fait que les places de marché électroniques concentrent cette multitude de sociétés effectuant des transactions engendre cependant un réel besoin sécuritaire ainsi qu'une maintenance continue au niveau des serveurs. En effet le responsable de la marketplace s'engage judiciairement à mettre en place des garanties en matière de :

- Sécurisation et confidentialité des échanges
- Qualité des acteurs (accès ouvert ou fermé, conditions d'utilisation)
- responsabilité/garanties liées aux exigences de fonctionnement permanent de la place de marché (maintenance constante, serveur miroir)

Ces garanties sont évidemment liées, de près ou de loin, à la conception du marketplace au niveau informatique. Pour les respecter des questions banales mais influençant la conception doivent se poser. Parmi elles on peut avoir :

- Quel langage utiliser, sachant que celui-ci doit permettre une évolutivité constante de la solution ?
- Quelles informations seront visibles par l'utilisateur ?
- A quel niveau sécuriser les échanges ? Certificats numériques, authentification par mot de passe, mise en place d'une méthode de cryptage d'informations ?

Ces questions doivent être une base de raisonnement dans la conception de l'application web demandée. Nous y répondrons dans le chapitre 4

Conclusions

La connaissance du domaine de l'E-Business et, en particulier, des places de marché nous a permis de nous poser des questions qui, sous leurs airs banals, ont fourni une aide non négligeable lors de la conception du marketplace, et ce dès les premiers choix techniques effectués. Une place de marché doit être un endroit simple d'accès et d'utilisation, permettant de s'identifier et de communiquer facilement tout en étant sécurisé. Cette vision un peu plus réfléchie obtenue après étude nous a permis de concevoir le marketplace tout en gardant en mémoire la structure désirée.

Deuxième partie

Conception

Chapitre 4

Étude technique

Introduction

Dans ce chapitre nous ferons un tour d’horizon des différents choix techniques effectués afin de commencer la conception du marketplace. Dans la partie 4.1 nous passerons en revue les cinq langages dynamiques actuels les plus usités dans le cadre d’un développement web. Nous évoquerons également en 4.1.6 une autre possibilité découverte sur le tard, trouvant sa place dans cette section malgré le fait qu’il s’agisse plus d’un Framework : le Google App Engine. Ensuite nous détaillerons le langage choisi, ses possibilités et son fonctionnement en 4.2. Enfin nous passerons en revue les autres aspects techniques en 4.3 : base de données, cryptage de données, etc.

4.1 Choix des solutions et description du langage choisi

Introduction

Choisir un langage de programmation n’est jamais chose aisée. En effet, la multitude de langages peut sembler de prime abord rebutant. Cependant lorsque les critères de choix sont posés il devient assez simple de prendre une décision. Ces derniers sont difficiles à définir même si certains sont plus important que d’autres dans le cadre du développement d’un prototype. En effet nous avons basés notre choix de critères sur le fait que l’application serait reprise par après et devait donc bénéficier d’un langage permettant une prise en main facile afin que l’évolutivité de la solution soit optimale. Nous avons donc isolés cinq critères qui nous paraissaient primordiaux :

- Orienté Objets : Cette notion permettant de garantir une bonne pratique de la programmation est indispensable dans le cadre de la programmation actuelle.

- Modèle-Vue-Contrôleur (MVC) : Un bon développement web se doit de séparer ces trois couches. Ce critère n'est pas essentiel étant donné que le programmeur doit avoir cette approche dans sa conception. Cependant les langages forçant le concepteur à adopter ce modèle seront préférés.
- Indépendance : L'indépendance vis-à-vis d'une plate-forme signifie que les programmes fonctionnent de manière parfaitement similaire sur différentes architectures matérielles. Le développement peut alors se faire sur une architecture donnée tout en faisant tourner l'application sur d'autres. Cette caractéristique d'un langage est importante afin de pouvoir bouger d'une plate-forme à une autre et donc d'assurer une évolutivité de la solution
- Typage fort : Afin de s'assurer un suivi et des possibilités de debuggage aisées tout au long de la conception le langage devra avoir un typage fort, c'est à dire que les données manipulées devront être décrites par des types (int, String, etc).
- Librairies et documentations : Afin de garantir un développement aisé le langage devra posséder une documentation conséquente ainsi qu'un large choix de librairies.

D'autres critères de choix seront bien sûr cités mais ceux-ci restent les principaux intervenant dans notre choix final. En effet les autres critères comme la facilité de syntaxe ou le temps d'apprentissage, bien qu'important, ne sont pas essentiels à la décision finale. Cependant nous privilégierons également les langages orientés vers le développement d'applications importantes.

4.1.1 PHP [Zer07]

PHP (Hypertext PreProcessor) est un langage de script libre utilisé afin de construire des pages web dynamiques. Bien qu'il soit actuellement le langage de développement web le plus utilisé, PHP est cependant moins utilisé dans le cadre du développement d'application RIA (Rich Internet Application) complexes. PHP possède certains défauts importants : tout d'abord il est faiblement typé, ce qui peut engendrer des problèmes de compréhension lors de la relecture du code, ou de la reprise du projet par une autre personne. De plus PHP étant un des langages les plus utilisés au monde, cela en fait également un des plus attaqués, ce qui pose problème dans le cadre du développement d'une application commerciale.

Quelques qualités appréciables viennent cependant redresser la balance : PHP est performant

dans son exécution, à une communauté dynamique et une API (Application Programming Interface) très complète ne nécessitant pas de faire appel à des packages spécifiques.

4.1.2 Python [Hol08]

Python est un langage de développement apparu en 1990. Il est disponible afin de créer des application web via le framework Django. Ce langage, profitant de sa création tardive, à de nombreux avantages.

Il est orienté objet et possède un typage fort, ce qui permet de commencer son apprentissage sur des bases connues et saines. De plus Django s'inspire du principe MVC ou MTV (Un template gère la vue) et est composé de trois parties distinctes :

- Un langage de template flexible qui permet de générer du html, xml ou des autres formes de texte
- Un controleur fourni sous la forme d'un "remapping" d'URL à base d'expressions rationnelles
- Un API HTML d'accès aux données automatiquement généré par le framework

Enfin Python profite d'une grande portabilité : il tourne sur des plateformes Windows, Linux/Unix, Mac OSX, etc. Cependant Python n'est pas encore utilisé massivement chez les développeurs web, ce qui peut avoir comme conséquences de ne pas toujours trouver une librairie voulu. De plus il est plus lent dans son exécution que les autres langages décrits dans cette section.

4.1.3 Ruby [Tho07]

Ruby, né en 1993, a été créé avec les mêmes idées de base que python : un langage lisible, facile à implémenter pour des petits projets et 100 % orienté objets. Via son framework "Rails" il impose également un développement web basé sur le modèle MVC. Il jouit également d'une grande portabilité. Il est de plus un peu moins permissif que Python au niveau des déclarations d'attributs et de méthodes.

Ruby a cependant un défaut majeur : il évolue vite et la compatibilité ascendante n'est pas toujours respectée. Il se peut donc que des librairies ne soient plus compatibles après certaines mises à jour.

4.1.4 J2EE [Keo02]

J2EE (Java Enterprise Edition) est une spécification répondant aux préoccupations d'entreprises au niveau de la conception de sites web dynamiques, essentiellement concentrées sur trois parties distinctes : la logique de présentation des données et les interfaces graphiques, la logique métier et les composants applicatifs et la logique d'accès aux données et aux ressources externes. Rien qu'en lisant cette description on peut remarquer que la découpe en couche et le MVC sont au coeur de la logique J2EE.

Basée essentiellement sur le langage Java J2EE hérite des mêmes spécificités : orienté objet, multi-plateforme via la Java Virtual Machine, typage fort, etc. De plus le programmeur utilisant J2EE pourra utiliser les bibliothèques Java de base. Java étant un langage de programmation très utilisé il en découle qu'en utilisant J2EE le programmeur aura accès à une bibliothèque importante et très documentée. En outre J2EE a été créé dans le but de développer des applications d'entreprise, ce qui correspond à nos attentes. Enfin J2EE est considéré comme le langage web dynamique le plus rapide car la compilation des fichiers servant à l'affichage n'est effectuée que si le serveur s'aperçoit que le fichier a changé depuis le dernier envoi au client.

La grande qualité de J2EE a également un côté plus désavantageux : tellement orienté vers le découpage du projet en plusieurs couches, il utilise un nombre impressionnant de technologies différentes n'étant toujours aisées à appréhender.

4.1.5 ASP .NET [Mac08]

ASP .NET est un ensemble de technologies de programmation web créé par Microsoft. Il permet de créer des sites web dynamiques en proposant de programmer côté client et côté serveur. La grande innovation de ASP .NET est qu'il permet d'utiliser des langages de programmations différents comme C# ou Python. Le grand principe de ASP est que le code côté serveur est compilé en de simple dll (Dynamic Link Library).

Ce langage supporte et promeut également la programmation orientée objet ainsi que le MVC afin de forcer l'auteur d'un site à effectuer une "bonne pratique" de la programmation. Il bénéficie de plus, via les langages comme c#, d'un typage fort. ASP .NET ayant été créé par Microsoft il bénéficie d'un suivi efficace et d'une bibliothèque assez imposante.

Le gros problème, reproché par un nombre important de programmeurs, est que ASP a une portabilité restreinte : Microsoft voulant évidemment l'empêcher de tourner sur d'autres plateformes que la sienne. Bien que ce problème commence à se résoudre (certaines version de Linux commencent à pouvoir le supporter) cela reste un défaut important à prendre en compte.

4.1.6 Google Web Toolkit (GWT) [Jab09]

Cette solution est un peu à part étant donnée qu'il s'agit d'un Framework de développement web estampillé AJAX (Asynchronous JavaScript and XML). Ajax est un ensemble de technologies libres s'appuyant sur trois piliers :

- Le DOM (Document Object Model), terme utilisé pour décrire la représentation arborescente d'une page HTML.
- XML afin de remplacer le format des données informatives (JSON) et visuelles (HTML).
- JavaScript afin d'afficher les informations et d'interagir avec.

La force d'AJAX est d'avoir su marier le DOM et les appels asynchrones au serveur web. Les préceptes de base du standard Ajax s'appuient sur deux mécanismes fondamentaux :

- Le navigateur échange de manière asynchrone avec le serveur par l'intermédiaire de messages (XML, texte brut, etc)
- Ces messages contiennent des données de présentation (portions de page HTML) ou des informations destinées à être interprétées par le navigateur qui modifie ensuite des portions de page HTML

Cette technologie utilise Java comme langage principal bien que toute la partie client de l'application soit automatiquement convertie en javascript, ce qui peut éventuellement poser problème aux utilisateurs ayant désactivé le support de leur navigateur. Cependant d'autres langages comme PHP peuvent être utilisés. De plus, les sites créés n'ont plus de rechargement de page : Seul les données sont rechargées afin d'actualiser la page via un appel asynchrone au serveur. Cette solution de développement ayant été découverte assez tard nous ne l'avons pas utilisée. Il est toutefois intéressant de savoir que ce Framework de développement existe et d'en dresser les avantages et inconvénient au vue d'un éventuel futur portage. Un des avantages majeurs de GWT est que le programmeur n'a plus à se soucier de la sécurité entre le serveur et le navigateur bien qu'il puisse également implémenter sa propre solution. De plus ce Framework étant développé par Google, société nait du web et ayant pour but le développement du web tout en ayant comme leitmotivs la rapidité, la souplesse et la légèreté des applications web. Enfin, un des derniers avantages, étant aussi un inconvénient, est que l'application doit tourner

sur un serveur web, appartenant à Google et possédant le serveur d'application *Google App Engine*. Il est cependant possible de le déployer sur un serveur personnel mais cela à comme conséquence un fonctionnement réduit. Le fait de déployer l'application sur un serveur appartenant à Google sous-entend que l'entretien de ce dernier est à leur charge, ce qui permet à la société responsable de l'application de ne pas se soucier de cette tâche. Cet hébergement implique également que le code de l'application se trouve sur un serveur uniquement accessible via une interface. Bien que Google signe un contrat de confidentialité lorsque l'on héberge une application sur ses serveur cet aspect n'en reste pas moins un gros désavantage dans le cadre du déploiement d'une application commerciale.

4.1.7 Choix final

Le tableau 4.1 récapitule les avantages et inconvénient cités ci-dessus :

	MVC	Orienté Objet	Bibliothèque	Multi-plateformes	Typage fort
PHP	Oui depuis la version 4	Oui	Importante	Oui	Non
Python	Oui	Oui	Partielle	Oui	Oui
Ruby	Oui	Oui	Partielle	Oui	Oui
J2EE	Oui	Oui	Importante	Oui	Oui
ASP .NET	Oui	Oui	Importante	Non	Oui

TABLE 4.1 – Avantages et inconvénients des différents langages de programmation dynamiques

Le fait d'écrire qu'une bibliothèque est partielle ne signifie pas, à nos yeux, que peu de librairies soient disponibles. Cela met plutôt en avant le fait que les autres langages proposés en possèdent plus et que celles-ci sont mieux documentées.

Nous pouvons observer que ces langages possèdent le plus souvent les avantages requis afin d'être utilisé dans le cadre de notre mémoire. Cependant J2EE se démarque légèrement. En plus du fait qu'il répond positivement à toutes nos attentes cette spécification a été spécialement développée afin de concevoir des sites web importants pour les entreprises. De plus le programme D-Sight étant programmé en Java l'utilisation de J2EE sera un plus car il suffira d'importer la librairie du programme afin de pouvoir utiliser l'ensemble de ses méthodes et classes. Enfin J2EE bénéficie de toutes les améliorations apportées au langage Java depuis des années. Notre choix final se portera donc sur cette spécification.

4.2 Description de la spécification J2EE [All02] [Gir02]

Dans cette section nous examinerons en détails le développement d'une application web avec J2EE. Dans la section 4.2.1 nous présenterons les différentes architectures multi-tiers et les couches correspondantes. Après cela nous passeront en revue les différentes technologies servant à implémenter ces dernières. De plus, nous introduirons la Java Standard Tag Library, utilisé conjointement avec les JSP et permettant un codage plus propre de celles-ci. Une fois ces notions acquises nous parlerons du serveur d'application utilisé dans le cadre du développement : Apache Tomcat.

4.2.1 Architectures Multi-tiers

J2EE propose différentes structures afin de développer une application web sur des bases saines. Typiquement les architectures J2EE spécifient plusieurs couches complémentaires qui comportent différents éléments ou composants de l'application J2EE déployée. Ces architectures se découpent selon différents scénarios :

- Une architecture à deux niveaux, comportant un serveur web J2EE et un serveur de base de données. Ce modèle est de type modèle client-serveur
- Une architecture à trois niveaux, comportant un serveur web J2EE, un serveur métier J2EE et un serveur de base de données
- Une architecture à quatre niveaux, comportant la partie cliente sur laquelle s'exécute des applications clientes, un serveur web J2EE, un serveur métier J2EE et un serveur de base de données

L'architecture utilisée dans notre développement sera à trois niveaux afin d'avoir un découpage entre les différentes parties de l'application, à savoir la présentation, le métier et les données. Cette séparation respectant tout à fait le paradigme MVC une évolution des niveaux indépendamment les uns des autres sera possible à l'avenir et permettra donc une meilleure évolutivité de la solution. De plus ce type d'architecture est relativement extensible, en vue par exemple de l'insertion de nouveaux tiers. La figure 4.1 provenant du site internet de Sun Microsystems¹ montre l'architecture adoptée par la plupart des applications web J2EE.

1. <http://java.sun.com>

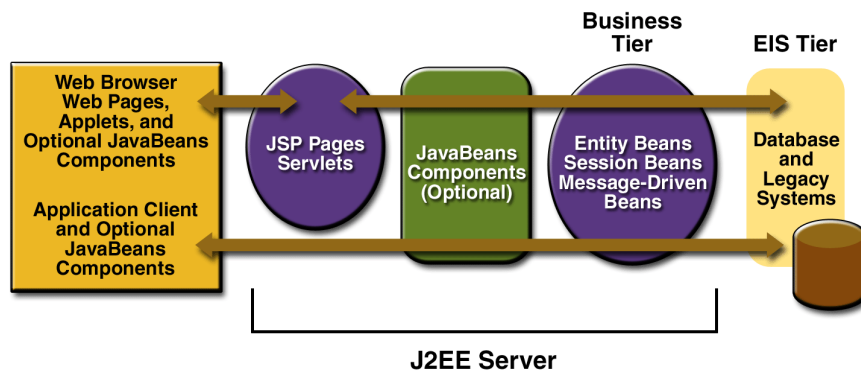


FIGURE 4.1 – Architecture d’une application web J2EE

4.2.2 Les servlets

Les servlets, implémentant la couche contrôleur du Pattern MVC, sont des programmes java qui s’intègrent à un serveur d’application afin de traiter, côté serveur, les requêtes issues d’un navigateur web client. Ils doivent obligatoirement s’appuyer sur un serveur web supportant la technologie Java Server. Les avantages des servlets sont nombreux :

- L’indépendance par rapport aux plates-formes
- L’indépendance par rapport aux protocoles de communication
- La réutilisation du code et l’intégration facilitée

Il est possible d’utiliser un programme client, quelque soit le langage, afin d’envoyer des requêtes à un servlet. Ce client peut par exemple être une page HTML, une applet java ou encore un programme écrit dans un langage autre que Java. De l’autre côté, c’est à dire au niveau serveur, le servlet traite la requête et génère une sortie dynamique qui est retournée au client, sortie qui peut être en HTML, XML ou encore un flux d’objets Java sérialisés.

4.2.3 Les Java Servlet Pages (JSP)

Les JSP, extension de la technologie des servlets, offrent un moyen simplifié d’écrire ces derniers avec une attention plus forte accordée à la présentation des pages web de l’application. Cette technologie permet d’implémenter la couche Vue du MVC. Si dans les servlets la logique de l’application se trouve dans un fichier java, les JSP permettent quant à eux de mélanger du HTML et du java dans un fichier portant l’extension .jsp. Ce fichier est traité par un compilateur

jsp afin de générer un servlet java. Le résultat est alors compilé à son tour afin de créer du code HTML. Une bonne pratique de codage est d'éviter que les JSP comportent du code Java. Pour cela des solutions à base de balises existent comme le Java Server Pages Standard Tag Library. Le fonctionnement de telles balises et ces avantages seront décrits à la section 4.2.6.

4.2.4 JavaBeans

Un JavaBean (ou bean) est une classe Java servant à représenter des objets d'un type précis et éventuellement des méthodes spécifiques à ces objets. Avec ces composants le programmeur peut procéder à la mise en place de composants Java réutilisable côté serveur pour gérer la présentation des données, l'accès au système de gestion de données, aux EJBs, indépendamment de l'aspect présentation de la page web. Les JavaBeans peuvent évidemment être insérés dans des pages HTML, permettant un affichage de ces attributs ou encore l'exécution de ses méthodes. Les principaux concepts mis en jeu par les JavaBeans sont les suivants :

- Propriétés : Un Bean est paramétrable, ses attributs étant mis à jour via des *getter* et des *setter*.
- Sérialisation : Un Bean est conçu afin que son état puisse être sauvé et restauré par la suite. La persistance de données est donc assurée.
- Réutilisation : Étant donnée qu'il ne contient que des données ou du code métier un bean n'a pas de lien direct avec la couche présentation ou la couche d'accès aux données. De ce fait il est indépendant et peut donc être réutilisé.

Afin de respecter la structure inhérente à ces composant quelques règles de constructions sont imposées lors de la création d'un JavaBean. Tout d'abord celui-ci doit être une classe publique ayant au moins un constructeur par défaut. Ce dernier doit également être public et sans paramètres. En outre il doit implémenter l'interface *serializable* afin de le rendre persistant et de pouvoir sauvegarder son état. Enfin l'ensemble de ses propriétés privées doit être accessible via des méthodes *getter* et *setter* suivant des règles de nommage.

4.2.5 Enterprise JavaBean (EJB) [MH02]

Les Enterprise JavaBean sont des composants de l'architecture J2EE implémentant la logique métier de l'application et interagissant avec les autres composants constituant l'application. Ils ne peuvent fonctionner que dans un conteneur d'EJB, pendant l'exécution des applica-

tions J2EE qui les manipulent. Ce conteneur fait partie intégrante de l'infrastructure du serveur d'application et est directement implémenté et géré par celui-ci. Plusieurs types de composants sont définis dans les spécifications des EJB. En premier lieu nous trouvons les Enterprise JavaBeans Session, principalement dédiés à la mise en place du processus de communication entre un acteur du système et l'application J2EE à l'aide de processus métier, plus particulièrement une entité métier(EJB entité, sessions de bases de données). En second lieu les EJB entité, adapté pour implémenter une entité métier. Cette dernière possède un état, conservé en permanence, généralement à l'aide d'un système de gestion de base de données. L'état de ces EJB est modifié à l'aide des processus métier afin qu'aucun utilisateur n'y ait accès directement. En dernier lieu les EJB message, un moyen de communication entre les composants ou les applications (application cliente, d'autres EJB, etc).

4.2.6 Java Standard Tag Library (JSTL)

Dans la section 4.2.3 nous avons indiqué que les JSP pouvaient mélanger du code HTML et du code Java. Cependant le code java, ou scriptlet, tend à disparaître au profit de balises permettant d'effectuer les mêmes actions (appel d'un bean, boucles, etc). Dans notre projet une utilisation de la Java Standard Tag Library sera faite afin de respecter cette règle de non prolifération du code Java dans les JSP. Les avantages induits par cette substitution sont au nombre de trois :

- Amélioration de la lisibilité.
- Diminution du code à écrire.
- Rendre à la vue son vrai rôle.

Le dernier avantage signifie que grâce aux JSTL les JSP ne contiennent plus de déclaration de variables ni d'appel de fonctions ce qui est préférable afin d'améliorer la cohérence du MVC. Avec ces balise la séparation entre les trois couches est plus claire ce qui en fait un outil indispensable au développement d'une application web J2EE.

4.2.7 Tomcat [Sek07]

Apache Tomcat est un serveur d'application c'est à dire un programme tournant sur un serveur (Apache)qui gère des servlets permettant de générer des pages web dynamiques.

Tomcat est également appelé conteneur de servlet. En effet le web fonctionnant par un

système de question-réponse (requêtes HTTP-réponses HTTP) Tomcat assure la transition : quand il reçoit une requête il instancie un objet `HttpServletRequest` contenant les informations de la requêtes et un objet `HttpServletResponse` servant à fournir la réponse attendue. Un fois cette étape passé Tomcat va utiliser la servlet correspondant à la requête et va invoquer la méthode adéquate de la servlet. La figure 4.2 illustre le schéma de fonctionnement de Tomcat lors du traitement d’une requête : Tomcat ne contient pas d’environnement d’exécution pour les

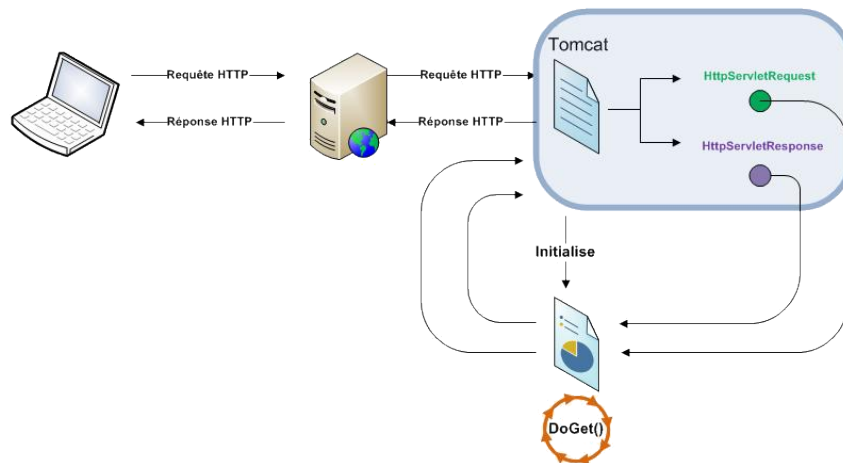


FIGURE 4.2 – Schéma récapitulatif du fonctionnement de Tomcat

EJB. Pour en avoir un le serveur GlassFish est indispensable. Nous reviendrons sur la question dans la partie 5 et expliquerons pourquoi nous avons fini par choisir Tomcat comme serveur d’application.

Conclusions

Si J2EE est une solution idéale en terme de développement d’applications web il n’en reste pas moins que la complexité de cette solution, du fait de ces nombreuses couches et technologies, reste un obstacle dans le cadre du développement proprement dit. Dans cette section nous avons introduits les éléments structuraux les plus courants d’une application J2EE cependant nous verrons au chapitre 5 qu’une application web peut également se construire sans pour autant y incorporer tous ces éléments.

4.3 Autres aspects techniques

Introduction

Dans cette partie nous parlerons des autres aspects techniques sur lesquels un concepteur d'application web doit se pencher. En raccord avec la partie 4.2.1 où nous parlons d'un serveur de base de données nous étudierons celle-ci en 4.3.1. A la partie 4.3.2 nous traiterons en détails de la sécurité de notre application via le cryptage des mots de passe en base de données ainsi que la sécurisation des transactions via un certificat numérique.

4.3.1 Base de données [Zim08] [Sou02]

Afin de stocker les informations des utilisateurs ainsi que toutes les autres informations inhérentes à un marketplace nous allons devoir concevoir une base de données. Étant donné que la plupart des informations fournies à l'application seront attachées les unes par rapport aux autres via certains attributs (un utilisateur sera lié à une société, un appel d'offre à un utilisateur, une offre à un appel, etc) nous opterons pour une base de donnée relationnelle.

Ce genre de base de données se doit de garantir l'intégrité référentielle, c'est à dire que pour chaque information d'une table A faisant référence à une information d'une table B, l'information référencée existe dans la table B. Par exemple si chaque ligne de la table utilisateur possède un numéro de société celui-ci doit exister dans la table société. De même toute modification violant cette intégrité devra être interdite ou paramétrée afin de garantir cette intégrité. Par exemple une suppression en cascade peut être configurée afin que la suppression d'un enregistrement de la table B entraîne la suppression des enregistrements de la table A contenant des informations faisant référence à ce dernier.

Afin de gérer cette base de données nous utiliserons un système de gestion de base de données (SGBD) MySQL. Cependant MySQL ne disposant pas par défaut de support d'intégrité automatique des tables, un moteur de stockage InnoDB devra être installé afin de garantir ce dernier. Un moteur de stockage d'une base de donnée est un composant logiciel contrôlant, lisant, enregistrant et triant les informations dans une ou plusieurs bases de données. InnoDB est un moteur de stockage pour le SGBDR MySQL permettant de gérer l'intégrité référentielle et donc les clefs étrangères.

4.3.2 Sécurité [Pan04]

La sécurité des données est également une des grandes bases dans le cadre du développement d'une application web. Quatre conditions indispensables sont nécessaires afin de garantir une

sécurité forte :

- L'utilisation de fonctions de hachage afin de protéger les données sensibles
- L'utilisation d'un certificat serveur
- L'obligation pour l'utilisateur de fournir un mot de passe correct
- Une bonne configuration du serveur web

Nous reviendrons dans cette partie sur les deux premières conditions et les technologies utilisables afin de les garantir. La troisième condition est facilement implémentable dans l'application en utilisant une fonction vérifiant si le mot de passe fourni correspond à des normes définies (longueur minimale, utilisation de plusieurs alphabets, etc).

La quatrième condition étant résolue en désactivant les services non utilisés du serveur web, en paramétrant les permissions sur les répertoires de la racine du serveur et en limitant le nombre de connections simultanées afin d'empêcher une attaque de type "dénier de service". Ces paramètres ne sont pas les seuls à configurer sur le serveur afin de se protéger d'une attaque mais les énumérer tous nous éloignerait du cadre de ce mémoire. En effet le serveur utilisé sera sans doute différent dans l'implémentation finale. De plus ces paramètres étant modifiables via l'interface du serveur il n'est pas bien difficile d'effectuer ces changements.

Fonctions de hachage

Les données sensibles se situant dans la base de données doivent obligatoirement être protégées afin qu'en cas de problème, comme une intrusion dans la base de données, ces dernières ne soient pas utilisables par l'auteur de l'intrusion. Afin de garantir cette protection nous allons utiliser une fonction de hachage permettant de transformer une séquence binaire de taille quelconque finie en une séquence de taille fixe n (appelé empreinte). Dans le but de garantir un hachage de qualité nous pouvons utiliser la fonction de hachage MD5 disponible via la librairie *security* de Java.

Afin de renforcer MD5 nous pouvons également utiliser un sel : le salage est le fait d'injecter une chaîne de caractère supplémentaire, avant et/ou après le mot de passe non haché, afin d'obtenir une chaîne de caractère plus longue. Après avoir "salé" le mot de passe on peut appliquer la fonction de hachage sur le résultat.

Une autre fonction de hachage intéressante et employée couramment est SHA-256 également disponible via la librairie *security* de Java. Nous pouvons également appliquer un salage avant de l'utiliser.

Ce qui est stocké dans la base de données n'est donc plus le mot de passe en clair mais le résultat du hachage du mot de passe salé. Une fois que l'utilisateur veut se connecter on resale son mot de passe avant de le hacher et on compare l'empreinte obtenue et l'empreinte stockée en base de données. Si elle est identique le mot de passe est correct.

La seule problématique pouvant survenir lorsqu'on utilise les fonctions de hachage est qu'une collision puisse arriver : ce terme désigne une situation où deux séquences initiales ont la même empreinte après la même fonction de hachage. Cependant une telle situation peut être considérée comme assez rare dans le cadre d'un site web.

Afin de crypter les mots de passe de l'application nous utiliserons la fonction de hachage SHA-256 appliquée au mot de passe salé avec le nom de l'utilisateur. Ainsi chaque personne aura un salage personnel afin que ce dernier ne soit pas trop reconnaissable.

La figure 4.3 illustrant le hachage du mot de passe avec la SHA-256 :




	ID	Societe_ID	Nom	Prenom	Mail	Login	Mdp	Telephone	Poste	Accepte
  	1	1	Aupaix	Xavier	aupaixxavier@gmail.com	xaupaix	688f151cb02bc70c8699203959b9557ff2d1ac3b	0499231383	admin	1

FIGURE 4.3 – Table "utilisateur" de la base de données contenant une entrée avec mot de passe haché avec la fonction SHA-256

Utilisation d'un certificat serveur [Moo07]

Afin d'établir une identité sur les réseaux des certificats numériques sont utilisés. Ces derniers permettent aux utilisateurs et aux serveurs web de s'authentifier mutuellement sur le réseau. Les certificats contiennent également des valeurs de cryptage, appelées clés, qui permettent d'établir une connexion SSL (Secure Sockets Layer) entre le client et le serveur. Les informations envoyées par l'intermédiaire de cette connexion (par exemple, un numéro de carte de crédit) sont cryptées et ne peuvent donc pas être utilisées par des personnes non autorisées. Les certificats serveurs sont des identifications numériques contenant des informations sur un serveur Web. Un certificat serveur permet aux utilisateurs d'authentifier un serveur, de vérifier la validité du site Web et d'établir une connexion sécurisée. Le certificat serveur contient également une clé publique utilisée pour établir une connexion sécurisée entre le client et le serveur. Ces certificats peuvent être obtenus de deux manières : soit le créateur du site web peut auto-signer ses certificats SSL soit il peut en demander un à une organisation tierce appelée autorité de certification moyennant un paiement annuel. Si l'avantage de la gratuité est intéressant le fait d'auto-signer ses certificats

n'est pas avantageux dans le cadre du développement d'un marketplace. En effet le rôle principal d'une autorité de certification est de confirmer l'identité des personnes demandant un certificat, garantissant ainsi la validité des informations d'authentification contenue dans le certificat. Ainsi le visiteur du site web sait qu'il peut avoir confiance et ainsi enregistrer ses coordonnées et ses données sensibles comme des informations sur sa carte de crédit. Au contraire si le certificat est auto-signé le navigateur indiquera au visiteur que le certificat n'est pas certifié par une autorité de certification ce qui le poussera à se méfier. Cela peut mener à une fréquentation basse d'un site ce qui serait inintéressant.

Le certificat auto-signé, bien que facile à implémenter ², ne sera pas installé sur notre serveur. En effet, l'application devra donner un sentiment de confiance afin que les sociétés s'inscrivent. Pour ce faire un certificat délivré par une autorité de certification sera plus approprié afin que les navigateur web préviennent l'utilisateur qu'il peu avoir confiance dans le site qu'il visite.

Conclusion

Dans ce chapitre nous avons exposé les différentes recherches techniques effectuées dans le cadre de ce mémoire. La base de toute cette démarche était de se reposer sur des outils de développement fort afin de construire une application web structurée, possédant une architecture en accord avec le pattern MVC. De plus l'étude des à-côtés techniques à montré les choix effectués au niveau de la structure de stockage, de la nécessité de construire des tables de type InnoDB afin de garantir l'intégrité référentielle. Nous avons également mis à la lumière du jour les différentes technologies nécessaires, dans le but de garantir une sécurité la plus évoluée possible, et particulièrement les fonctions de hachage ainsi que les certificats numériques. L'ensemble de ces technologies (mis à part les certificats) seront évidemment réutilisées au moment de l'implémentation.

2. <http://www.linux-france.org/prj/edu/archinet/systeme/ch24s03.html>

Chapitre 5

Développement et implémentation

Introduction

Dans ce chapitre nous présenterons les différentes étapes traversées dans l'ensemble du développement de l'application web. Dans la partie 5.1 nous parlerons des deux architectures importantes : l'architecture de la base de données, premier élément à avoir été construit dans le cadre de ce mémoire et l'architecture finale de l'application, se reposant sur des *Design Patterns* reconnus. Ensuite nous aborderons l'implémentation en 5.2. Dans cette partie nous décrirons les différents packages faisant partie de l'application et détaillerons le fonctionnement du programme via un cas d'utilisation réel. Enfin nous conclurons en parlant des différentes librairies utilisées.

5.1 Architectures

Introduction

Dans cette section nous détaillerons tout d'abord la démarche utilisée afin de construire la base de données. Lors de cette conception une connaissance du projet et de ses concepts sera évidemment nécessaire. Les deux principaux piliers de notre conception seront les méthodes d'aide à la décision PROMÉTHÉE-GAIA et le fonctionnement des marketplaces décrits respectivement dans les chapitres 2 et 3. Ensuite nous passerons à l'architecture générale de notre site web. Nous expliquerons les problèmes rencontrés par rapport à l'architecture de la figure 4.1 et expliquerons les choix effectués afin de garantir une architecture stable possédant des couches indépendantes les unes des autres.

5.1.1 Conception de la base de données [Zim08]

Problème de base

La conception d'une base de données part généralement d'une description poussée du problème. Afin de d'en construire une correctement, reprenons l'énoncé du problème proposé dans l'introduction et voyons les acteurs qui en découlent :

Notre application web devra permettre à des utilisateurs de s'enregistrer, ces-derniers représenteront des sociétés et pourront choisir un des deux rôles suivants en plus du rôle d'acheteur : fournisseur ou expert.

Un acheteur sera un rôle permettant de passer des appels d'offre. Ces derniers devront comporter un certain nombre de rounds allant de 1 à n. De plus, un appel d'offre devra comporter des critères servant à effectuer un tri multicritère parmi les offres reçues. Ces critères devront être définis via une fonction de préférence comme vu en 2.3.2. Chaque critère sera donc décrit par un ou deux paramètres représentant le seuil de préférence et/ou le seuil d'indifférence.

Un fournisseur pourra quant à lui répondre à un appel d'offre si le créateur de ce dernier l'y autorise. Son offre devra répondre aux différents critères établis lors de la création de l'appel.

Enfin un expert pourra regarder les offres proposées et donner une appréciation de la qualité de l'offre afin d'aider l'acheteur dans son choix.

Un système de communication sera mis en place afin que chaque utilisateur puisse envoyer des messages. De plus chaque appel d'offre ou réponse à celui-ci pourra être accompagné d'un fichier le décrivant en détail.

Le modèle entité-association

Afin de garantir une bonne construction de la base de données nous allons établir un modèle entité-relation. Ce dernier sera la fondation de la base de données. En effet nous utiliserons ensuite un logiciel permettant de concevoir visuellement des bases de données, c'est à dire en se basant sur ce modèle entité-association. En se basant sur la description du problème de la section précédente nous sommes à même d'identifier les différents acteurs, les "entités". Nous montrons un exemple d'entités avec ses attributs et ses clefs. La clef principale se trouve être soulignée alors que la clef étrangère est en italique :

Utilisateur : ID, *Societe_ID*, Nom, Prénom, Mail, Login, Mot de passe, Téléphone, Poste, Accepte.

L'attribut "Accepte" est un booléen servant à savoir si l'administrateur du site a validé l'utilisateur, sachant que ce dernier ne peut rien faire avant cette validation. Voyons maintenant un schéma entité-association concernant un utilisateur et une société créée avec le programme DB-

Designer. Nous devons appliquer une cardinalité étant donné que plusieurs utilisateurs peuvent appartenir à la même société. La figure 5.1 nous montre ce schéma :

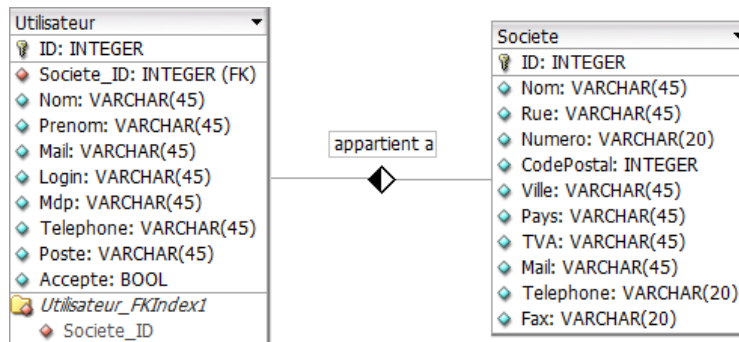


FIGURE 5.1 – Modèle entité-relation d'un cas simple avec les entités utilisateur et société

Si nous généralisons maintenant cette méthode de travail au problème entier nous obtenons alors le modèle entité-association de l'entièreté de notre base de données. Le schéma de ce dernier est donnée en annexe. Les tables sont de type InnoDB afin de garantir l'intégrité référentielle. Le script de création généré sera alors importé dans un système de gestion de base de données afin de créer les différentes tables. La base de données étant désormais construite nous allons maintenant définir l'architecture du programme.

5.1.2 Architecture du marketplace

Comme nous l'avons vu à la section 4.2.1 et plus particulièrement à la figure 4.1 une architecture idéale dans un développement d'une application web se compose d'une couche de présentation fournie par la compilation de fichiers jsp, d'une couche de contrôle implémentée par les servlet et d'une couche métier fournies par les EJB. Comme nous l'avons vu les EJB doivent être déployés sur un serveur possédant un conteneur d'EJB. Un tel serveur peut se trouver en version libre comme le serveur GlassFish, spécialement conçu pour le déploiement d'applications J2EE.

Ayant débuté dans l'optique de développer via ce serveur nous avons cependant dû nous heurter à de sérieux problèmes de mise en place : problèmes de compatibilité entre la version de Glassfish et la version de notre IDE (Netbeans), problème de déploiement des EJB sur le serveur retournant des rapports d'erreur (Stack Trace) vides, des erreurs de compilation ne trouvant aucunes réponses sur le web, etc.

Partant du constat simple qu'une application web J2EE pouvait cependant se construire sans

utiliser les EJB et que chaque couche serait, de par la méthode de développement, indépendante du reste du programme nous avons décidé de développer l'application web en utilisant uniquement des JavaBeans normaux et des classes d'accès à la base de données. Pour cela nous avons développé une architecture respectant le paradigme MVC en utilisant des pattern appropriés afin d'augmenter la souplesse et la robustesse de notre programme. Nous expliquons ci-dessous les pattern utilisés et leur utilité.

Le pattern DAO [Mic02]

Ce pattern signifiant Data Access Object permet de faire le lien entre la couche d'accès aux données et la couche métier d'une application. Son but est de permettre aux programmeurs une meilleure maîtrise sur les changements susceptibles d'être opérés sur les systèmes de stockage des données et par extension d'avoir une meilleure évolutivité en cas de migration d'un système à un autre. Afin de permettre cela nous séparerons l'accès aux données (base de données SQL) et aux objets métier (JavaBeans ou Plain Old Java Object (POJO)). Les JavaBeans seront donc utilisés afin de manipuler les objets de la base de données. Afin de réaliser cela voyons la structure du Pattern DAO : imaginons que nous avons deux tables Utilisateur et Societe dans notre base de données. Nous aurons donc une classe abstraite générique et deux classes implémentant celle-ci comme montré à la figure 5.2 :

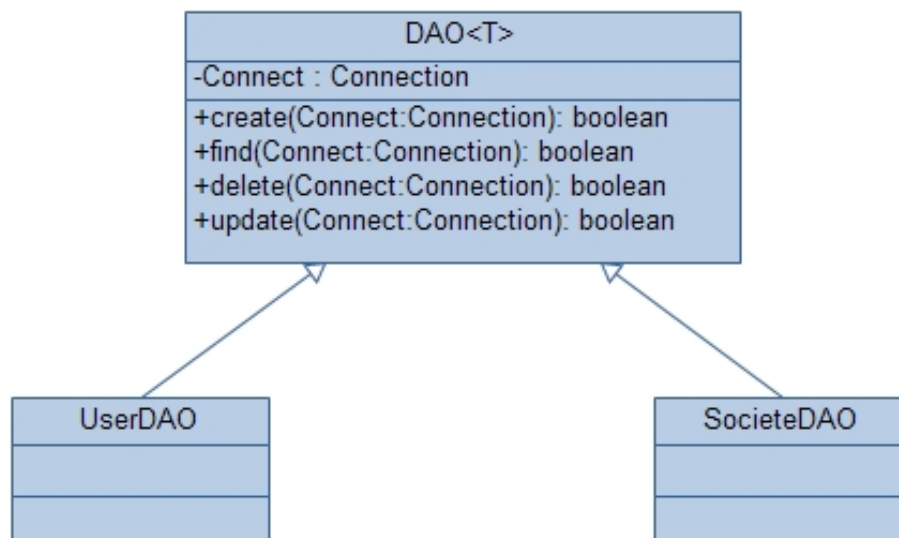


FIGURE 5.2 – Exemple d'implémentation du pattern DAO via l'utilisation d'une classe générique

Ce pattern nous permettra d'accéder à notre base de données via une classe abstraite mettant en oeuvre les différentes méthodes que l'on désire. Ces méthodes seront redéfinies par les classes héritant de la classe abstraite afin d'assurer le polymorphisme. L'étape suivante sera de construire une classe permettant d'instancier les objets héritant de la classe abstraite DAO.

Cela se fera grâce au pattern factory.

Le pattern factory

Ce pattern est souvent considéré comme un complément logique du pattern DAO. La classe construite afin d'implémenter ce pattern est nommé "une fabrique" et se charge de construire une instance de l'objet désiré. Nous pouvons voir via le schéma 5.3 comment fonctionne ce pattern :

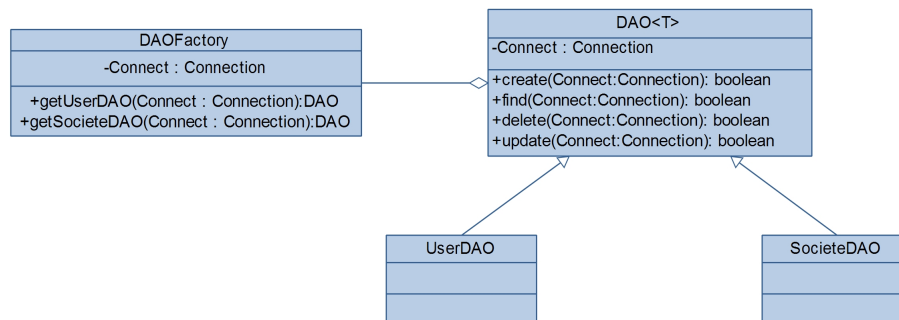


FIGURE 5.3 – Exemple d'implémentation du pattern factory

Lors de la conception de ce mémoire nous avons voulu avoir la meilleure structure dans le cadre d'une évolution future de l'application web. Étant donné que dans un futur proche le programme sera peut-être amené à construire des fichiers XML afin de communiquer avec D-Sight nous avons également voulu que ce système de sauvegarde de données soit facilement implémentable. Pour cela nous avons décidé d'implémenter la classe DAOXMLFactory afin de pouvoir construire des instances de classes héritant de la classe abstraite XMLDAO. Pour cela nous avons implémenté une classe abstraite AbstractDAOFactory dont hérite la classe DAOFactory vue précédemment et la classe XMLDAOFactory. Nous avons donc implémenté une fabrique de fabrique, permettant de choisir facilement entre un DAO portant sur le XML ou un DAO d'accès à une base de données.

Maintenant que les principaux pattern ont été introduits nous pouvons dès maintenant construire l'architecture générale de notre site.

Architecture globale de l'application web

Notre architecture finale est représentée à la figure 5.4. On y voit les différentes étapes franchies afin de réaliser une requête nécessitant un accès à la base de données : La requête est reçu par Tomcat, comme vu en 4.2.7, qui appelle à son tour la servlet nécessaire au traitement. Cette servlet initialise zéro, un ou des JavaBeans selon la requête. Une fois cela effectué la servlet appelle la classe DAOFactory et lui demande de créer une instance de la classe DAO

correspondante. Le résultat de la requête effectuée par la classe DAO est sauvegardé dans les JavaBeans qui sont injectés dans la page jsp. Le fichier jsp est compilé afin de construire le fichier HTML qui est renvoyé vers le navigateur de l'utilisateur.

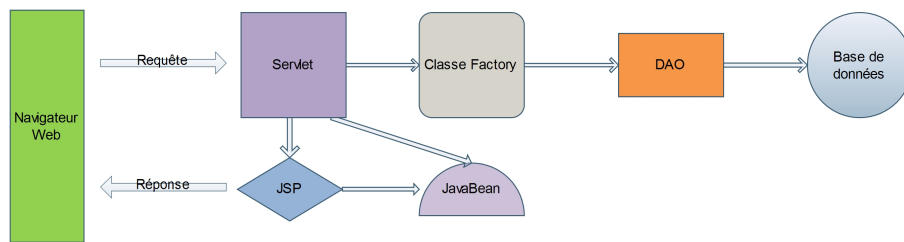


FIGURE 5.4 – Architecture finale de l'application web

Conclusions

Au final l'architecture de l'application n'est pas la plus complète pouvant être créée avec J2EE. Cependant devant les problèmes rencontrés avec le serveur d'application GlassFish lors de l'implémentation des EJB nous avons dû nous y contraindre. Cependant l'application web étant bien structurée, flexible et composée de modules indépendants, une implémentation future ne posera cependant pas de problèmes. Nous avons vu que l'architecture de l'application reposait sur l'utilisation de deux *Design Patterns* importants et permettant d'accéder à la base de données tout en l'isolant du reste du programme.

5.2 Implémentation

Introduction

Dans ce chapitre nous expliquerons le fonctionnement de l'application en utilisant un exemple de cas d'utilisation. Dans la partie 5.2.1 nous introduirons les différents packages créés afin de séparer les différentes classes intervenant dans l'application. Nous détaillerons également un cas d'utilisation afin d'expliquer une partie du fonctionnement de l'application. Enfin dans la dernière partie 5.2.2 nous parlerons de la communication avec D-Sight et des différentes bibliothèques utilisées dans la conception de l'application.

5.2.1 Packages et exemple de fonctionnement

Dans cette partie nous partirons d'un cas d'utilisation, la connexion et la création d'un appel d'offre par un utilisateur, afin de découvrir la structure de l'application. Avant cela nous

présenterons les différents packages créés dans l'application. Ces derniers sont au nombre de 5 :

- Le package Bean : il comporte l'ensemble des Javabeans. Etant donné qu'un bean est censé représenter une table de la base de données nous aurons donc autant de bean que de tables, les attributs de chacun d'entre eux correspondant aux différentes colonnes des tables.
- Le package Connexion : il n'est composé que d'une classe servant à établir la connexion avec la base de données à l'aide d'un pilote nommé JDBC (Java DataBase Connectivity). Nous reviendrons sur ce dernier plus loin dans ce chapitre.
- Le package DAO : comme expliqué au chapitre précédent ce package comporte les classes DAO ainsi que les classes factory. De même que pour les Javabeans un DAO existe pour chaque table présente dans la base de données.
- Le package servlet : il comporte l'ensemble des servlets utilisées (les contrôleurs) afin de recevoir les requêtes et d'y répondre.
- Le package protection : il possède la classe SHA.java permettant le hachage de chaînes de caractères.

L'ensemble de ces packages est complété par l'ensemble des pages jsp se trouvant dans le répertoire approprié du projet Netbeans. Nous allons maintenant voir le fonctionnement du tout via un cas d'utilisation.

Cas d'utilisation

Afin de concevoir le marketplace nous avons également établis les différents cas d'utilisation ainsi que les diagrammes de cas d'utilisation correspondant. Un de ces premiers *Use Case* possibles auxquels nous avons dû faire face était qu'un utilisateur inscrit en tant qu'acheteur puisse passer un appel d'offre après s'être connecté. Une fois cet appel passé il pourra accorder des autorisations aux autres utilisateurs afin de permettre ou non une interaction avec cet appel (proposer une offre pour un fournisseur, noter les offres pour un expert, etc). Nous allons voir maintenant le fonctionnement logiciel d'un tel scénario.

Connexion : l'utilisateur arrive sur le site et est directement dirigé par le serveur sur la page d'accueil Index.jsp, définie dans le fichier de configuration. Cette dernière ne comporte que des

inclusions d'autres pages. En effet, pour pouvoir faciliter la programmation des pages, nous avons découpé la structure de celles-ci en plusieurs parties :

- Header.jsp : ce fichier charge l'en-tête des pages, incluant le menu de l'utilisateur. Étant donné que le menu est dynamique, c'est à dire qu'il comporte des onglets différents selon le type de rôle joué par l'utilisateur, nous ne pouvons le générer à partir d'un page jsp car cela reviendrait à faire un test sur le type d'utilisateur dans le code de la couche affichage. Afin de contrer cela nous incluons dans le fichier Header un appel à la servlet *CheckUser* afin que ce contrôle se déroule au sein d'une servlet.
- Identification.jsp : Ce jsp comprend la partie de page permettant de se connecter au site.
- Welcome.jsp : contient le message d'accueil du site.
- Footer.jsp : contient le pied de page du site.

L'avantage de cette découpe est que les fichiers sont plus clair en cas de modifications (pour changer le message d'accueil par exemple).

Une fois l'utilisateur sur cette page il peut entrer son nom d'utilisateur et son mot de passe sur le côté gauche de celle-ci. Après avoir appuyé sur le bouton "connexion" une requête est envoyée à Tomcat. Afin de la traiter celui-ci appelle la servlet requise ("Login") qui se charge à son tour de récupérer les informations fournies par l'utilisateur, d'instancier un objet *UserDAO* afin de récupérer, via les méthodes adéquates, les informations à propos de l'utilisateur. Une fois cela effectué la servlet compare les mots de passes (leur empreinte comme vu en 4.3.2). Si ils sont différents la servlet appelle la page *Erreur.jsp* et lui passe en paramètre le message d'erreur adéquat. Si au contraire les deux empreintes sont identiques la servlet met à jour le *JavaBean* user se trouvant en session et appelle la page *Index.jsp*. Cette dernière s'affiche à nouveau mais, voyant que l'utilisateur est connecté charge la frame *connected.jsp* à la place d'*identification.jsp*.

Création d'un appel d'offre Comme l'utilisateur a le rôle d'acheteur le menu lui propose maintenant un onglet "Appel d'offre" lui permettant d'en créer un. Une fois sélectionné, une page propose à l'utilisateur une page où il peut indiquer le nombre de rounds et de critères souhaités pour son appel d'offre. Cette page est appelée d'une servlet dont le seul but est de vérifier si l'utilisateur a la permission d'y accéder.

Une fois cela choisi, l'utilisateur passe encore par une page intermédiaire permettant de

choisir une fonction de préférence pour chaque critère choisi. Enfin il arrive sur la page de création d'appel d'offre ou des champs relatifs à celle-ci lui sont proposés. En fonction des fonctions de préférence choisies il aura aucun, un ou deux paramètres à introduire par critère. L'utilisateur aura également la possibilité de joindre un fichier pdf afin de donner une description précise de son offre. Il devra également indiquer la date de livraison souhaitée ainsi que la date limite de chaque round.

Après cela il pourra valider son appel, la requête est alors transmise à la servlet correspondante qui va utiliser les objets *DAO* afin d'ajouter un ou des enregistrements aux tables *appel*, *critere*, *description_appel*, *parametre* et *enrollment*. Son rôle sera également d'envoyer le fichier joint sur le serveur afin que les fournisseurs puissent le télécharger.

Cet exemple nous a montré comme le programme se comporte généralement. Cependant, certaines situations, comme le fait d'envoyer un fichier PDF sur le serveur, demanderons l'utilisation de bibliothèques externes afin d'assurer un bon fonctionnement.

5.2.2 Bibliothèques utilisées

Les bibliothèques externes de Java sont nombreuses et permettent généralement de simplifier la programmation d'une tâche qui pourrait être laborieuse à la base. Les différentes bibliothèques utilisées dans le cadre de ce mémoire sont les suivantes :

- D-Sight : La bibliothèque fournie par le service CoDE-SMG de l'ULB et incluant les différents packages, classes et méthodes du logiciel D-Sight. Cette bibliothèque fût l'outil principal afin de ranger les différentes offres à l'aide des méthodes multicritère PROMÉTHÉE
- JfreeChart : cette API open source fût utilisée afin de créer des graphiques représentant les flux nets par action calculés au travers de la méthode PROMÉTHÉE de la bibliothèque D-Sight.
- FileUpload : Cette bibliothèque permet de sauvegarder des fichiers sur le serveur. Dans notre application, son utilité est de sauvegarder les fichiers de description des appels d'offres ainsi que des offres émises. Le nom du fichier est sauvegardé dans la table *Description..Appel* ou *Description..Offre* afin de le retrouver facilement via un chemin relatif. Cette solution a été choisie afin de ne pas surcharger le serveur et le SGBD. En effet une autre méthode aurait été de stocker les fichiers dans la base de donnée à l'aide d'une transformation en Binary Large Object (BLOB).

- JSTL : Cette librairie de balises est d’une utilité cruciale afin de se passer de code java au sein des pages jsp (scriptlet). Nous avons déjà traité de cet API dans la section 4.2.6.

conclusions

Cette section a permis de comprendre en détail le fonctionnement interne de l’application web créée. Étant donnée que parler du code d’une application informatique est assez contraignant nous avons préféré nous focaliser sur deux points. Dans la première partie nous avons décrit les deux architecture principales de toute application web : celle de la base de donnée, permettant de sauvegarder les données de l’utilisateur et celle de l’application, permettant de programmer en se reposant sur une structure stable et flexible. Dans la seconde partie nous avons tenté de donner un aperçu du fonctionnement du programme via un cas d’utilisation réel et utilisant un grand nombre de mécanismes inhérent à l’application. Enfin nous avons décrit les différentes parties de notre architecture via les packages et les librairies.

Chapitre 6

Conclusions

L'objectif initial de ce mémoire était de procéder à une analyse des places de marché virtuelles afin d'en concevoir un prototype utilisant la méthodologie d'aide à la décision PROMÉTHÉE-GAIA. Cela sous-entendait évidemment de s'appuyer sur une étude de ces dites méthodes afin de concevoir une architecture générale de l'application et de la base de données.

La méthode de résolution suivie fut de commencer par cet aspect analytique du problème afin d'assimiler les différents concepts importants. Rapidement le développement commença en parallèle de cette étude. La première étape franchie dans le cadre du développement fut de choisir un langage de programmation répondant aux attentes de la spin-off Decision Sights. Après cela une étude de la spécification J2EE fut menée. Bien que compliquée à la base les technologies utilisées furent assimilées au fur et à mesure.

Le problème le plus important rencontré au cours de ce travail fut l'impossibilité d'implémenter les EJB. Suite à ce problème nous avons migré vers le serveur d'application *Apache-Tomcat* afin de poursuivre le développement. Enfin nous avons terminé la modélisation de la base de données. Une fois l'architecture de l'application également fixée nous avons pu commencer le développement du programme.

Ce mémoire nous aura permis de travailler au sein d'un environnement professionnel, encourageant un développement en spirale. En effet les différentes phases de ce type de développement se sont retrouvées dans ce mémoire, les rendez-vous réguliers permettant de déterminer de nouveaux objectifs à implémenter.

L'application élaborée est fonctionnelle (des captures d'écran se trouvent en annexe) et peut être

déployée sur un serveur d'application. Elle n'implémente évidemment pas la totalité des fonctionnalités des marketplaces trouvables sur internet. En effet ces sites fournissent un nombre élevé de services. De plus l'implémentation de ce genre d'application prend énormément de temps afin de coder, tester et debugger. Cependant le programme permet déjà d'effectuer un certain nombre d'actions :

- Enregistrement d'utilisateurs
- Choix de rôles spécifiques au sein du marketplace
- Création d'appels d'offres comportant plusieurs critères et rounds
- Envoi de fichiers sur le serveur
- Ajout d'autorisations pour un appel d'offre
- Création d'une offre
- Rangement multicritère des différentes offres et obtention d'un graphique fournissant les flux de surclassement net
- Outil de recherche (utilisateur, produit ou société
- Envoi de messages à un utilisateur

Il conviendra donc de continuer l'implémentation de l'application en ayant deux objectifs. Le premier sera de compléter cette liste afin de pouvoir fournir une plate-forme complète et répondant aux diverses attentes de l'utilisateur. La seconde sera d'améliorer l'architecture du programme. Soit en implémentant la logique métier avec les Enterprise JavaBean, soit en migrant l'application vers une solution tierce comme le serveur d'application *Google App Engine* présenté en 4.1.6 et utilisant J2EE.

Bibliographie

- [All02] Paul Allen. *J2EE : Sun certified enterprise architect for J2EE*. Prentice Hall ; Stg edition, 2002.
- [Ber04] Hugues Bersini. *L'orienté objet : 2ème édition*. Eyrolles, 2004.
- [Gir02] Michael Girdley. *J2EE : Application and BEA WebLogic Server*. Prentice Hall, 2002.
- [Hol08] Adrian Holovaty. *The Definitive Guide to Django : Web Development Done Right*. Apress, 2008.
- [Isa08] Henri Isaac. *E-commerce : de la stratégie à la mise en oeuvre opérationnelle*. Pearson Education, 2008.
- [Jab09] Sami Jaber. *Programmation GWT 2*. Eyrolles, 2009.
- [Keo02] Jim Keogh. *J2EE : The complete reference*. Tata McGraw-Hill, 2002.
- [Mac08] Matthew MacDonald. *Beginning ASP.NET 3.5*. Apress, 2008.
- [Mar02] Bertrand Mareschal. *Prométhé-Gaia : Une méthodologie d'aide à la décision en présence de critères multiples*. Université de Bruxelles, 2002.
- [Mar04] Bruno Martin. *Codage, cryptologie et applications*. Presses polytechniques et universitaires romandes, 2004.
- [May94] Lucien Yves Maystre. *Méthodes multicritères ELECTRE : Description, conseils pratiques et cas d'application à la gestion environnementale*. Presses Polytechniques et Universitaires Romandes, 1994.
- [MH02] Richard Monson-Haefel. *Enterprise javaBeans*. O'Reilly, 2002.
- [Mic02] Sun Microsystems. *Core j2ee patterns - data access object*, 2002.
- [Moo07] Matthew Moodie. *Pro Apache Tomcat 6*. Apress, 2007.
- [Pan04] Raymond Panko. *Sécurité des systèmes d'information et des réseaux*. Pearson Education, 2004.
- [Roy85] Bernard Roy. *Méthodologie Multicritère d'aide à la décision*. Economica, 1985.
- [Roy93] Bernard Roy. *Aide multicritère à la décision : Méthodes et cas*. Economica, 1993.

- [Sch02] Gari Schneider. *Electronic Commerce*. Course Technology, 2002.
- [Sek07] Michael Sekler. *Beginning JSP, JSF and Tomcat Web Development*. Apress, 2007.
- [Sou02] Christian Soutou. *De UML à SQL : Conception de bases de données*. Eyrolles, 2002.
- [Tho07] Dave Thomas. *Agile Web Development with rails*. Pragmatic Bookshelf, 2007.
- [Vin89] Philippe Vincke. *L'aide multicritère à la décision*. Université de Bruxelles, 1989.
- [Zer07] Quentin Zervaas. *Practical Web 2.0 Applications with PHP*. Apress, 2007.
- [Zim08] Esteban Zimányi. *Bases de données*. Université Libre de Bruxelles, 2008.
- [Zim09] Esteban Zimányi. *Analyse et conception par objet*. Université Libre de Bruxelles, 2009.

Annexes

Analyse en composantes principales

Les détails de ce calcul proviennent en totalité de [Mar02]. Le raisonnement de base est le suivant : on essaye d'abord de projeter les points α_i sur une direction matérialisée dans R^k par un vecteur unité u centré à l'origine. Ce vecteur doit être celui qui minimise la somme des carrées des écarts entre les points α_i et leurs projections p_i .

$$\text{Min} \sum_{i=1}^n |\alpha_i p_i|^2$$

équivalent à

$$\text{Max} \sum_{i=1}^n |Op_i|^2$$

via les relations :

$$|Op_i|^2 + |\alpha_i p_i|^2 = |O\alpha_i|^2 \text{ et } \sum_{i=1}^n |O\alpha_i|^2 = \text{constante}$$

Or sachant que u est un vecteur unité on a :

$$|Op_i| = \alpha'_i u, \quad (u'u=1)$$

Il résulte que le problème peut se ramener à :

$$\begin{cases} \text{Max } u'Cu \\ u'u - 1 = 0 \end{cases}$$

où $n \times n$ $C = \Phi' \Phi$, C étant la matrice de variance-covariance des ϕ_j . Étant donné que la fonction lagrangienne associée au problème est

$$L(u, \lambda) = u'Cu - \lambda(u'u - 1)$$

il convient de résoudre le système

$$\begin{cases} Cu = \lambda u \\ u'u = 1 \end{cases}$$

Algébriquement il en résulte que λ est une valeur propre de la matrice C et u un vecteur propre unité associé à λ . Si nous appelons λ_1 la plus grande de ces valeurs propres on peut voir que :

$$\text{Max } u'Cu = \lambda_1$$

La solution du problème est donc fournie par un vecteur propre unité correspondant à la plus grande valeur propre de la matrice C .

Considérons à présent la deuxième plus grande valeur propre λ_2 et un vecteur propre unité v associé. Il est aisé de vérifier que $u \perp v$ et que ces deux vecteurs détermineront le meilleur plan sur lequel peuvent être projetés les points du nuage. De plus :

$$\sum_{i=1}^n |O\alpha_i|^2 = \sum_{i=1}^n \|\alpha_i\|^2 = \text{Trace}(C) = \sum_{j=1}^k \lambda_j$$

Or comme :

$$\begin{cases} \sum_{i=1}^n |Op_i|^2 = u'Cu = \lambda_1 \\ \sum_{i=1}^n |Oq_i|^2 = v' Cv = \lambda_2 \end{cases}$$

où p_i et q_i ($i=1,2,\dots,n$) sont respectivement les projections des points α_i sur les vecteurs u et v , la quantité :

$$\delta = \frac{\lambda_1 + \lambda_2}{\sum_{j=1}^k \lambda_j}$$

définit le pourcentage de l'inertie totale qui est expliqué par le plan (u,v) . δ mesure donc la quantité d'information préservée par la projection. Appelons A_i les projections des points α_i sur le plan (u,v) . Ces points A_i ont comme coordonnées :

$$\begin{cases} |Op_i| = \alpha'_i u \\ |Oq_i| = \alpha'_i v \end{cases}$$

De façon générale, les coordonnées de la projection d'un point quelconque x de R^k sur le plan (u,v) sont données par $(x'u)$ et $(x'v)$.

Modèle entité-association de la base de données

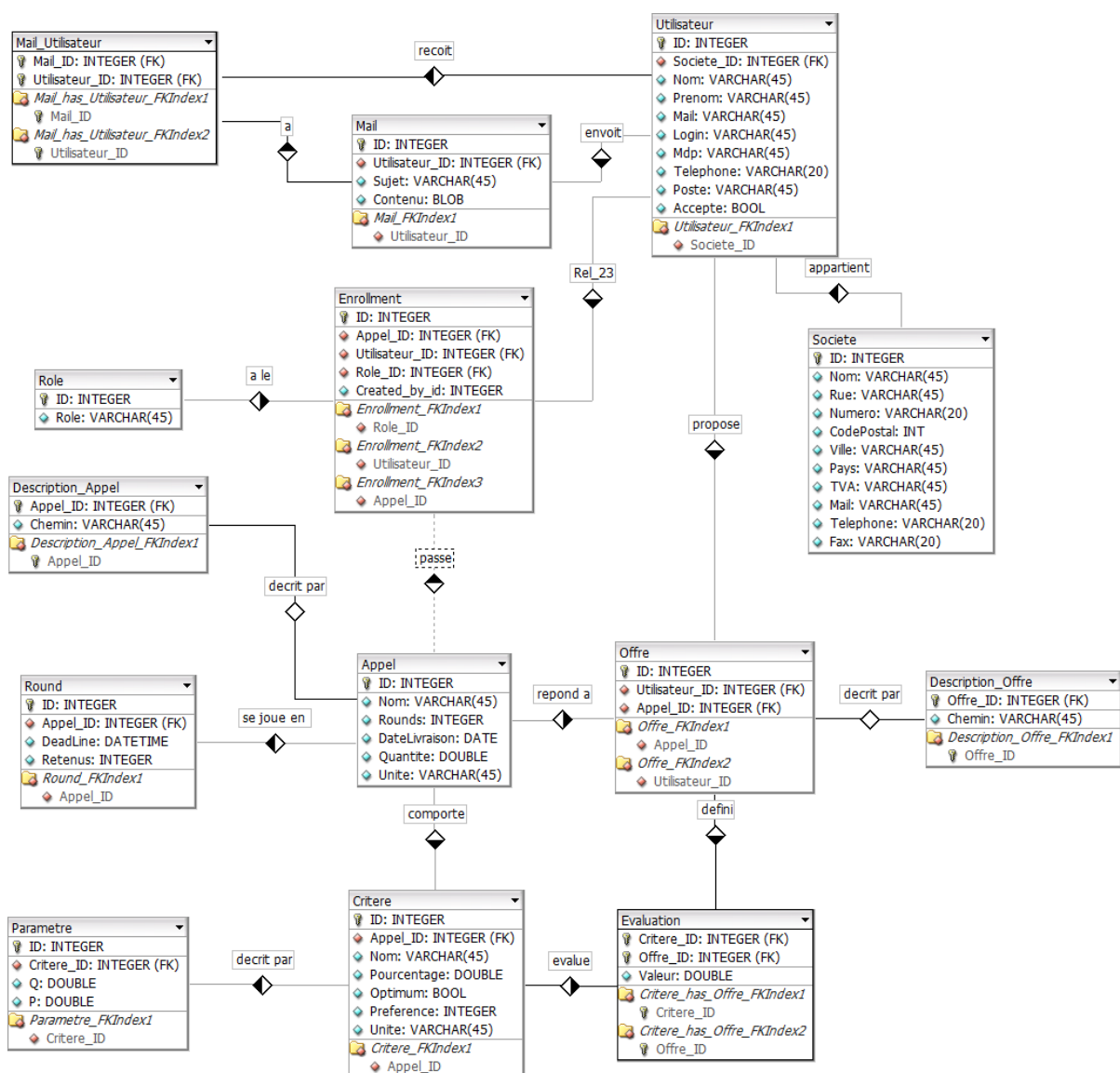


FIGURE 6.1 – Modèle entité-relation de la base de données

Capture d'écran de l'application web

Bienvenue sur D-Sight.be
MarketPlace basé sur la méthodologie Prométhée-Gaia

Accueil Inscription Conditions d'utilisation

Connexion

Identifiant:

Mot de passe:

Se connecter

Inscription

Informations générales

Identifiant:

Mot de passe:

Nom:

Prénom:

E-mail:

société:

Poste:

Numéro de téléphone:

Roles joué en plus qu'acheteur

Vous apparaitrez dans la liste des fournisseurs et experts selon la (les) case(s) cochées

Fournisseur: ☐

Expert: ☐

Autres informations

J'accepte les conditions générales: ☐
(Consultez les conditions)

FIGURE 6.2 – Page d'inscription du marketplace

Bienvenue sur D-Sight.be
MarketPlace basé sur la méthodologie Prométhée-Gaia

Accueil Appel d'offre Mes appels d'offres Moteur de recherche Conditions d'utilisation

Connexion

Vous êtes connecté

Logout

Profil

Utilisateur: Harry Dupont

Voir mon profil

Passer un appel d'offre

Veuillez tout d'abord choisir le nombre de rounds et le nombre de critères désirés;

Nombre de rounds:
(Plus d'informations sur les rounds)

Nombre de critères:
(Plus d'informations sur les critères)

Valider

Accueil Appel d'offre Mes appels d'offres Voir appels d'offre Moteur de recherche Conditions d'utilisation

Connexion

Vous êtes connecté

Logout

Profil

Utilisateur: Bertrand Dubois

Voir mon profil

Formulaire de réponse à un appel d'offre

Informations générales

Nom: Dubois
Prénom: Bertrand
Téléphone: 0472137653
Mail: bdubois@Societe2.com
Poste: Chargé des ventes
Entreprise: Société 2

Renseignements sur l'offre

Produit demandé: Carte graphique
Quantité: 100.0 Pièces

Critère 1 : Mémoire

Pourcentage du critère: 30.0
Optimisation du critère: Maximiser
Votre offre: Mo

Critère 2 : Fréquence

Pourcentage du critère: 35.0
Optimisation du critère: Maximiser
Votre offre: Mhz

Critère 3 : Prix

Pourcentage du critère: 35.0
Optimisation du critère: Minimiser

FIGURE 6.4 – Création d'une offre dans le marketplace

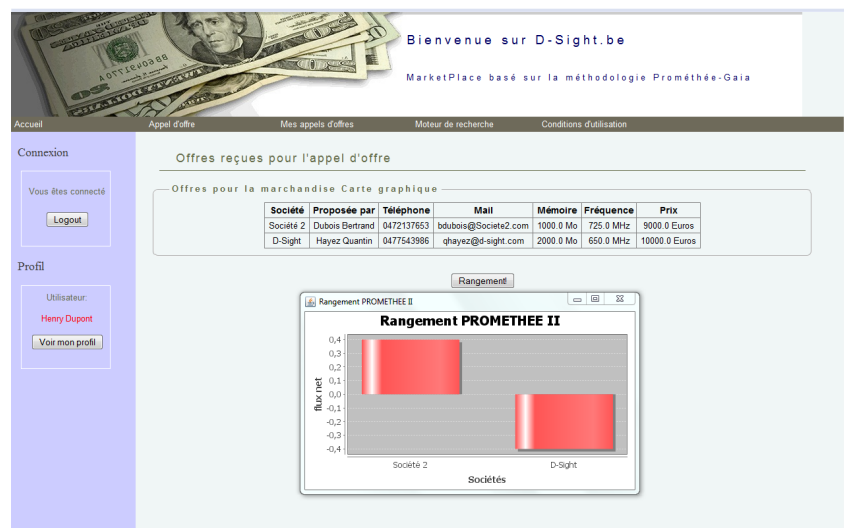


FIGURE 6.5 – Graphique résultant de la demande de rangement dans le marketplace